MODULO: A GENERATIVE AUDIO WORKSTATION FOR AI-NATIVE MUSIC CO-CREATION

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Master of Science

in

Engineering Sciences

by Taka Khoo

Thayer School of Engineering
Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

May 2026

Examining Committee:

Chairman Signed by: Peter Chin 6C2D735E650D4A2... Peter Chin, Ph.D.

Member Signed by: Eugene Santos C771426EFBF647B... Eugene Santos, Ph.D.

Member Signed by: Nikhil Singh A3F25BBFB6594BD... Nikhil Singh, Ph.D.

F. Jon Kull, Ph.D.
Dean of Guarini School of Graduate and Advanced Studies

ABSTRACT

Generative music has reached an inflection point. AI systems turn text prompts into album-quality audio in seconds, and the browser has made music production one click away. What these systems cannot do is collaborate at the level musicians actually think. True artistry lies in chord theory, polyphony, subjective nuance, and the high-level musical reasoning that spans every harmonic choice and mix decision.

This thesis presents MODULO — Musician-Owned DAW for User-Led Orchestration —, an AI-native desktop Generative Audio Workstation. Built in C++ on Tracktion Engine, MODULO embeds structured agentic co-creation directly inside a professional multi-track timeline with Virtual Studio Technology (VST) and Audio Unit (AU) plugin hosting, low-latency audio, and full mixing infrastructure, so that every generated note remains subject to the musician's ear and intent.

In one session MODULO offers parallel customizable chord generation, a Chord Workshop that treats harmony as an editable abstraction, and a rule-based adaptive harmony engine that produces multiple independent voice lines in seconds. An audio-to-MIDI (Musical Instrument Digital Interface) pipeline converts live recordings into editable material, and one-click stem separation opens finished tracks back up for rework. Prompt-conditioned music generation handles full-song synthesis with automatic section-aware stem layout, a sound-effects module fills in non-melodic content, and a composition planner turns free-form intent into section-level blueprints. Underneath sits a parametric-equalization (EQ) mixer with buses, sends, and plugin hosting, wrapped in a musician-facing interaction layer of domain-specific hotkeys and contextual affordances.

Five within-subjects user studies with working musicians drawn from both university and industry settings, four at $N{=}25$ and one at $N{=}10$, validated the system using the NASA

Task Load Index (NASA-TLX), the User Engagement Scale – Short Form (UES-SF), the System Usability Scale (SUS), and paired t-tests with Benjamini–Hochberg correction. Treatment arms cut time-to-satisfaction by 50–89%, reduced NASA-TLX workload by 47–74%, and drove forced-choice preference to 64–100% across every measured dimension (§14.3). A final integrated evaluation with five professionals returned a median SUS of 85.0 (§13.9), well above the 68-point acceptability threshold.

MODULO argues that the contribution lies in the *joint* design of model and interface: a generative model is only useful inside a DAW if the surrounding interface lets a musician audition, compare, and selectively commit; an interface that promises agency is only useful if the underlying model produces meaningfully diverse, theory-aware candidates for the musician to choose between. Each subsystem in this thesis pairs a specific modeling strategy with a specific interaction paradigm, and the empirical claims belong to that pairing, not to either component in isolation. With these joint design strategies, generative power and musical agency are not in tension but mutually reinforcing.

PREFACE

ACKNOWLEDGEMENTS

This thesis is the product of a decade of chasing the intersection of engineering and music, and it would not exist without the people who believed that intersection was worth pursuing.

My deepest gratitude goes to my thesis advisor, Dr. Peter Chin, director of the Learning, Intelligence + Signal Processing (LISP) Lab at Dartmouth. Peter took a chance on me as an undergraduate researcher, funded my work, gave me the intellectual freedom to build something ambitious, and then offered me a position in his lab to continue it as a Master's student. His mentorship shaped not only the technical direction of this thesis, but also the way I think about systems: rigorously, honestly, and with an eye toward what actually works. The breadth of his research, from game-theoretic reinforcement learning to signal processing and computational neuroscience, taught me to treat "systems thinking" as a first-class discipline. The LISP Lab became my home at Dartmouth, and the foundation of everything presented in these pages was built within it.

I owe an equally profound debt to Dr. Nikhil Singh, my secondary advisor and director of the SAHAS Lab (Science and Art of Human-AI Systems), who introduced me to the human-computer interaction space and challenged me to look beyond pure technical deep learning. Nikhil's paper *Discovering and Steering Interpretable Concepts in Large Generative Music Models*, which uses sparse autoencoders to extract and steer interpretable features from the residual stream of an autoregressive music transformer, was the work that most directly shaped the opening direction of my graduate research. A Berklee-trained composer as well as an MIT-trained researcher, Nikhil pushed me to think about creativity as a design problem, not a benchmark, and his mentorship transformed the way I approach

research.

I thank Dr. Michael Casey, Professor of Computer Science and Music at Dartmouth, whose work at the intersection of creative AI, music theory, and cognitive neuroscience inspired the earliest directions of my research. Michael was the primary advisor on my music honors thesis and the secondary advisor on my engineering honors thesis, and his courses in computational music gave me the vocabulary to reason formally about sound. His ISMIR publications and his broader program in music, neuroscience, and medicine remain a model of what interdisciplinary research can look like when done at full intensity. His influence runs through every chapter of this work.

I am grateful to Professor Sang-Wook Nam, Grammy-nominated mastering engineer and the secondary advisor on my music honors thesis, *My Life in Sound*, a professionally composed, performed, and mastered multi-genre album with accompanying manuscript. From my freshman year I worked in his studio learning the anatomy of professional post-production: VU metering, frequency masking, dynamic-range sculpting, and the way subtle changes in signal flow shape a listening experience. His teaching bridged artistic sensitivity and technical fluency, and his standards for sonic quality remain the benchmark against which I measure my own work.

Two peers deserve special recognition. Matt Keating and Noah Schaffer are both doctoral students in Nikhil Singh's SAHAS Lab and have been my closest intellectual companions throughout this journey. Matt's research on algorithmic jazz-guitar voice leading sits squarely in the same territory as the harmonic subsystems of MODULO, and Noah's work on control protocols for music generation and on generative audio editing speaks directly to the questions of authorship that drive this thesis. The three of us started the AI Music Reading Group at Dartmouth and have spent countless hours dissecting papers on generative models, music representation, and what it means for a musician to keep authorship over a generative system. They are the two people I most trust to challenge my ideas, and the two I most want to play with afterward.

I thank the musicians who participated in the user study, generously donating their time, creative energy, and candid feedback to evaluate a system that was still rough around the edges. Their enthusiasm validated the premise of this work more than any metric could.

I thank the faculty and staff of the Thayer School of Engineering and the Department of Music at Dartmouth College for fostering an environment where a student could pursue simultaneous honors theses across both departments, and for never once suggesting that was an unreasonable thing to do.

To Paige, who has been beside me through the late nights, the debugging sessions that stretched past midnight, and the moments of doubt that come with building something this large alone: thank you. Your steady support, patience, and belief in this work kept me going when the scope felt impossible.

To my family (Miya, Kris, and Larry), whose love, sacrifices, and own achievements inspired the long nights I devoted to this work. You put a piano in front of a four-year-old, drove me to every recital and competition, and never questioned the path I chose, even when it meant three theses, startups, and very little sleep. Everything I build starts with what you built in me.

I see this thesis not as a conclusion but as a foundation. The question of how musicians and machines create together is only beginning to be asked seriously, and I intend to spend my career answering it.

PRIOR SUBMISSIONS

A self-contained presentation of the chord-generation and harmony work covered in Chapters 4 through 6 of this thesis was prepared for the ACM Symposium on User Interface Software and Technology, UIST 2026 in Detroit, MI, under the title *Parallel Harmonic Exploration for Musical Accompaniment with MODULO*, with co-authors T. Khoo, P. Chin,

and N. Singh. The conference paper draws on the within-subjects user study reported in Chapter 4, the Chord Workshop interface of Chapter 5, and the rule-based harmony engine of Chapter 6. Its teaser figure is reproduced here as Figure 4.1, the implementation diagram as Figure 3.4, and the two stimulus melodies as Figure 4.6. The thesis surrounds that single contribution with four additional studies covering audio-to-MIDI transcription, stem separation, full-song generation, and an integrated five-professional evaluation. It also presents the production infrastructure that hosts all of them inside one DAW: a mixer with sends and buses, a parametric EQ, plugin hosting, and the musician-facing interaction layer of domain-specific hotkeys and contextual affordances.

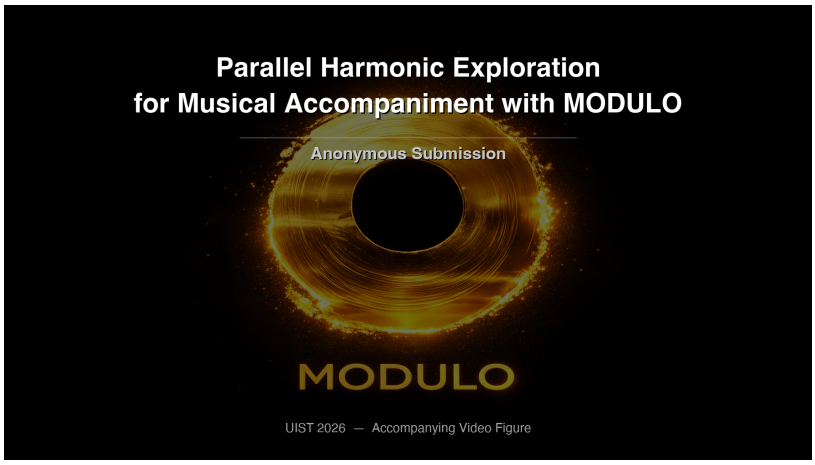


Figure 1: Title slide from the accompanying video figure submitted with the UIST 2026 paper. The conference submission corresponds to Chapters 4–6 of this thesis.

ON SOUND

We are living through the most significant transformation in music creation since the invention of multitrack recording. Large language models write lyrics, diffusion models generate instrumentals, and proprietary prompt-based systems produce radio-ready songs from a single sentence. And much of the music community is afraid.

The fear is understandable. When a model can produce in seconds what took a songwriter weeks, it is natural to ask whether the craft itself is being devalued, whether the years spent learning voicings, training one's ear, and developing a personal sound still matter. The backlash against AI in creative spaces is loud, emotional, and in many ways justified: the first wave of generative tools did treat music as content to be optimized rather than art to be shaped.

But refusing to engage with these systems is not an option. It would be the equivalent of refusing to use the internet twenty years ago: a principled stance that, in practice, simply cedes the future to those less concerned with getting it right. The question is not whether AI will reshape music creation; it already has. The question is whether the tools we build will preserve the things that make music worth creating in the first place: intention, interpretation, and the irreplaceable judgment of a trained ear.

My foundation in music was not built in conservatories. It began on a bedroom floor, with a laptop, a MIDI keyboard, and an obsession with getting the sound right. Before I studied signal processing, I was shaping soundscapes in Logic Pro, obsessing over balance and feel, layering and looping without the vocabulary to explain what I was doing. I have perfect pitch and a form of synesthesia that maps intervals to colors: blue for perfect fourths, maroon for minor sixths, golden dusk for a B-flat major seventh. When I improvise, my mind goes nearly blank; the music comes from somewhere I cannot fully articulate, and I suspect most musicians would say the same.

That inarticulate space, the gap between knowing what sounds right and being able to specify it formally, is precisely where generative AI can help or harm. A system that bypasses that space, that generates a finished product without the musician's involvement in shaping it, has not assisted creativity; it has replaced it. But a system that offers structured material for the musician to evaluate, compare, reshape, and commit to on their own terms has done something genuinely new. It has made the creative process faster without making it shallower.

Years later, those bedroom experiments evolved into formal training. Under Professor Nam, I learned the anatomy of professional postproduction in a Grammy-winning studio. At Igloo Music in Los Angeles, where I was the first intern hired without a Berklee degree on the strength of my engineering work alone, I sat in sessions with producers and engineers behind some of the most recognized recordings of the last decade. I became my year's only student to complete two simultaneous honors theses: one in engineering sciences, building the first discrete-token system for multi-effect music restoration, and one in music, producing a fully mastered album spanning five instruments and four genres. Those two theses were designed as two halves of the same question: how far can we push the machine, and how far can we push the musician, and where do they meet?

This thesis is where they meet. MODULO is my answer to the question of how generative AI enters the creative workflow without consuming it. It is built on the conviction that the interface between musician and model is not a convenience layer; it is the site where authorship is either preserved or lost. Every design decision in this system, from the chord workshop to the bus routing to the keyboard shortcuts, was made by a musician who has spent a lifetime on both sides of that interface.

The world does not need another system that generates music. It needs systems that help musicians create music that is unmistakably, irreplaceably theirs.

Contents

I Foundations 1

1 Introduction and Problem Space 2
- 1.1 The Chord and Harmony Problem . . . 3
  - 1.1.1 The Combinatorial Search Space . . . 3
  - 1.1.2 Behavioral Consequences of Intractability . . . 4
  - 1.1.3 Framing as a Co-Creative Search Problem . . . 5
- 1.2 One-Shot Limitations . . . 5
  - 1.2.1 The Generation–Composition Distinction . . . 6
  - 1.2.2 Empirical Evidence of the Limitation . . . 6
  - 1.2.3 The Prompt–Regenerate Loop . . . 7
- 1.3 GAW Landscape . . . 8
  - 1.3.1 Existing Systems . . . 9
  - 1.3.2 The Integration Gap . . . 11
- 1.4 Research Questions . . . 11
- 1.5 Thesis Contributions . . . 13
- 1.6 Thesis Structure . . . 15

2 Models and AI Systems 17
- 2.1 MusicGen . . . 18
  - 2.1.1 Audio Tokenization via a Neural Codec . . . 18
  - 2.1.2 Codebook Interleaving . . . 19
  - 2.1.3 Classifier-Free Guidance . . . 19
  - 2.1.4 Melody Conditioning and What It Does Not Provide . . . 20
  - 2.1.5 Takeaway . . . 21
- 2.2 Suno . . . 21
- 2.3 Udio . . . 23
- 2.4 ElevenLabs . . . 24
- 2.5 ReaLChords . . . 25
  - 2.5.1 Problem Formulation . . . 25
  - 2.5.2 Transformer Architecture . . . 26
  - 2.5.3 Pre-Training and GAPT Fine-Tuning . . . 26
  - 2.5.4 Online Accompaniment: Lookahead and Commitahead . . . 27
  - 2.5.5 Relevance . . . 27
- 2.6 Basic Pitch . . . 28
  - 2.6.1 Input Representation: Harmonic Constant-Q Transform . . . 28
  - 2.6.2 Multi-Output CNN Architecture . . . 29
- 2.7 HTDemucs . . . 30

2.7.1 Dual-Domain Encoder-Decoder . . . 31
2.7.2 Cross-Domain Attention . . . 31
2.8 Summary . . . 32

3 DAW Architecture **34**
3.1 DAW Concepts . . . 35
3.1.1 The Signal Processing Graph . . . 35
3.1.2 The Project Data Model . . . 36
3.2 The Tracktion Engine . . . 36
3.2.1 Architecture Overview . . . 36
3.2.2 Real-Time Audio Processing . . . 37
3.2.3 Plugin Hosting . . . 38
3.2.4 MIDI Subsystem . . . 38
3.2.5 Selection Rationale . . . 39
3.3 The JUCE Application Framework . . . 39
3.3.1 Core Capabilities . . . 39
3.4 Alternative: Ardour . . . 40
3.5 Commercial DAW Limitations . . . 41
3.6 The GAW Architecture Pattern . . . 43
3.6.1 Formal Definition . . . 43
3.6.2 Service-Oriented Generation . . . 44
3.6.3 Tradeoffs: Desktop-Native vs. Web-Based . . . 45
3.6.4 Implications for the Remaining Chapters . . . 45

II Harmonic Co-Creation **48**

4 Chord Generation System **49**
4.1 Architecture . . . 50
4.2 The Autoregressive Chord Model . . . 52
4.2.1 Transformer Architecture . . . 52
4.2.2 Reinforcement Learning Fine-Tuning . . . 53
4.3 Temperature Control . . . 54
4.4 Preprocessing Pipeline . . . 56
4.5 Playback Style Rendering . . . 57
4.6 Selective Commitment . . . 58
4.7 Auditioning Workflow . . . 59
4.8 Pipeline Diagram . . . 60
4.9 Evaluation: User Study Results . . . 60
4.9.1 Quantitative Results . . . 60
4.9.2 Subjective Ratings . . . 63
4.9.3 Qualitative Themes . . . 64
4.9.4 Per-Participant Case Studies . . . 66
4.9.5 Where the Comparison Was Structurally Favorable to MODULO . . . 66

5 The Chord Workshop **68**
5.1 Design Motivation . . . 68
5.2 Chord Cell Representation . . . 69
5.3 The Chord Parameter Space . . . 70
5.4 Numeral Display . . . 73
5.5 Out-of-Key Detection . . . 73
5.6 Preview Note Editing . . . 75
5.7 Interaction Design Principles . . . 77

6 Rule-Based Harmony Engine **78**
6.1 Voice Leading Motivation . . . 78
6.2 Scale and Interval Arithmetic . . . 79
6.3 The Five Interval Strategies . . . 80
6.4 The Adaptive Scoring Function . . . 82
6.4.1 Interval Consonance Term . . . 83
6.4.2 Smoothness Penalty . . . 83
6.4.3 Contrary Motion Reward . . . 84
6.4.4 Weight Calibration . . . 84
6.5 Consonance Hierarchy . . . 84
6.6 Parallel Motion Constraints . . . 85
6.7 Algorithm Complexity . . . 86
6.8 Harmony Study . . . 88
6.8.1 Experimental Design . . . 88
6.8.2 Task Flow . . . 89
6.8.3 Dependent Variables . . . 90
6.8.4 Statistical Analysis . . . 94
6.8.5 Hypotheses . . . 94
6.8.6 Power Analysis . . . 95
6.8.7 Results . . . 95
6.8.8 Where the Comparison Was Structurally Favorable to MODULO . 104

III Transcription & Stems **106**

7 Audio-to-MIDI Transcription **107**
7.1 Polyphonic Transcription . . . 107
7.2 AMT Landscape . . . 108
7.2.1 Onset and Frames . . . 108
7.2.2 CREPE: Monophonic Pitch Estimation . . . 109
7.2.3 MT3: Multi-Track Music Transcription . . . 109
7.2.4 Hybrid Time-Frequency Source Separation . . . 110
7.3 Harmonic CQT . . . 110
7.4 Multi-Pitch Neural Architecture . . . 111
7.5 Service Architecture and Caching . . . 112
7.6 Temporal Quantization Pipeline . . . 114

7.7 Note Merging Algorithm . . . 115
7.8 Chord Inference . . . 115
7.9 Transcription Pipeline Summary . . . 116
7.10 A2M Evaluation Study . . . 117
7.10.1 Participants . . . 117
7.10.2 Audio Stimuli . . . 118
7.10.3 Experimental Conditions . . . 118
7.10.4 Counterbalancing and Clip Assignment . . . 120
7.10.5 Guided Study Environment . . . 121
7.10.6 Task Flow . . . 122
7.10.7 Objective Accuracy Metrics . . . 123
7.10.8 Efficiency Metrics . . . 124
7.10.9 Self-Report Instruments . . . 125
7.10.10 Hypotheses . . . 126
7.10.11 Statistical Analysis Plan . . . 127
7.10.12 Sample Size Justification . . . 127
7.10.13 Results . . . 127

8 Stem Separation 138
8.1 Motivation . . . 138
8.2 System . . . 139
8.2.1 End-to-End Flow . . . 139
8.2.2 Engineering Details . . . 140
8.2.3 Why ElevenLabs, Not a Local Model? . . . 141
8.3 User Study . . . 142
8.3.1 Research Questions . . . 142
8.3.2 Design and Participants . . . 142
8.3.3 Stimuli . . . 144
8.3.4 Task Lifecycle . . . 144
8.3.5 Dependent Variables . . . 145
8.3.6 Hypotheses . . . 146
8.3.7 Statistical Plan . . . 147
8.4 Results . . . 147
8.4.1 Workflow: Time and Actions . . . 147
8.4.2 Audio Quality Against Reference Stems . . . 148
8.4.3 NASA-TLX and UES-SF . . . 150
8.4.4 Paper-Specific Likert Items . . . 151
8.4.5 Forced-Choice Final Preference . . . 152
8.4.6 Hypotheses . . . 153
8.4.7 Qualitative Themes . . . 153
8.5 Discussion . . . 155
8.5.1 Metric Asymmetry . . . 155
8.5.2 Workflow Not Model . . . 156
8.5.3 When External Tools Still Win . . . 156
8.5.4 Where the Comparison Was Structurally Favorable to MODULO . . . 157

8.5.5 Limitations . . . 157
8.6 Summary . . . 158

IV Generative Audio Features 159

9 Sound Effects Generation 160
9.1 SFX Sourcing Problem . . . 160
9.2 Text-Conditioned Audio Generation . . . 162
9.3 Generation Pipeline in MODULO . . . 163
9.3.1 Dialog Design . . . 164
9.3.2 Example Prompts . . . 166
9.4 Sourcing vs. Generation: A Workflow-Level Account . . . 167
9.5 Why ElevenLabs, Not a Local Model? . . . 168

10 Music Generation and Planning 170
10.1 Generation Problem . . . 170
10.2 Two-Phase Architecture . . . 171
10.2.1 Direct Generation . . . 171
10.2.2 Composition Plan Generation . . . 172
10.3 Service Interface and Reliability . . . 173
10.4 Prompt Design . . . 174
10.5 Section Layout Algorithm . . . 175
10.6 Stem Separation . . . 176
10.7 Service Interaction Sequence . . . 177
10.8 BPM and Key Accuracy . . . 179
10.9 Composition Planning . . . 179
10.9.1 Plan Representation . . . 180
10.9.2 Composition Plan Dialog . . . 180
10.9.3 Auto-Fill from Prompt . . . 181
10.9.4 Section-Boundary Audio Splitting . . . 181
10.10 Platform Comparison . . . 182
10.11 Composition Study . . . 184
10.11.1 Experimental Design . . . 184
10.11.2 Composition Briefs . . . 185
10.11.3 Conditions . . . 186
10.11.4 Dependent Variables . . . 187
10.11.5 Hypotheses . . . 189
10.11.6 Statistical Analysis Plan . . . 189
10.11.7 Results . . . 190
10.11.8 Discussion . . . 199

V Production Infrastructure 203

11 Mixer and Audio Engine 204
- 11.1 Audio Engine Architecture . . . 204
- 11.2 Signal Flow Model . . . 205
- 11.3 Parametric Equalization . . . 206
- 11.4 Plugin Hosting and Effects System . . . 208
- 11.5 Automation System . . . 209
- 11.6 Mixer Interface . . . 210

12 Musician-Facing UI 213
- 12.1 Design Philosophy . . . 213
- 12.2 Keyboard Shortcuts . . . 214
- 12.3 Hierarchical Track Organization . . . 216
- 12.4 Bus Creation and Routing . . . 217
- 12.5 Auto Labeling . . . 218
- 12.6 Idea Markers . . . 219
- 12.7 Icons and Affordances . . . 219
- 12.8 Piano Roll Editor . . . 220
- 12.9 Hotkey and Tool-Icon Inventory . . . 221
- 12.10 Design Rationale in Context . . . 224

VI Evaluation and Synthesis 226

13 Integrated System Evaluation 227
- 13.1 Why Five Professional Musicians . . . 228
- 13.2 Workflow Archetypes and Scenario Assignment . . . 229
- 13.3 Session Protocol . . . 231
- 13.4 Click-Level Observation Instrument . . . 233
- 13.5 Analysis Plan . . . 235
- 13.6 Competitor Positioning . . . 236
- 13.7 Scenarios Revisited . . . 238
- 13.8 Other Potential Composition Use Cases . . . 240
- 13.9 Results . . . 240
  - 13.9.1 Session-Level Behavioral Metrics . . . 241
  - 13.9.2 Feature Utilization Matrix . . . 242
  - 13.9.3 Per-Participant Session Narratives . . . 243
  - 13.9.4 Survey Outputs . . . 248
  - 13.9.5 Critical-Incident Catalog . . . 249
  - 13.9.6 Thematic Synthesis of Post-Session Interviews . . . 251
  - 13.9.7 Integration Verdict . . . 253

14 Cross-Study Synthesis **254**
14.1 Common Reporting Protocol . . . 255
14.2 Cross-Study Summary Tables . . . 256
14.3 Three Headline Gains Across the Studies . . . 258
14.4 Cross-Cutting Patterns . . . 258
14.5 Effect-Size Landscape . . . 262
14.6 Joint Evidence . . . 263
14.7 Closing Verdict . . . 264

15 Limitations **267**
15.1 External Model Dependency . . . 267
15.2 Desktop Architecture . . . 268
15.3 Evaluation Scope . . . 269
15.4 Think-Aloud Reactivity . . . 271
15.5 Musical Domain Coverage . . . 271
15.5.1 Scale and Mode Support . . . 271
15.5.2 Symbolic Domain Constraint . . . 272
15.5.3 Harmony Engine Determinism . . . 272
15.5.4 Evaluation Timing . . . 273
15.6 Baseline Strength and Ecological Validity of Controls . . . 273
15.7 Coupling of Interface and Modeling Decisions . . . 275
15.8 What MODULO Does Not Solve . . . 276
15.9 Interpretive Guidance: Surprising and Unsurprising Findings . . . 278
15.10 Where MODULO Could Not Reasonably Lose . . . 279

16 Future Work **282**
16.1 Audio Selective Commitment . . . 282
16.2 Proprietary Harmony Model . . . 283
16.3 Extended Scale and Mode Support . . . 284
16.4 Web Deployment . . . 285
16.5 Longitudinal Studies . . . 286
16.6 Multi-Instrument Workflows . . . 286
16.7 Collaborative Composition . . . 287
16.8 Production Automation . . . 288
16.9 Agentic Workflows and Model Context Protocols . . . 289

17 Coda **292**

A List of Acronyms **294**

B Glossary of Terms **297**

C Study Materials **310**
C.1 Participant Pool and Recruitment . . . 310
C.2 Session Logistics . . . 311
C.3 Intake Questionnaire . . . 313

C.4 Shared Survey Instruments 314
C.4.1 NASA-TLX 314
C.4.2 UES-SF 315
C.4.3 SUS 316
C.5 Per-Study Protocol Notes 317
C.5.1 Chord generation 318
C.5.2 Harmony generation 318
C.5.3 Audio-to-MIDI transcription 318
C.5.4 Stem separation 319
C.5.5 Composition pipeline 319
C.5.6 Integrated evaluation 319
C.6 Custom Likert Batteries 320
C.7 Chord Study Stimulus Melodies 322

Bibliography **323**

List of Tables

1.1 Table of GAW capability comparison across existing systems 11

2.1 Table of models employed by or compared against MODULO 33

4.1 Table of quantitative results from the chord generation study 64
4.2 Table of selected subjective rating results 65

6.1 Table of interval consonance scores 85
6.2 Table of workflow and musical quality metrics 96
6.3 Table of NASA-TLX subscale scores 98
6.4 Table of UES-SF and custom Likert item scores 101

7.1 Table of grid and snap-threshold parameters per quantization mode 114
7.2 Table of eight audio stimuli used in the transcription study 118
7.3 Table of the four experimental conditions in the A2M study 119
7.4 Table of Latin-square condition ordering for four groups 120
7.5 Table of A2M workflow and efficiency metrics by condition 128
7.6 Table of objective note-level transcription accuracy metrics 129
7.7 Table of NASA-TLX subscale scores for the A2M study 130
7.8 Table of UES-SF subscales and custom Likert items for the A2M study . . 131
7.9 Table of omnibus RM-ANOVA and pairwise effect sizes for A2M 132
7.10 Table of final three-way comparison counts for the A2M study 133

8.1 Table of objective audio-quality metrics by separation tool 149
8.2 Table of paper-specific Likert items by condition 151

10.1 Table of workflow-efficiency metrics across MODULO, Udio, and Suno . . 190
10.2 Table of NASA-TLX subscales and composite across conditions 193
10.3 Table of UES-SF subscales and composite across conditions 194
10.4 Table of prompt-to-output fidelity and workflow Likert items 196
10.5 Table of composition-plan interface Likert ratings 196
10.6 Table of final-comparison forced-choice counts 197

13.1 Table of feature coverage across the five assigned workflows. 230
13.2 Table of feature comparison across generative and traditional music production tools, April 2026. 237
13.3 Table of per-session behavioral metrics across the five participants. 241
13.4 Table of feature utilization across the five participants. 243
13.5 Table of survey outcomes per participant. 249

14.1 Table of headline outcomes per study. 256
14.2 Table of cross-study contrasts. 266

A.1 List of acronyms and initialisms used in this thesis. 294

B.1 Glossary of musical and technical terms used in this thesis. 297

C.1 Table of modified NASA-TLX subscales and anchor wording. 314

List of Figures

1 Title slide from the accompanying video figure submitted with the UIST 2026 paper. The conference submission corresponds to Chapters 4–6 of this thesis. . . . vii

1.1 Screenshot of the Suno free-tier one-shot generator interface . . . 7
1.2 Screenshot of Suno Studio web-based GAW interface . . . 9
1.3 TikZ diagram of the thesis landscape placing MODULO among DAWs and generators . . . 12

2.1 Diagram of codebook interleaving patterns in MusicGen . . . 20
2.2 Architecture diagram of ReaLChords RL training pipeline . . . 28
2.3 Architecture diagram of the Basic Pitch transcription CNN . . . 30
2.4 Architecture diagram of HTDemucs hybrid time-frequency separation . . . 32

3.1 Screenshot of the Ardour DAW interface . . . 41
3.2 Screenshot of Logic Pro arrangement view . . . 42
3.3 TikZ diagram of MODULO's layered architecture . . . 46
3.4 Architecture diagram of MODULO implementation subsystems . . . 47

4.1 Screenshot of MODULO chord generation and harmony surface . . . 50
4.2 Screenshot of the Generate Chords dialog . . . 51
4.3 Screenshot of five temperature columns with playback styles . . . 55
4.4 Screenshot of the selective commitment panel . . . 59
4.5 TikZ diagram of the chord generation pipeline . . . 61
4.6 Score of melodies A and B used as study stimuli . . . 62
4.7 Bar graph of temporal redistribution of creative effort . . . 63
4.8 Bar graph of harmonic vocabulary breadth . . . 64
4.9 Bar graph of selected subjective Likert ratings . . . 65
4.10 Bar graph of NASA-TLX subscale ratings . . . 66

5.1 Screenshot of the Chord Workshop interface . . . 70
5.2 Screenshot of root and quality selectors . . . 72
5.3 Screenshot of absolute versus Roman numeral display . . . 74
5.4 Screenshot of out-of-key highlighting on a timeline track . . . 75
5.5 Screenshot of the favorited-chord grid . . . 76

6.1 TikZ diagram of adaptive-strategy scoring . . . 86
6.2 Screenshot of treatment-condition harmony workspace . . . 90
6.3 TikZ diagram of harmony-study task lifecycle . . . 91
6.4 Bar graph of per-melody harmony task times . . . 97
6.5 Bar graph of harmony musical quality metrics . . . 99
6.6 Bar graph of NASA-TLX subscales . . . 100

6.7 Bar graph of UES-SF and custom Likert items . . . 102
6.8 Score of melody with five harmony strategies . . . 105

7.1 TikZ diagram of the Basic Pitch multi-pitch transcription architecture . . . 113
7.2 TikZ diagram of the full audio-to-MIDI transcription pipeline . . . 117
7.4 Screenshot of the in-app Audio-to-MIDI dialog for condition C1 . . . 120
7.5 TikZ diagram of the per-participant A2M task flow across four conditions . 122
7.3 Screenshot of the Basic Pitch web demo running on a polyphonic clip . . . 135
7.6 Bar graph of A2M transcription accuracy across three conditions . . . 136
7.7 Bar graph of per-clip A2M task time across three conditions . . . 136
7.8 Bar graph of NASA-TLX subscale scores for the A2M study . . . 137

8.1 Screenshot of the Extract Stems mode-selector dialog . . . 143
8.2 Bar graph of stem separation workflow time by condition . . . 148
8.3 Spectrograms of the vocal stem for Track 1 across two backends . . . 150
8.4 Mel-spectrograms of the drum stem for Track 1 . . . 150
8.5 Chromagrams of the vocal stem for Track 2 in voc/inst mode . . . 151
8.6 Bar graph of NASA-TLX six-subscale means for the stems study . . . 152
8.7 Bar graph of forced-choice final preferences across eight dimensions . . . 153

9.1 Screenshot of the Generate SFX dialog and Prompt Tips sidebar . . . 166

10.1 Sequence diagram of two-phase generation with stem separation . . . 178
10.2 TikZ diagram of composition plan mapped onto the timeline . . . 182
10.3 Screenshots of Suno and Udio free-tier prompt interfaces . . . 183
10.4 Bar graph of per-brief composition time across conditions . . . 191
10.5 Bar graph of browser round-trip time per brief . . . 192
10.6 Bar graph of NASA-TLX subscales and composite across conditions . . . 194
10.7 Bar graph of UES-SF subscales and composite across conditions . . . 195
10.8 Bar graph of paper-specific Likert items across conditions . . . 197
10.9 Bar graph of final-comparison forced-choice preferences . . . 198
10.10 Screenshot of MODULO composition plan dialog after auto-generation . . . 202

11.1 TikZ diagram of mixer signal flow from tracks to master . . . 206
11.2 Screenshot of four-band parametric EQ designer . . . 207
11.3 Screenshots of FX chain management and plugin hosting . . . 209
11.4 Screenshot of three automation lanes on the timeline . . . 210
11.5 Screenshot of the complete mixer view . . . 211
11.6 Annotated screenshot of a single channel strip . . . 212
11.7 TikZ diagram of bus routing topology . . . 212
11.8 Screenshots of Logic Pro and Ableton Live mixers for comparison . . . 212

12.1 Cheat sheet of MODULO keyboard shortcuts on a Mac keyboard . . . 215
12.2 Screenshot of track-stack hierarchy on the timeline . . . 217
12.3 Screenshot of auto-labeled tracks and clips on the timeline . . . 218
12.4 Screenshot of the complete MODULO transport bar . . . 219

12.5 Screenshot of piano-roll editor with chord stack in context 220

14.1 Coverage heatmap of study dimensions across the six studies. 257
14.2 Bar graph of wall-clock time reductions across the five component studies. . 259
14.3 Bar graph of NASA-TLX workload reductions across four studies. 260
14.4 Bar graph of forced-choice preference rates across three studies. 261
14.5 Bar graph of cross-study effect-size landscape across four studies. 263

Part I: Foundations

CHAPTER 1

Introduction and Problem Space

Music composition is, at its mathematical core, a search problem. A composer navigating harmonic space must select, at every rhythmic position, a chord drawn from a combinatorially vast vocabulary of pitch-class subsets, each admitting multiple voicings and inversions, each interacting with surrounding harmonic context in ways governed by centuries of tonal convention [4, 78] and decades of genre-specific practice. Modern generative systems [23] have made it possible to bypass this search entirely by prompting a model and receiving a finished audio track, but in doing so they eliminate the deliberate, note-level decision-making that distinguishes composition from consumption. The present wave of commercial audio generators, including Suno, Udio, and ElevenLabs Music, descends from a single research inflection point: MusicGen [28], introduced in the summer of 2023, which established the recipe of transformer-based audio token modeling that now underpins every major generator. Chapter 2 traces that lineage in detail. This thesis investigates the space between full automation and manual labor: a Generative Audio Workstation (generative audio workstation) that embeds structured generative capabilities within a professional multi-track production environment, preserving the musician's role as author while dramatically reducing the cost of harmonic exploration.

This chapter formalizes the harmonic search problem (Section 1.1), identifies the limitations of existing one-shot generative systems (Section 1.2), surveys the emerging generative audio workstation landscape (Section 1.3), states the research questions that guide the work (Section 1.4), enumerates the thesis contributions (Section 1.5), and outlines the

structure of the remaining chapters (Section 1.6).

1.1 THE CHORD AND HARMONY PROBLEM

1.1.1 The Combinatorial Search Space

Western tonal music discretizes pitch into twelve pitch classes [4], $\mathcal{P} = \{0, 1, \ldots, 11\}$, conventionally labeled $\{\mathrm{C}, \mathrm{C}\sharp, \mathrm{D}, \ldots, \mathrm{B}\}$. A *chord* is an ordered pair (r, q) where $r \in \mathcal{P}$ is a root pitch class and $q \in \mathcal{Q}$ is a quality drawn from a vocabulary of recognized sonority types [103]. In common practice [78], the quality set includes at minimum major, minor, dominant seventh, major seventh, minor seventh, diminished, augmented, and suspended variants:

$$\mathcal{Q} = \{\text{maj}, \text{min}, \text{dom7}, \text{maj7}, \text{min7}, \text{dim}, \text{aug}, \text{sus2}, \text{sus4}, \ldots\}. \tag{1.1}$$

Let $|\mathcal{Q}| = Q$. Each chord may be realized in one of K inversions, comprising root position plus one inversion per non-root chord tone, yielding a voicing set of size

$$|\mathcal{V}| = |\mathcal{P}| \times Q \times K = 12 \cdot Q \cdot K. \tag{1.2}$$

For a conservative estimate with $Q = 8$ quality types and an average of $K = 3$ inversions per quality, the single-position vocabulary is $|\mathcal{V}| = 12 \times 8 \times 3 = 288$ distinct voicings.

A chord progression of length N, measured in beats, bars, or other temporal quanta, is a sequence $\mathbf{c} = (v_1, v_2, \ldots, v_N) \in \mathcal{V}^N$. The cardinality of the progression space is

$$|\mathcal{V}^N| = |\mathcal{V}|^N = (12 \cdot Q \cdot K)^N. \tag{1.3}$$

For an eight-bar phrase in common time with one chord per beat ($N = 32$), the search space exceeds $288^{32} \approx 10^{78}$, a number exceeding the estimated count of atoms in the observable universe.

Even restricting attention to diatonic harmony within a single key reduces the root set to seven and the quality palette to perhaps five common diatonic sonorities, but the space remains $35^{32} \approx 10^{49}$ for a 32-beat phrase. In practice, musicians handle this intractability by relying on a small set of memorized patterns.

1.1.2 Behavioral Consequences of Intractability

The sheer size of the harmonic search space has measurable consequences for compositional behavior. Corpus analyses of popular music reveal that a small number of progressions dominate: the diatonic cycle I–V–vi–IV and its rotations account for a disproportionate share of commercial songwriting [78]. This clustering is not merely a reflection of aesthetic preference; it is a rational response to search intractability. When the cost of exploring an alternative chord is high, requiring manual voicing construction, auditioning, and contextual evaluation, musicians converge on the lowest-cost choices they already know.

Formally, let $\mathcal{S} : \mathcal{V}^N \to \mathbb{R}$ be a subjective quality function that a musician implicitly evaluates when assessing a candidate progression. The musician's task is to find

$$\mathbf{c}^* = \arg\max_{\mathbf{c} \in \mathcal{V}^N} \mathcal{S}(\mathbf{c}), \tag{1.4}$$

subject to a time budget B. Because exact maximization is infeasible, the musician instead performs a local search from an initial guess $\mathbf{c}^{(0)}$ drawn from memorized patterns, visiting a neighborhood $\mathcal{N}(\mathbf{c}^{(0)}) \subset \mathcal{V}^N$ defined by single-chord substitutions:

$$\mathcal{N}(\mathbf{c}) = \left\{ \mathbf{c}' \in \mathcal{V}^N \;\middle|\; d_H(\mathbf{c}, \mathbf{c}') = 1 \right\}, \tag{1.5}$$

where d_H denotes Hamming distance. Each neighborhood visit requires the musician to (i) select a position, (ii) choose a replacement voicing, (iii) enter it via the input device, and (iv) audition the result in context. Let the per-visit cost be Δt. The total number of ex-

plorable alternatives within budget B is then $\lfloor B/\Delta t \rfloor$; for typical $\Delta t \approx 15$–30 seconds and a 30-minute session, this yields at most 60–120 alternatives, a vanishingly small fraction of $|\mathcal{N}(\mathbf{c})| = N \cdot (|\mathcal{V}| - 1)$.

1.1.3 Framing as a Co-Creative Search Problem

The central insight motivating this thesis is that generative models can dramatically increase the exploration rate without removing the musician from the evaluation loop [89, 39]. Rather than replacing $\mathcal{S}$ with a learned surrogate, the system augments the musician's search by proposing multiple candidate continuations in parallel, each drawn from a learned distribution:

$$\mathbf{c}^{(i)} \sim p_\theta(\mathbf{c} \mid \mathbf{m}, \tau_i), \qquad i = 1, \ldots, N_{\text{col}}, \tag{1.6}$$

where p_θ is the autoregressive chord model, $\mathbf{m}$ is the melody context, τ_i is the temperature parameter controlling the entropy of the i-th sampling stream, and N_{col} is the number of parallel columns. The musician then evaluates and selectively commits individual chords from potentially different columns, constructing a hybrid progression:

$$\mathbf{c}^* = \left(c_{\pi(1)}^{(\sigma(1))}, c_{\pi(2)}^{(\sigma(2))}, \ldots, c_{\pi(N)}^{(\sigma(N))} \right), \tag{1.7}$$

where $\sigma(t) \in \{1, \ldots, N_{\text{col}}\}$ selects the column and $\pi(t)$ indexes the temporal position. This formulation preserves the musician as the maximizer of $\mathcal{S}$ while replacing the bottleneck of manual voicing construction with rapid audition of model-generated candidates.

1.2 LIMITATIONS OF ONE-SHOT GENERATIVE SYSTEMS

The past three years have witnessed an extraordinary acceleration in generative music capabilities [18, 23]. Systems capable of producing full-length, multi-instrument audio from

text prompts or melodic seeds have moved from research prototypes to commercial products with millions of users. Yet the interaction paradigm shared by these systems exhibits a fundamental limitation: they are *generation* tools, not *composition* tools [77, 25].

1.2.1 The Generation–Composition Distinction

We define this distinction formally. Let $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$ be a generation function mapping an input specification $x \in \mathcal{X}$, e.g., a text prompt, to an output $y \in \mathcal{Y}$, e.g., an audio waveform. A *generation tool* provides access to $\mathcal{G}$ and allows the user to iterate on x but provides no mechanism for structured modification of y beyond regeneration. A *composition tool*, by contrast, exposes the internal structure of y and provides operations for inspecting, comparing, and editing constituent elements at multiple levels of musical abstraction: notes, chords, phrases, sections, and tracks.

Formally, let $y = (y^{(1)}, y^{(2)}, \ldots, y^{(S)})$ be a decomposition of the output into S structural components. A composition tool provides:

(i) **Inspection**: the ability to isolate and render any $y^{(s)}$ independently;

(ii) **Comparison**: the ability to juxtapose alternative realizations $y_a^{(s)}$ and $y_b^{(s)}$ at corresponding temporal positions;

(iii) **Selective editing**: the ability to replace $y^{(s)}$ while holding $y^{(-s)}$ fixed.

One-shot generators provide none of these capabilities. The output is monolithic: the user may accept, reject, or regenerate, but cannot reach inside to adjust individual harmonic, melodic, or timbral decisions.

1.2.2 Empirical Evidence of the Limitation

Recent empirical work corroborates this structural deficiency. A study of secondary music education contexts found that a timeline-based interface was essential for students to engage meaningfully with generated material: without one, interaction reduced to repeated

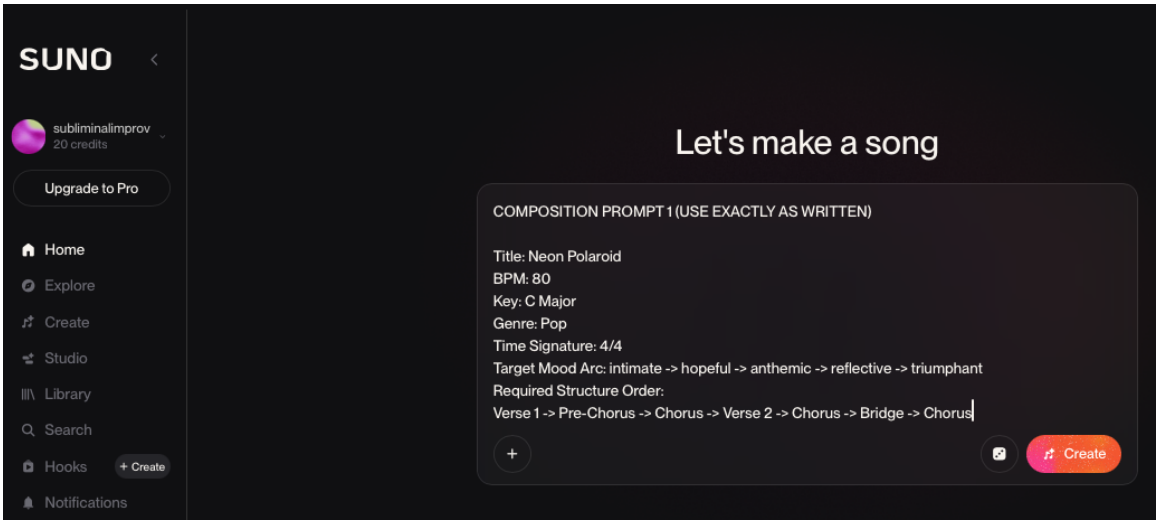


Figure 1.1: The prompt-and-receive paradigm of a representative one-shot generator, the Suno free tier [97]. The interface exposes text entry, style tags, and playback but no timeline, note editor, or chord inspector; structural revision requires returning to the prompt.

prompting until an acceptable result appeared by chance [24]. Large-scale analysis of prompt–output pairs has shown that users of text-conditioned generators develop elaborate prompting workarounds, specifying key signatures, tempo markings, instrument lists, and even chord progressions in natural language, to compensate for the absence of direct structural editing [22]. A co-design study with practicing musicians found that musicians value granular control over individual musical moments rather than holistic output acceptance, reporting that creative satisfaction correlates with the ability to shape material at the phrase and chord level rather than at the song level [63].

1.2.3 The Prompt–Regenerate Loop

The interaction pattern that emerges from one-shot systems can be modeled as a Markov chain over the prompt space $\mathcal{X}$. At each step k, the user submits prompt x_k, receives output $y_k = \mathcal{G}(x_k)$, evaluates $\mathcal{S}(y_k)$, and either terminates when $\mathcal{S}(y_k) \geq \theta_{\text{sat}}$ for some satisfaction threshold θ_{sat}, or modifies the prompt to obtain x_{k+1}. The expected number of iterations

before satisfaction is

$$\mathbb{E}[\text{iterations}] = \frac{1}{\Pr[\mathcal{S}(\mathcal{G}(x)) \geq \theta_{\text{sat}}]}, \tag{1.8}$$

which grows rapidly as the musician's quality threshold rises. Crucially, each failed iteration provides no partial credit: the musician cannot salvage the harmonically interesting bridge from an otherwise unsatisfactory generation [67, 90].

1.3 THE GENERATIVE AUDIO WORKSTATION LANDSCAPE

The limitations of both traditional DAWs, which lack generation, and one-shot generators, which lack editing, have catalyzed a new category of tools that we term *Generative Audio Workstations* (GAWs) [77]. We define a GAW formally as follows.

Definition 1 (Generative Audio Workstation)**.** *A Generative Audio Workstation is a software system $\mathcal{W} = (\mathcal{D}, \mathcal{G}, \mathcal{I})$ where:*

(i) $\mathcal{D}$ is a digital audio workstation core providing multi-track recording, editing, mixing, and playback;

(ii) $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_K\}$ is a set of generative services producing musical material at one or more levels of abstraction (audio, symbolic, structural);

(iii) $\mathcal{I}$ is a co-creation interface layer that mediates interaction between the musician and $\mathcal{G}$ within the context of $\mathcal{D}$, providing at minimum inspection, comparison, and selective editing of generated material.

This definition excludes pure generators, which lack $\mathcal{D}$; traditional DAWs, which lack $\mathcal{G}$; and systems that provide generation without in-context co-creation interfaces, which lack $\mathcal{I}$. We now survey existing systems against this definition.

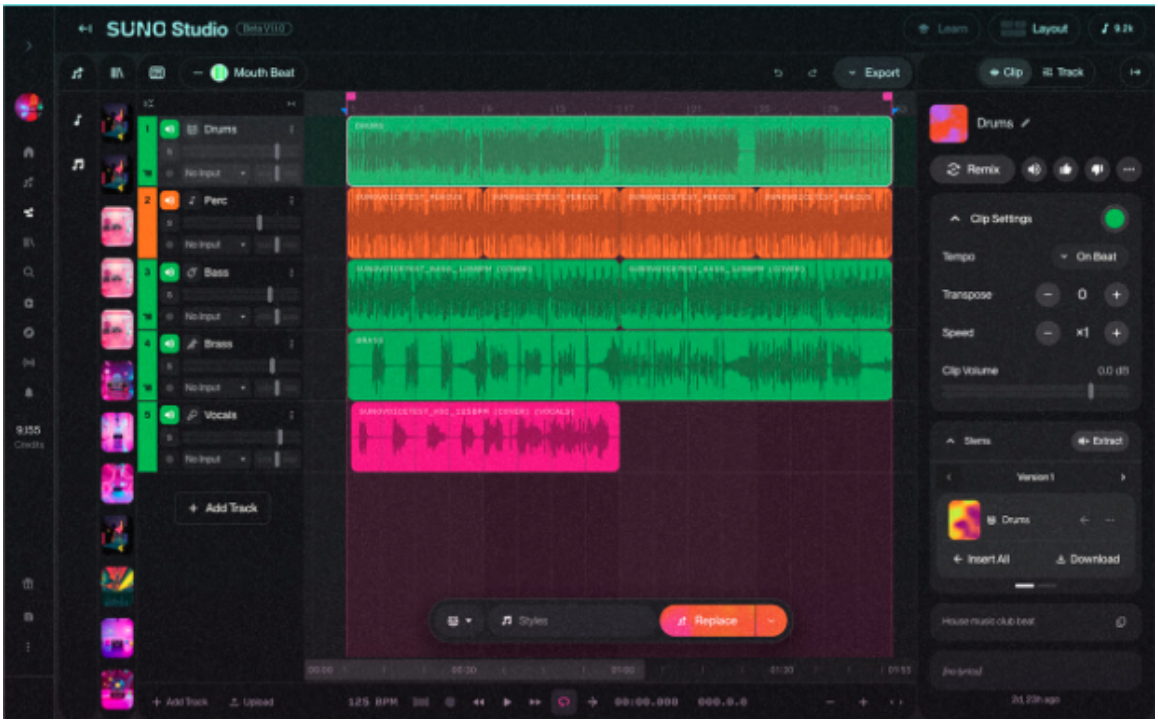


Figure 1.2: Suno Studio [98]: a web-based generative audio workstation with timeline arrangement, AI music generation, stem separation, audio-to-MIDI transcription, and basic mixing. Distinguishes from MODULO by its web-only deployment, absence of plug-in hosting, and lack of parallel chord exploration with selective commitment.

1.3.1 Existing Systems

Web-based generative workstations have rapidly matured into serious contenders for the GAW category. Suno Studio [98], launched in late 2025, extends the original Suno song generator into a full timeline-based production environment with AI music generation, stem separation, audio-to-MIDI transcription, and basic mixing capabilities. Mozart AI [71] provides a comparable feature set with a stronger emphasis on symbolic notation-level composition, also offering AI-assisted generation, transcription, and rudimentary mixing. Both platforms check the majority of boxes in a generative workstation capability matrix and represent the current leaders in the commercial GAW space. However, both remain web-only platforms with important limitations: neither provides a desktop-native application, advanced mixing infrastructure such as auxiliary bus routing, send/return configurations, and parametric equalization with per-band controls, plug-in hosting for VST3, Audio Unit,

or AAX, symbolic chord editing with inversion and voicing control, a rule-based harmony generation engine, parallel candidate exploration with temperature-controlled sampling, or selective commitment of individual chords from browsable option sets. In the notation of Definition 1, these systems provide strong $\mathcal{G}$ and partial $\mathcal{D}$ but incomplete $\mathcal{I}$, the co-creation interface layer that enables fine-grained musical decision-making.

AI-powered audio editors such as RipX DAW [48] focus on a complementary capability: decomposing existing audio into constituent stems via source separation models. This enables powerful audio-level editing such as isolating vocals, adjusting instrument levels, and repitching individual notes in the audio domain, but the representation remains fundamentally audio-centric. RipX provides a desktop application with some mixing capability and audio-level note manipulation, but there is no symbolic chord layer, no harmonic generation, and no mechanism for generating new content conditioned on existing material. It excels as a post-production tool for remixing and stem extraction rather than as a composition environment.

Additional web-based composition tools occupy a broader ecosystem. Udio [105] provides high-quality prompt-based audio generation with some post-generation editing such as extending and remixing, though it remains primarily a generation-first platform without full workstation capabilities [73]. BandLab [11] offers a free, web-based multitrack DAW with basic AI features including auto-mastering and sample suggestion, but its generative capabilities are limited to preset-driven assistance rather than structured co-creation. AIVA [3] focuses on AI-driven composition for film, game, and commercial scoring, generating MIDI-level compositions in specific styles, but operates as a standalone generator without integrated production tools.

Chord-specific generators such as boombox.io [16] and ChordSeqAI [26] provide focused tools for harmonic exploration, including chord progression generation, continuation suggestions, and harmonic relationship visualizations, but exist as standalone utilities disconnected from any production environment. The generated progressions must be manually

Table 1.1: Capability comparison of existing systems against the GAW definition. ✓ = full support; ∼ = partial or basic support; – = absent.

Capability	**Suno St.**	**Mozart**	**RipX**	**boombox**	**ChordSeq**	**MODULO**
AI music generation	✓	✓	–	–	–	✓
Audio-to-MIDI	✓	✓	–	–	–	✓
Stem separation	✓	∼	✓	–	–	✓
Symbolic chord editing	∼	✓	–	∼	∼	✓
Harmonic generation	–	∼	–	✓	✓	✓
Parallel exploration	–	–	–	–	–	✓
Selective commitment	–	–	–	–	–	✓
Mixing & mastering	∼	∼	∼	–	–	✓
Bus routing & sends	–	–	–	–	–	✓
Plugin hosting (VST/AU)	–	–	–	–	–	✓
Desktop-native	–	–	✓	–	–	✓
Musician hotkeys	–	–	–	–	–	✓

transcribed or exported, losing the in-context evaluation that is essential to compositional judgment.

1.3.2 The Integration Gap

Table 1.1 summarizes the capability matrix. Suno Studio and Mozart AI are formidable platforms that cover the majority of generative workstation features; the critical differentiators for MODULO are: (i) desktop-native execution with no browser dependency, (ii) symbolic chord editing with inversion and voicing control, (iii) a rule-based harmonic generation engine, (iv) parallel candidate exploration with temperature-controlled sampling, (v) selective commitment of individual chords from browsable option sets, (vi) advanced mixing infrastructure with auxiliary bus routing, and (vii) plug-in hosting for third-party VST3 and Audio Unit instruments and effects.

1.4 RESEARCH QUESTIONS

The analysis of the preceding sections motivates three research questions, each addressing a distinct facet of the generation–composition tension.

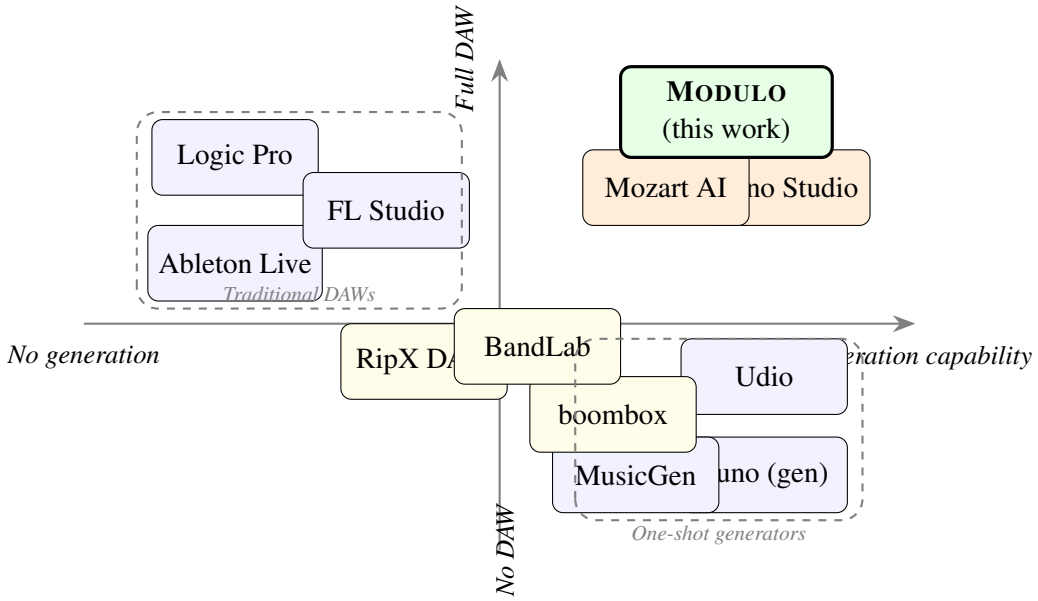


Figure 1.3: The thesis landscape. Traditional DAWs in the upper-left offer full production capability without generation. One-shot generators in the lower-right produce complete audio without structural editing. Suno Studio and Mozart AI in the upper-center-right are strong GAW competitors covering most generative and basic production features. MODULO in the upper-right differentiates through desktop-native execution, advanced mixing with bus routing, plug-in hosting, symbolic chord editing with harmonic generation, and musician-facing interaction design.

RQ1: Can structured harmonic co-creation within a DAW accelerate accompaniment composition?

Specifically, when parallel model-generated chord candidates are presented on the timeline for in-context auditioning and selective commitment, does this reduce the time required for a musician to arrive at a satisfactory accompaniment, compared to manual chord entry in a conventional DAW workflow? We operationalize "satisfactory" as the musician's self-reported completion signal and measure time-to-first-satisfaction as the primary dependent variable.

RQ2: Does parallel exploration with temperature-controlled generation broaden harmonic vocabulary?

Beyond speed, when alternatives are sampled at varying temperatures to span the conservative-to-exploratory spectrum, does exposure to them lead musicians to em-

ploy a wider range of chord roots, qualities, and voicings than they would select independently? We operationalize "breadth" as the count of unique chord roots and unique chord qualities in the committed progression.

RQ3: Can a unified system integrating generation, transcription, and production maintain musician agency?

As the number of AI-assisted capabilities grows to include chord generation, audio-to-MIDI transcription, prompt-conditioned music generation, stem separation, sound effects synthesis, and composition planning, does the musician retain the sense of authorial control and creative ownership, or does the system's automation erode perceived agency? We operationalize "agency" through validated self-report instruments [43, 76, 20] and qualitative analysis of think-aloud protocols [37, 17].

These questions are deliberately ordered from most concrete to most integrative. RQ1 admits a crisp quantitative answer from a controlled within-subjects experiment [27]. RQ2 extends the evaluation to a second dimension of creative outcome. RQ3 addresses the holistic experience of working within a system that is simultaneously a generative engine and a production environment.

1.5 THESIS CONTRIBUTIONS

The scientific contribution of this thesis is the *joint design of model and interface* for music co-creation: each subsystem pairs a specific modeling strategy with a specific interaction paradigm, and the empirical claims attach to those pairings rather than to either component in isolation. The selective-commitment chord workshop is only useful because the multi-temperature transformer produces meaningfully different candidates at each τ; the multi-temperature transformer is only useful inside a DAW because the workshop lets a musician audition and commit per-chord. The same logic applies to every other subsystem. With this

framing in mind, the thesis makes seven joint design contributions, each corresponding to a distinct technical subsystem and its associated evaluation:

C1: Harmonic co-creation: parallel multi-temperature generation paired with selective-commitment chord workshop. The joint design of a reinforcement-learning-tuned chord generation transformer [111] that produces parallel temperature-varied candidates, a direct-manipulation chord editing interface [53] that exposes those candidates as browsable cells, and a rule-based voice-leading harmony engine [52] steered through a five-strategy panel. Neither the model alone nor the interface alone produces the observed effects; the contribution is the pairing, as detailed in Chapters 4–6.

C2: In-DAW audio-to-MIDI: neural transcription paired with one-button workflow. The pairing of a lightweight neural pitch-estimation model [14] with a single in-DAW button that bypasses external upload/download cycles. The model produces accuracy parity with web-tool baselines; the interface eliminates the round-trip; the workflow improvement requires both decisions together (Chapter 7).

C3: Prompt-conditioned generation with stem extraction: source-separation model paired with section-aware automatic timeline layout. A hybrid-transformer source-separation model [83] paired with a deterministic section-block layout algorithm that places generated audio on the timeline as individually editable section tracks. The model returns stems; the interface contribution is that the stems land in a structure a musician can immediately edit (Chapters 10 and 8).

C4: Sound effects generation paired with in-DAW import. A text-conditioned sound effect generator paired with an in-context preview-and-place workflow that lands generated SFX directly on the cursor's timeline position; see Chapter 9.

C5: Composition planning: free-form natural-language intent paired with

section-level structured blueprints. A composition planning interface that translates natural-language briefs into section-level parametric specifications, paired with a deterministic generator that consumes those specifications. Both the structured-prompt design and the section-block visual grammar contribute to the observed effect; see Chapter 10.

C6: Mixing and signal processing infrastructure as the production substrate. A complete mixing and signal processing infrastructure [118] with parametric equalization, effects chaining, auxiliary bus routing, and plug-in hosting; included not as a novel research contribution in itself but as the production substrate without which the generative subsystems above would not be usable for finished work; see Chapter 11.

C7: Musician-facing interaction layer paired with the underlying generative actions. A musician-facing interaction layer with domain-specific keyboard shortcuts, automatic track organization, and contextual visual affordances [5] that maps to the generative actions of subsystems C1–C5; see Chapter 12.

The user studies in Chapters 4, 6, 7, 8, and 10 bundle interface and model into the treatment condition by design, so they cannot fully disentangle interface effects from modeling effects. The empirical claims therefore belong to the joint pairings rather than to either component alone; Section 15.7 expands on this and identifies the factorial designs that would isolate the individual contributions.

1.6 THESIS STRUCTURE

The remainder of this thesis is organized into six parts.

Part I: Foundations (Chapters 1–3) establishes the problem context. Chapter 2 surveys the generative models and AI systems employed by and compared against MODULO.

Chapter 3 describes the DAW engine, application framework, and systems infrastructure underlying the implementation.

Part II: Harmonic Co-Creation (Chapters 4–6) presents the core harmonic subsystems. Chapter 4 details the parallel chord generation pipeline. Chapter 5 describes the direct-manipulation chord editing interface. Chapter 6 formalizes the rule-based voice-leading harmonization engine.

Part III: Transcription and Stem Separation (Chapters 7–8) covers the audio-to-symbolic bridge and the stem-level decomposition of existing audio.

Part IV: Generative Audio Features (Chapters 10–10) addresses prompt-conditioned music generation, sound effects synthesis, and composition planning.

Part V: Production Infrastructure (Chapters 11–12) describes the mixing engine and musician-facing interaction design.

Part VI: Evaluation and Synthesis (Chapters 13–16) presents the integrated evaluation framework, experimental results, limitations, and directions for future work.

CHAPTER 2

Models and AI Systems

Every generative capability that MODULO exposes to the musician is backed by a concrete model, and every one of those models inherits assumptions from a research wave that began in the summer of 2023. MusicGen [28] was the inflection point. Within eighteen months of its release, a cluster of commercial services [98, 105, 34] had converted its technical recipe, comprising discrete audio tokens, transformer language modelling over codebooks, and text and melody conditioning, into mainstream products; a parallel line of symbolic-level research [111, 84] had started exploring the same ideas at the chord and score level. This chapter surveys the models that shape MODULO's design and evaluation. Section 2.1 positions MusicGen as the catalyst of the current wave. Sections 2.2, 2.3, and 2.4 describe the three commercial descendants that this thesis compares against or integrates with directly. Section 2.5 formalizes ReaLChords, the symbolic chord model behind MODULO's harmonic engine. Sections 2.6 and 2.7 cover the transcription and separation models that bridge audio and symbolic representations. Section 2.8 synthesizes the common structural limitation these models share and motivates the interface-level contribution of the remaining chapters.

Three forward references orient the rest of the chapter. MusicGen, Suno, and Udio appear in Chapter 10 and Chapter 10 as prompt-based baselines against which MODULO's composition pipeline is compared. ElevenLabs is the cloud backend that MODULO integrates for full-track music generation in Chapter 10, stem separation in Chapter 8, and sound-effects synthesis in Chapter 9. ReaLChords and Basic Pitch run locally inside MOD-

ULO and are evaluated in Chapter 4 and Chapter 7 respectively.

2.1 MUSICGEN: THE CATALYST OF THE MODERN WAVE

Text-to-music generation did not begin with MusicGen. Jukebox [31] demonstrated multi-minute audio synthesis in 2020 using a hierarchy of vector-quantized variational autoencoders (VQ-VAEs) and sparse transformers, but its inference was slow, its outputs were artefact-laden, and the only available conditioning was genre, artist, and lyrics. MusicLM [2] followed in early 2023 with a three-stage hierarchical approach composed of SoundStream tokens, a semantic model, and an acoustic model, demonstrating high-fidelity text-conditioned generation for the first time, though it was never released as an interactive product. MusicGen [28], published at NeurIPS 2023 by the Meta AI research group, was the first system to combine high fidelity, controllable conditioning, practical inference speed, and open weights in a single release. Within a calendar year, the major commercial music generators available to musicians, including Suno, Udio, and ElevenLabs Music, had shipped products whose architectures closely followed the MusicGen recipe: a transformer language model over discrete audio tokens produced by a neural codec [30], conditioned on text embeddings and optionally a reference melody.

2.1.1 Audio Tokenization via a Neural Codec

MusicGen operates over discrete audio tokens produced by EnCodec [30], a convolutional autoencoder with residual vector quantization. The codec maps a waveform $\mathbf{x} \in \mathbb{R}^{T_s}$ (at sample rate f_s) to a matrix of codebook indices $\mathbf{Z} \in \{1, \ldots, N_b\}^{K_c \times T_c}$, where K_c is the number of residual vector-quantization codebooks and $T_c = T_s/(f_s \cdot r)$ is the number of temporal frames at downsampling ratio r. At 32 kHz with $K_c = 4$ and $r \approx 640$, a one-second audio clip is reduced to roughly 50 frames and 200 tokens, a representation

compact enough for a transformer to model autoregressively while still reconstructing to perceptually clean audio.

2.1.2 Codebook Interleaving

The central architectural innovation in MusicGen is the *codebook interleaving pattern*, which determines the order in which tokens across codebooks and time steps are presented to the autoregressive model. Three patterns are defined.

(i) **Flat**: all $K_c \cdot T_c$ tokens are serialized into a single sequence, yielding maximum expressiveness at the cost of $\mathcal{O}(K_c \cdot T_c)$ sequence length.

(ii) **Parallel**: at each time step, all K_c codebook tokens are predicted simultaneously, reducing sequence length to T_c but requiring the model to capture inter-codebook dependencies implicitly.

(iii) **Delay**: codebook k is delayed by $k-1$ time steps, introducing a diagonal pattern that balances sequence length ($T_c + K_c - 1$) with explicit inter-codebook conditioning.

The delay pattern is the default in public MusicGen checkpoints and was subsequently adopted with minor variations by MAGNeT [117], a non-autoregressive descendant from the same lab, and by every commercial descendant examined in this chapter.

2.1.3 Classifier-Free Guidance

To strengthen adherence to conditioning signals such as text descriptions or melody references, MusicGen employs classifier-free guidance (CFG) [82]. During inference, the model computes both a conditioned logit vector $\boldsymbol{\ell}_c$ and an unconditioned logit vector $\boldsymbol{\ell}_u$, combining them as

$$\boldsymbol{\ell}_{\text{cfg}} = (1 + \lambda)\, \boldsymbol{\ell}_c - \lambda\, \boldsymbol{\ell}_u, \tag{2.1}$$

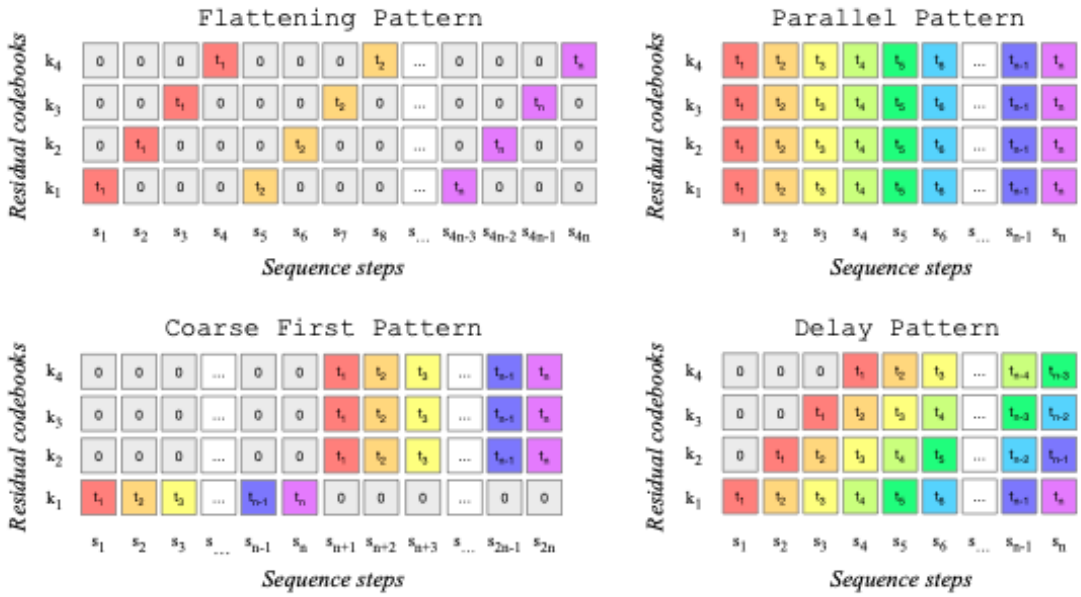


Figure 2.1: Codebook interleaving patterns from MusicGen [28, Fig. 1]. Each time step $t_1, t_2, \ldots, t_n$ is composed of four quantized values corresponding to codebooks $k_1, \ldots, k_4$. When doing autoregressive modelling, the four codebook streams can be flattened or interleaved in various ways, producing a new sequence with four parallel streams and steps $s_1, s_2, \ldots, s_m$; the total number of sequence steps S depends on the pattern and the original number of steps T, and a special token 0 marks empty positions. The delay pattern, used in the public checkpoints and adopted by every commercial descendant in this chapter, is the diagonal pattern at the right.

where $\lambda \geq 0$ is the guidance strength. In probability space, this corresponds to

$$p_{\text{cfg}}(\mathbf{z}_t \mid \mathbf{z}_{<t}, \mathbf{c}) \propto \frac{p(\mathbf{z}_t \mid \mathbf{z}_{<t}, \mathbf{c})^{1+\lambda}}{p(\mathbf{z}_t \mid \mathbf{z}_{<t})^{\lambda}}, \tag{2.2}$$

where $\mathbf{c}$ is the conditioning input. Setting $\lambda = 0$ recovers standard conditional sampling; increasing λ sharpens the distribution toward outputs that maximally distinguish the conditioned from the unconditioned model.

2.1.4 Melody Conditioning and What It Does Not Provide

For melody-conditioned generation, a reference melody is encoded via a chromagram $\mathbf{C}_{\text{mel}} \in \mathbb{R}^{12 \times T_c}$ and injected into the transformer via cross-attention layers. The chro-

magram represents the pitch-class energy distribution at each temporal frame and provides a pitch-class-level constraint without specifying exact voicings or timbres. The major commercial generators built in the eighteen months after MusicGen adopted some variant of this mechanism.

The constraint the chromagram cannot provide is the structural handle this thesis argues for: it anchors pitch content in aggregate but does not expose individual chords, notes, sections, or stems to the user for inspection or selective editing. The musician influences the output only by iterating on $(\text{prompt}, \text{melody})$ pairs and regenerating. Chapter 10 measures the cost of this loop in practice, and Chapter 10 compares it head-to-head against MODULO's section-aware workflow.

2.1.5 Takeaway

MusicGen mattered not because its samples were categorically better than MusicLM's but because it shipped as open weights with a simple, teachable recipe. The recipe, consisting of codec tokens, a transformer over codebooks, CFG, and chromagram conditioning, was reproducible in a small lab or startup. The wave of commercial products described in the following three sections was enabled by exactly this reproducibility. Those products also inherited, without modification, the limitation that motivates the rest of this thesis: one prompt in, one monolithic audio stream out.

2.2 SUNO: CONSUMER TEXT-TO-MUSIC AT SCALE

Suno [98] was among the first commercial products to operationalize the MusicGen recipe for consumer-scale music generation. Its public service generates full-length, multi-instrument audio with synchronized vocals and lyrics from a short text prompt. Data-driven analyses of its outputs [22] report that Suno generations are identifiable by their distribution of tempo, key, and structural repetition patterns, suggesting a training corpus biased

toward contemporary popular music.

The public product does not disclose its architecture, but independent analysis and the behaviour of its application programming interface (API) [22, 73] are consistent with a MusicGen-family design: text encoder, transformer over discrete audio tokens, neural codec decoder. The system supports two user-facing modes. *Prompt mode* accepts a natural-language description of style, mood, instrumentation, and lyrics. *Custom mode* additionally accepts a structured prompt that names sections such as verse, chorus, and bridge, and allows lyric-to-section alignment. Outputs are delivered as a stereo audio file with synchronized lyrics metadata.

Relevance to this thesis: Suno is one of the two external services benchmarked in the Chapter 10 composition pipeline evaluation. Participants who used Suno in that study described the same behavioural pattern documented in the secondary-education study of Choi and Chang [24]: when the first generation did not match the brief, users reverted to iterative prompt engineering because no mechanism existed to edit any single section. The integrated MODULO workflow measured against this behaviour produced a 47% reduction in per-brief composition time relative to Suno (MODULO $284 \pm 9\,\mathrm{s}$ vs. Suno $541 \pm 17\,\mathrm{s}$, $N = 10$); see Chapter 14.

Suno Studio [98], launched in late 2025, layers a timeline view and basic section editing on top of the generator and represents the closest commercial analogue to the generative audio workstation category defined in Chapter 1. Chapter 1 already discusses how Studio compares against MODULO along the GAW capability matrix; this chapter restricts its scope to the generator.

2.3 UDIO: AUDIO-FIRST REFINEMENT AND EXTENSION

Udio [105] emerged roughly six months after Suno as a direct competitor in the consumer text-to-music category. Its outputs are broadly comparable in fidelity and duration. The technical differentiator relevant to this thesis is Udio's *extension and remix* operations: the service allows a user to select a portion of a previously generated track and request a continuation, a variation, or a stylistic remix conditioned on that segment. This is an audio-domain analogue of the symbolic selective-commitment behaviour MODULO supports at the chord level.

That Udio adopted this capability before Suno Studio did is notable. It demonstrates that the commercial market for text-to-music generation began to pull toward finer-grained control within a year of the category's launch, consistent with the co-design findings of Król et al. [63] that musicians prefer per-segment interaction over whole-track acceptance. The ceiling of that pull is audio-level editing: even with extension and remix, Udio operates on waveform segments rather than on symbolic chord, note, or stem representations.

Relevance to this thesis: Udio is the second prompt-based baseline in the Chapter 10 evaluation. All ten composition-study participants preferred MODULO for section navigation and stem editability, even after acknowledging that Udio's extension operation was a useful coarse-grained control. The comparative analysis of Suno and Udio outputs by Casini et al. [22] is the primary external reference for quantitative characterisation of both services.

2.4 ELEVENLABS: THE CLOUD BACKEND INSIDE MODULO

ElevenLabs [33] began as a text-to-speech and voice-cloning company. Beginning in late 2024 and continuing through 2025, the company shipped three audio-generation endpoints that are directly integrated into MODULO:

(i) a music generation endpoint [34] producing full tracks from text prompts with optional section-level style descriptors;

(ii) a stem separation endpoint [36] that decomposes an input audio file into vocals, drums, bass, and other stems;

(iii) a sound-effects endpoint [35] for short text-conditioned audio clips of 1–22 seconds.

The internal architecture of each endpoint is proprietary and not publicly documented. Still, two properties suggest a MusicGen-family backbone running on managed graphics processing unit (GPU) infrastructure: the interface accepts a text prompt and returns audio, with structural metadata accepted as optional section descriptors; and the latency profile is on the order of tens of seconds per minute of audio produced.

Relevance to this thesis: ElevenLabs is not a baseline; it is the backend that MODULO talks to. The design choice to build on a cloud backend rather than a locally hosted MusicGen instance is treated in detail in Chapter 10. The short version of the argument: cloud inference eliminates the deployment-and-hardware-support problem on consumer laptops, the endpoint latency is roughly $2\times$ lower than a bundled local MusicGen would achieve on a typical user machine, and the backend company commits to forward compatibility in a way that an archived checkpoint does not. The cost is network dependence, which MODULO mitigates through progress feedback, asynchronous queuing, and the offline fallbacks described in Chapter 3.

The three endpoints together provide the audio-generation, stem-separation, and sound-effects capabilities that Chapters 10, 8, and 9 build on. In each study, the evaluation is not of the ElevenLabs endpoint in isolation but of the MODULO interface that wraps it.

2.5 REALCHORDS: AUTOREGRESSIVE CHORD GENERATION

In parallel with the audio-level wave initiated by MusicGen, a smaller body of research continued to target musical structure at the symbolic chord and score level [111, 84]. ReaLChords [111] is the most direct influence on MODULO's local chord generator. This section formalizes its architecture, training objective, and reinforcement-learning fine-tuning procedure.

2.5.1 Problem Formulation

Let $\mathbf{m} = (m_1, m_2, \ldots, m_T)$ denote a melody represented as a sequence of discrete tokens over T temporal frames, and let $\mathbf{c} = (c_1, c_2, \ldots, c_T)$ denote the corresponding chord sequence, where each c_t belongs to a finite vocabulary $\mathcal{V}_c$ of chord symbols augmented with onset and offset markers. The model estimates the conditional distribution

$$p_\theta(\mathbf{c} \mid \mathbf{m}) = \prod_{t=1}^{T} p_\theta(c_t \mid \mathbf{c}_{<t},\ \mathbf{m}_{\leq t+L}), \tag{2.3}$$

where θ denotes the model parameters and $L \geq 0$ is a *lookahead window* that allows the model to condition on future melody events up to L frames ahead. The lookahead parameter is critical for online accompaniment, where the model must anticipate melodic trajectories to produce harmonically coherent chords in real time.

2.5.2 Transformer Architecture

The model follows the standard decoder-only transformer design [108] with $\mathcal{L}$ layers of masked multi-head self-attention and position-wise feed-forward networks. At each position t, the input is formed by summing a token embedding $\mathbf{e}(c_t) \in \mathbb{R}^d$, a melody-context embedding $\mathbf{f}(\mathbf{m}_{\leq t+L}) \in \mathbb{R}^d$, and a sinusoidal positional encoding $\mathbf{p}_t \in \mathbb{R}^d$:

$$\mathbf{x}_t = \mathbf{e}(c_t) + \mathbf{f}(\mathbf{m}_{\leq t+L}) + \mathbf{p}_t. \tag{2.4}$$

The self-attention mechanism at layer ℓ computes

$$\text{Attn}^{(\ell)}(\mathbf{X}) = \text{softmax}\left(\frac{\mathbf{Q}^{(\ell)}(\mathbf{K}^{(\ell)})^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}^{(\ell)}, \tag{2.5}$$

where $\mathbf{Q}^{(\ell)}, \mathbf{K}^{(\ell)}, \mathbf{V}^{(\ell)}$ are the query, key, and value projections, d_k is the key dimension, and $\mathbf{M}$ is a causal mask ensuring that position t attends only to positions $\leq t$.

2.5.3 Pre-Training and GAPT Fine-Tuning

The model is first trained via maximum likelihood estimation on a corpus of melody and chord pairs [45]:

$$\mathcal{L}_{\text{MLE}}(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\log p_\theta\left(c_t^{(n)} \mid \mathbf{c}_{<t}^{(n)},\ \mathbf{m}^{(n)}\right). \tag{2.6}$$

The MLE-trained model is then fine-tuned via Generative Adversarial Post-Training (GAPT), a reinforcement learning from human feedback (RLHF) procedure [86] using two complementary reward models. A contrastive reward R_{con} is trained via a Bradley–Terry objective

$$\mathcal{L}_{\text{con}}(\phi) = -\mathbb{E}\left[\log\sigma\left(R_{\text{con}}(\mathbf{c}^+, \mathbf{m}; \phi) - R_{\text{con}}(\mathbf{c}^-, \mathbf{m}; \phi)\right)\right], \tag{2.7}$$

where $\mathbf{c}^+$ and $\mathbf{c}^-$ are preferred and dispreferred chord sequences. A discriminative reward

R_{disc} is a binary classifier trained to distinguish model outputs from human-composed sequences:

$$R_{\text{disc}}(\mathbf{c}, \mathbf{m}; \psi) = \log \frac{p_\psi(\text{human} \mid \mathbf{c}, \mathbf{m})}{p_\psi(\text{model} \mid \mathbf{c}, \mathbf{m})}. \tag{2.8}$$

The combined reward $R = \alpha R_{\text{con}} + \beta R_{\text{disc}}$ is maximized via Proximal Policy Optimization (PPO) with a Kullback–Leibler (KL) penalty anchoring the policy to the pre-trained base:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_{\mathbf{c}\sim p_\theta}[R(\mathbf{c}, \mathbf{m}) - \lambda_{\text{KL}}\, D_{\text{KL}}(p_\theta(\cdot \mid \mathbf{m}) \parallel p_{\theta_0}(\cdot \mid \mathbf{m}))]\,. \tag{2.9}$$

2.5.4 Online Accompaniment: Lookahead and Commitahead

For real-time accompaniment, the model operates under two temporal constraints. *Lookahead* L allows the model to observe up to L future melody frames before committing a chord at the current position. *Commitahead* A requires the model to commit chords up to A frames into the future at each decision point, amortizing inference cost across multiple frames. The effective conditional at decision point t becomes

$$p_\theta(c_{t:t+A} \mid \mathbf{c}_{<t},\ \mathbf{m}_{\leq t+L}), \tag{2.10}$$

requiring the model to produce a short burst of chord decisions from a partially observed melody context.

2.5.5 Relevance

ReaLChords is the local engine behind MODULO's chord generator, evaluated at $N = 25$ in Chapter 4. The lookahead and commitahead parameters are directly exposed to the interface so that the parallel-column generator in Chapter 4 can request alternative futures at different temperatures without blocking on full re-inference per column. The contrast with MusicGen is instructive: both models produce a single autoregressive stream, but ReaLChords emits discrete chord symbols that the MODULO interface can decompose, display

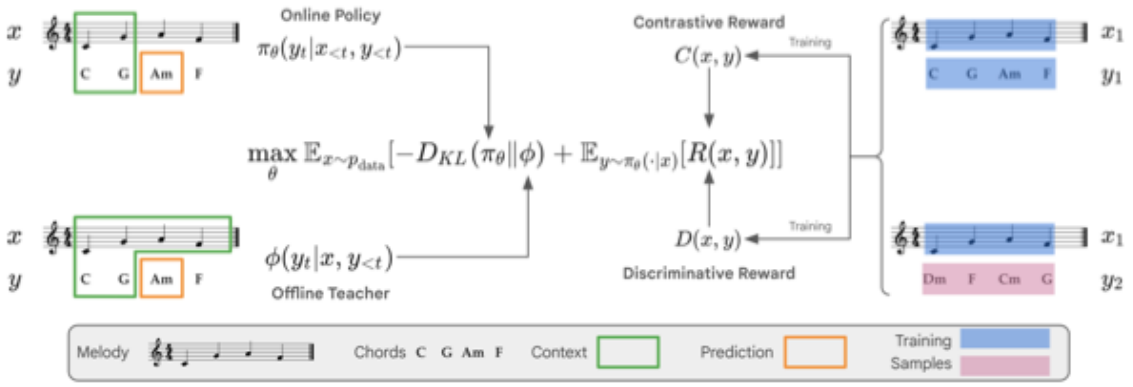


Figure 2.2: ReaLChords [111, Fig. 2] leverages RL fine-tuning to learn anticipation and adaptation for online melody-to-chord accompaniment. Initialized from a model π_θ pre-trained by MLE, the policy generates a complete chord response to a melody from the dataset, with each chord predicted given only previous melody and chords, as shown top left. In contrast, the offline model ϕ_ω, also trained by MLE, predicts each chord given the complete melody, as shown bottom left. A KL-divergence penalty distills the predictions of the offline model into the online model, improving its ability to anticipate the future. Right: the reward stems from an ensemble of multi-scale contrastive and discriminative models that evaluate the musical coherence between melody and chord. The final training objective in ReaLChords is a sum of the reward and the distillation loss, shown at center.

as browsable alternatives, and commit to the timeline one at a time. The symbolic level of abstraction is what enables the co-creative mode measured in Chapter 4 and Chapter 6.

2.6 BASIC PITCH: AUDIO-TO-MIDI TRANSCRIPTION

MODULO's audio-to-MIDI transcription capability is based on Basic Pitch [14], a lightweight convolutional model for polyphonic note transcription and multipitch estimation released by Spotify.

2.6.1 Input Representation: Harmonic Constant-Q Transform

The model operates on a harmonic constant-Q transform (HCQT) representation. The standard constant-Q transform computes spectral coefficients at frequencies spaced logarithmi-

cally:

$$f_k = f_{\min} \cdot 2^{k/B}, \qquad k = 0, 1, \ldots, K-1, \tag{2.11}$$

where $f_{\min}$ is the minimum frequency and B is the number of bins per octave. The Harmonic Constant-Q Transform (HCQT) extends this by computing the Constant-Q Transform (CQT) at multiple harmonic ratios $h \in \{0.5, 1, 2, 3, 4, 5\}$ and stacking the resulting spectrograms along a harmonic dimension:

$$\mathbf{S}_{\text{HCQT}}(h, k, t) = \text{CQT}(x; f_{\min} \cdot h, B)(k, t). \tag{2.12}$$

This multi-harmonic representation provides the model with explicit access to the harmonic series of each pitch, facilitating disambiguation of fundamentals from overtones.

2.6.2 Multi-Output CNN Architecture

The transcription model is a convolutional neural network that maps the HCQT input to three parallel output heads: a note posterior $\hat{y}_{\text{note}}(p, t) \in [0, 1]$ giving the probability that pitch p is active at frame t; an onset posterior $\hat{y}_{\text{onset}}(p, t)$ giving the probability that pitch p has an onset at frame t; and a contour posterior $\hat{y}_{\text{contour}}(p, t)$ providing a continuous pitch activity map that permits sub-semitone resolution for pitch-bend estimation. The three heads share a common encoder backbone g_ϕ and branch into task-specific decoders:

$$\begin{aligned} \hat{y}_{\text{note}} &= \sigma(h_{\text{note}}(g_\phi(\mathbf{S}_{\text{HCQT}}))) \,, \\ \hat{y}_{\text{onset}} &= \sigma(h_{\text{onset}}(g_\phi(\mathbf{S}_{\text{HCQT}}))) \,, \\ \hat{y}_{\text{contour}} &= \sigma(h_{\text{contour}}(g_\phi(\mathbf{S}_{\text{HCQT}}))) \,. \end{aligned} \tag{2.13}$$

MIDI events are extracted by thresholding the note and onset posteriors, applying a minimum-duration filter, and optionally using the contour output to estimate pitch-bend envelopes.

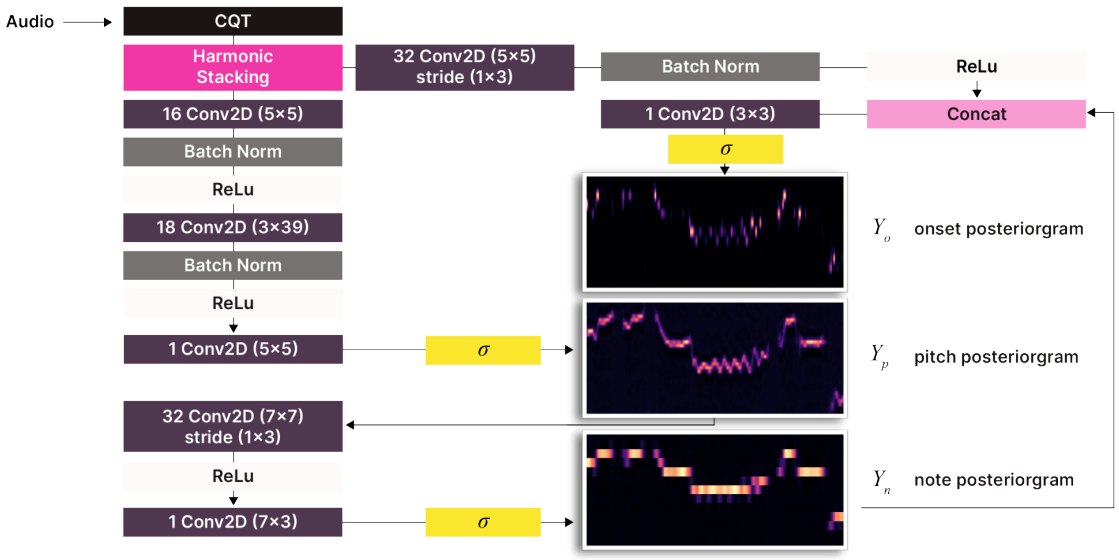


Figure 2.3: Basic Pitch architecture [14]. An HCQT feeds a shared convolutional encoder that branches into three sigmoid heads for note activity, onset, and continuous pitch contour.

Relevance to this thesis: Basic Pitch is the local backend behind the audio-to-MIDI pipeline evaluated at $N = 25$ in Chapter 7. The study compares three conditions: manual transcription in a conventional DAW, an external Basic Pitch web tool, and MODULO's in-app integration. The in-app condition achieves an 87% time reduction against manual and eliminates the external round-trip that adds a full 443 seconds per clip to the external-tool condition; see Chapter 7 and Chapter 14.

2.7 HTDEMUCS: HYBRID TIME-FREQUENCY SOURCE SEPARATION

Source separation decomposes a mixture signal into its constituent stems. The hybrid transformer architecture HTDemucs [83], building on the original Demucs convolutional design [29], achieves high separation quality by operating simultaneously in the time and frequency domains.

2.7.1 Dual-Domain Encoder-Decoder

The architecture comprises two parallel encoder-decoder pathways. The time-domain branch operates on the raw waveform $\mathbf{x} \in \mathbb{R}^{T_s}$ via a 1-D convolutional encoder with strided downsampling, producing latent representations $\mathbf{h}_t \in \mathbb{R}^{D \times T_t}$. The frequency-domain branch operates on the complex spectrogram $\mathbf{X} \in \mathbb{C}^{F \times T_f}$ computed via the short-time Fourier transform, using a 2-D convolutional encoder to produce latent representations $\mathbf{h}_f \in \mathbb{R}^{D \times F' \times T'_f}$.

2.7.2 Cross-Domain Attention

The two branches exchange information via cross-domain transformer layers. At the bottleneck, the time-domain latent and the reshaped frequency-domain latent are concatenated along the sequence dimension and processed by a transformer with bidirectional self-attention:

$$[\tilde{\mathbf{h}}_t;\ \tilde{\mathbf{h}}_f] = \text{Transformer}([\mathbf{h}_t;\ \text{reshape}(\mathbf{h}_f)]), \tag{2.14}$$

where $[\cdot\,;\,\cdot]$ denotes sequence concatenation. The enriched representations are then decoded by their respective branches, producing time-domain and frequency-domain source estimates that are summed to form the final separated stems. The model supports separation into vocals, drums, bass, and optionally guitar, piano, and other stems, each produced by a dedicated output head sharing the common encoder, attention, and decoder backbone.

Relevance to this thesis: HTDemucs is the canonical open-weights baseline and is one of two external tools compared against MODULO's in-app stem separation in Chapter 8. The MODULO condition in that chapter routes to an ElevenLabs stem-separation endpoint rather than a bundled local HTDemucs instance. The rationale and the trade-off are developed in Chapter 8, Section 8.4.2: time-domain fidelity metrics favour local HTDemucs, while perceptual metrics favour the cloud backend.

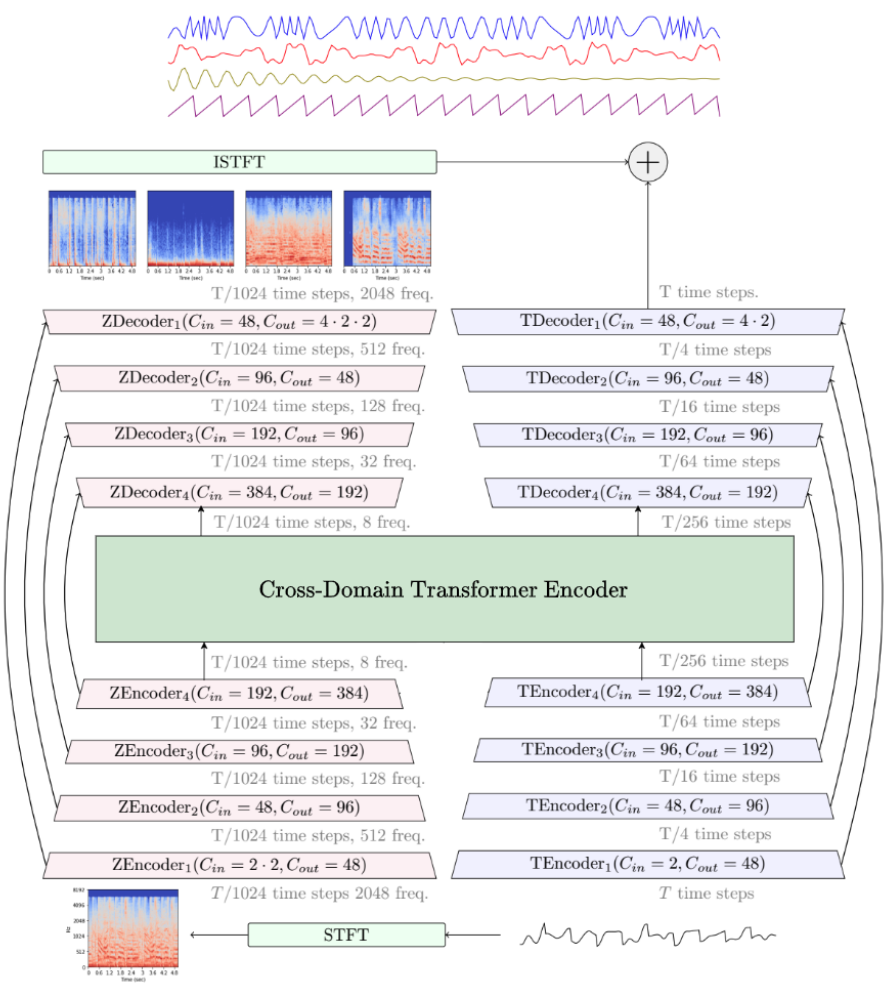


Figure 2.4: HTDemucs [83]. Parallel time- and frequency-domain encoder-decoder branches exchange information via cross-domain transformer attention at the bottleneck, producing per-stem estimates that are summed in their native domain.

2.8 SUMMARY AND THE COMMON LIMITATION

The nine models surveyed in this chapter span two output modalities, audio and symbolic; three deployment contexts, local, cloud, and baseline only; and a full decade of architectural evolution. What they share is one property: each implements a function $\mathcal{G}_k : \mathcal{X}_k \to \mathcal{Y}_k$ that is opaque to the user [25, 77]. The user may modify the input $x \in \mathcal{X}_k$ and observe the corresponding output $y = \mathcal{G}_k(x)$, but cannot inspect the intermediate representations, compare alternative outputs at sub-sequence granularity without regenerating the entire

Table 2.1: Models employed by or compared against MODULO, grouped by role. The **Deploy.** column is *Local* for models hosted inside MODULO, *Cloud* for models reached through a remote API from inside MODULO, and *Baseline* for models referenced only for comparison.

Model	**Task**	**Output**	**Architecture**	**Deploy.**
MusicGen	Music generation	Audio	Transformer over codec tokens	Baseline
Suno	Music generation	Audio	Proprietary, MusicGen family	Baseline
Udio	Music generation	Audio	Proprietary, MusicGen family	Baseline
ElevenLabs Music	Music generation	Audio	Proprietary	Cloud
ElevenLabs Stems	Source separation	Audio	Proprietary	Cloud
ElevenLabs SFX	Sound effects	Audio	Proprietary	Cloud
ReaLChords	Chord generation	Symbolic	Decoder-only transformer + RL	Local
Basic Pitch	Audio-to-MIDI	Symbolic	Multi-head CNN	Local
HTDemucs	Source separation	Audio	Hybrid transformer	Baseline

output, or selectively accept individual decisions while rejecting others within the same generation.

This limitation is a consequence of the interaction paradigm in which each model is deployed, not of the model itself. The same model that produces opaque, monolithic output when accessed through a prompt-and-receive interface can become a powerful co-creative tool when embedded within an interface that decomposes its output, presents alternatives in parallel, and supports selective commitment [53, 89]. The interface layer, not the model itself, is the primary determinant of whether a generative system supports composition or merely generation; this is the central thesis of this work. The remaining chapters detail the design, implementation, and evaluation of that interface layer.

CHAPTER 3

DAW Architecture and Systems Infrastructure

A Generative Audio Workstation inherits its production capabilities from the digital audio workstation tradition and its generative capabilities from the machine learning systems described in Chapter 2. This chapter describes the systems infrastructure bridging these two domains: the audio engine, the application framework, the alternative architectures considered, the limitations of commercial DAWs, and the formal architectural pattern defining the GAW category.

Section 3.1 establishes the foundational concepts of digital audio workstation design. Section 3.2 provides a technical deep dive into the open-source audio engine that powers MODULO. Section 3.3 describes the cross-platform application framework. Section 3.4 evaluates the primary open-source alternative considered during the design phase. Section 3.5 analyzes the limitations of commercial DAWs with respect to generative integration. Section 3.6 formalizes the GAW architecture pattern and situates MODULO within it.

3.1 DIGITAL AUDIO WORKSTATIONS: FOUNDATIONAL CONCEPTS

A digital audio workstation is a software system for recording, editing, mixing, and mastering audio [81]. At the computational level, a DAW implements a real-time signal processing graph, a directed acyclic graph (DAG) in which nodes represent audio processing operations and edges represent audio or control signal flow [118].

3.1.1 The Signal Processing Graph

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be the audio processing graph, where $\mathcal{N} = \{n_1, \ldots, n_M\}$ is the set of processing nodes and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of directed edges. Each node n_i implements a processing function

$$n_i : \mathbb{R}^{C_{\text{in}} \times B} \to \mathbb{R}^{C_{\text{out}} \times B}, \tag{3.1}$$

where C_{in} and C_{out} are input and output channel counts and B is the buffer size, measured in samples per processing block. The buffer size determines the fundamental tradeoff between latency and computational efficiency [91]:

$$\text{latency} = \frac{B}{f_s} \text{ seconds}, \tag{3.2}$$

where f_s is the sample rate. Typical configurations use $B \in \{64, 128, 256, 512, 1024\}$ at $f_s = 44100$ or 48000 Hz, yielding latencies from approximately 1.3 ms to 23 ms.

The graph is evaluated in topological order at each audio callback, with the real-time thread guaranteed to complete all node evaluations within the buffer period B/f_s. Failure to meet this deadline results in an audible discontinuity known as a buffer underrun, making deterministic worst-case execution time a critical design constraint.

3.1.2 The Project Data Model

The persistent state of a DAW session is organized hierarchically as a four-level structure:

$$\text{Project} \supset \mathcal{T} = \{T_1, \ldots, T_K\} \supset \text{Clips} \supset \text{Data}, \tag{3.3}$$

where each track T_k contains an ordered sequence of non-overlapping clips, and each clip references either audio sample data or MIDI event data. The project, also called an "edit," is the root container, encapsulating the complete state of a composition including track configuration, plugin parameters, automation envelopes, and temporal markers.

Non-destructive editing is a defining principle [81]: all modifications are represented as transformations applied to immutable source data, enabling unlimited undo and version comparison without data duplication. Formally, let $\mathbf{d}$ be the source data and $\phi_1, \phi_2, \ldots, \phi_J$ be a sequence of edit operations. The rendered output is

$$\mathbf{y} = (\phi_J \circ \phi_{J-1} \circ \cdots \circ \phi_1)(\mathbf{d}), \tag{3.4}$$

and any prefix $\phi_1, \ldots, \phi_j$ can be replayed to recover the state at edit step j.

3.2 THE TRACKTION ENGINE

MODULO is built on an open-source DAW engine [102] providing the audio processing, MIDI handling, plugin hosting, and project management infrastructure. This section describes the engine's architecture and the specific capabilities motivating its selection.

3.2.1 Architecture Overview

The engine implements the project data model described in Section 3.1 with a concrete class hierarchy rooted at the edit level. The edit is the fundamental unit of persistence,

containing:

(i) A set of tracks $\mathcal{T}$, each holding clips, plugins, and automation data;

(ii) A transport controller managing playback position, tempo map, and time signature;

(iii) An audio graph that compiles the track/clip/plugin configuration into a real-time processing DAG;

(iv) A project model providing file management, undo history, and serialization.

The audio graph is recompiled whenever the track or plugin configuration changes, producing a new DAG that is atomically swapped into the real-time audio thread. This design ensures that structural modifications, such as adding tracks, inserting plugins, or rearranging clips, never require locking the audio thread, preserving glitch-free playback during editing operations.

3.2.2 Real-Time Audio Processing

The engine's audio graph evaluates nodes in a topologically sorted order determined at compile time. For a graph with M nodes, the per-buffer computational cost is

$$C_{\text{buffer}} = \sum_{i=1}^{M} c(n_i, B), \tag{3.5}$$

where $c(n_i, B)$ is the cost of evaluating node n_i on a buffer of B samples. The engine supports configurable latency with automatic latency compensation: when a plugin introduces processing delay δ_i samples, upstream paths are padded to maintain temporal alignment:

$$\delta_{\text{path}}(n_j) = \max_{\text{paths } P:\text{source}\to n_j} \sum_{n_i \in P} \delta_i, \tag{3.6}$$

and each node's output is delayed by $\delta_{\text{path}}(n_j) - \delta_{\text{local}}(n_j)$ to ensure all signals arriving at any summing point are temporally aligned.

3.2.3 Plugin Hosting

The engine supports two major plugin standards through the application framework's plugin hosting abstraction:

(i) **VST3** [94]: Steinberg's cross-platform plugin standard, supporting audio effects and virtual instruments with parameter automation and preset management.

(ii) **Audio Unit (AU)** [8]: Apple's native plugin format for macOS, providing equivalent capabilities within the platform's audio infrastructure.

Each hosted plugin is wrapped as a node in the audio processing graph, with its parameters exposed for automation and its audio I/O routed according to the track's channel configuration. The wrapping handles the impedance mismatch between the plugin's expected buffer format and the engine's internal routing.

3.2.4 MIDI Subsystem

MIDI events are represented as timestamped messages within clips and processed through a dedicated MIDI pipeline:

(i) **Recording**: incoming MIDI from hardware controllers or virtual keyboards is timestamped against the transport position and stored in the active clip.

(ii) **Quantization**: recorded events are snapped to a configurable temporal grid, such as sixteenth notes or eighth-note triplets, with adjustable strength $\alpha \in [0, 1]$:

$$t_{\text{quantized}} = (1 - \alpha)\, t_{\text{original}} + \alpha\, t_{\text{grid}}, \tag{3.7}$$

where t_{grid} is the nearest grid position.

(iii) **Playback**: events are dispatched to their target plugin at the scheduled time, with sub-sample accuracy achieved through interpolation within the audio buffer.

3.2.5 Selection Rationale

The engine was selected over alternatives based on four criteria:

(i) **Open-source availability**: dual-licensed under GPL v3 and a commercial license, permitting academic use and future commercialization.

(ii) **API maturity**: well-documented programming interface with stable abstractions for track management, clip editing, plugin hosting, and audio routing.

(iii) **Framework integration**: native integration with the cross-platform application framework described in Section 3.3, sharing the same build system, audio device abstraction, and GUI rendering pipeline.

(iv) **Active development**: ongoing maintenance with regular releases, ensuring compatibility with current operating systems and plugin standards.

3.3 THE JUCE APPLICATION FRAMEWORK

The application framework [56] provides the cross-platform abstraction layer on which both the DAW engine and MODULO's custom user interface are built.

3.3.1 Core Capabilities

The framework offers four capabilities critical to this work:

(i) **Audio device abstraction**: a unified interface to platform-specific audio drivers, including CoreAudio on macOS, WASAPI/ASIO on Windows, and ALSA/JACK on Linux, with configurable sample rate, buffer size, and channel configuration.

(ii) **GUI rendering**: a retained-mode component hierarchy with GPU-accelerated 2D rendering, supporting the complex visual affordances required by timeline, piano roll, and chord editing interfaces.

(iii) **Plugin hosting**: scanner and wrapper classes for VST3 and Audio Unit plugins, managing plugin discovery, instantiation, parameter enumeration, and state persistence.

(iv) **Platform integration**: file I/O, threading, networking, and inter-process communication abstractions that compile to native code on macOS, Windows, and Linux.

The framework is implemented in C++ and used by a wide ecosystem of commercial audio products, providing a mature and well-tested foundation.

3.4 ALTERNATIVE CONSIDERED: ARDOUR

The primary open-source alternative evaluated during the design phase was Ardour [10], a mature DAW distributed under the GPL v2 license.

Ardour provides comprehensive multi-track recording, editing, and mixing capabilities with professional-grade audio quality. However, several architectural characteristics made it less suitable for the GAW use case:

(i) **GUI framework coupling**: the application's user interface is built on GTK+, a general-purpose widget toolkit not optimized for the custom, high-frequency visual updates required by real-time musical interfaces such as scrolling timelines, animated chord annotations, and responsive piano rolls.

(ii) **Audio backend constraints**: primary audio I/O is routed through JACK, an external audio connection manager that introduces configuration complexity and potential inter-process latency.

(iii) **Plugin hosting limitations**: while supporting major plugin standards, the hosting implementation is tightly integrated with the application's internal architecture, making it difficult to extend with custom plugin-like processing nodes of the kind required for the chord rendering and harmonization engines.

Figure 3.1: The Ardour DAW [10] showing its GTK+-based interface and conventional track/mixer layout. While comprehensive, its coupling to JACK audio routing and a general-purpose GUI toolkit made it less suitable for embedding real-time generative co-creation interfaces.

(iv) **Framework incompatibility**: the absence of JUCE integration means that none of the framework's plugin hosting, GUI rendering, or audio device abstractions can be shared between the engine and the application layer, requiring substantial re-implementation.

3.5 COMMERCIAL DAW LIMITATIONS

The dominant commercial DAWs [9, 1] represent decades of refined production workflow design, yet none currently provides the symbolic-level co-creation capabilities central to the GAW concept.

Platform-exclusive ecosystems. Major commercial DAWs are restricted to single operating systems or require platform-specific technologies, limiting both user access and developer extensibility. The closed-source nature of these products precludes the deep in-

Figure 3.2: Logic Pro arrangement view [9], a representative commercial DAW. Sophisticated multi-track editing, mixing, and plug-in hosting, but all musical content must be manually performed, recorded, or programmed; no generative co-creation tools are integrated.

tegration required to embed generative models within the audio processing graph or to create custom co-creation interfaces.

Absence of symbolic-level AI tools. While some commercial DAWs have introduced machine-learning-assisted features such as drum pattern suggestion, loop recommendation, and basic audio separation, none provides structured harmonic generation with parallel exploration, selective commitment, or in-context chord editing. The AI features that exist operate at the suggestion level, recommending presets, patterns, or samples, rather than at the compositional level of generating and presenting harmonic alternatives for musician evaluation.

Industry trajectory. The emergence of systems such as Suno Studio [98] and Mozart AI [71] signals that the industry recognizes the demand for generation-aware production environments. These platforms have matured rapidly: both now offer AI music generation,

audio-to-MIDI transcription, stem separation, and basic mixing within timeline-based interfaces, making them genuine GAW competitors. However, their approach has been to add production capabilities to generation platforms rather than to embed generation capabilities within a full production environment. As a result, the production infrastructure remains shallow: neither platform supports auxiliary bus routing, send/return signal chains, parametric equalization with per-band quality and gain controls, third-party plug-in hosting for VST3 or Audio Unit, or desktop-native execution. RipX DAW [48] takes yet another path, building on stem separation and audio-level note manipulation as a desktop tool, but without generative composition features. This thesis argues that building from the DAW side is architecturally sounder, as the production infrastructure of audio routing, plug-in hosting, mixing, and mastering is the harder component to replicate in a web environment and the more critical component for professional use. MODULO demonstrates this approach: a desktop-native C++ application with full audio engine infrastructure, onto which generative and co-creation capabilities are layered.

3.6 THE GAW ARCHITECTURE PATTERN

Drawing on the analysis of the preceding sections, we formalize the Generative Audio Workstation as an architectural pattern that extends the traditional DAW with generation services and co-creation interfaces.

3.6.1 Formal Definition

Recalling Definition 1 from Chapter 1, a GAW is a triple $\mathcal{W} = (\mathcal{D}, \mathcal{G}, \mathcal{I})$. We now refine this into a layered architecture:

$$\mathcal{W} = \underbrace{\mathcal{U}}_{\text{Presentation}} \circ \underbrace{\mathcal{S}}_{\text{Session}} \circ \underbrace{\mathcal{D}}_{\text{Engine}} \circ \underbrace{(\mathcal{G}_{\text{local}} \cup \mathcal{G}_{\text{cloud}})}_{\text{Services}}, \tag{3.8}$$

where:

- $\mathcal{U}$ is the **presentation layer**: the musician-facing user interface, including the timeline, chord inspector, piano roll, mixer, and all co-creation affordances;
- $\mathcal{S}$ is the **session layer**: an abstraction that mediates between the presentation layer and the engine, managing application state, undo/redo, and the mapping between user actions and engine operations;
- $\mathcal{D}$ is the **engine layer**: the DAW core providing real-time audio processing, MIDI handling, plugin hosting, and project persistence;
- $\mathcal{G}_{\text{local}}$ and $\mathcal{G}_{\text{cloud}}$ are **service layers**: local inference servers for chord generation and audio-to-MIDI transcription, and cloud API endpoints for music generation, stem separation, and sound effects synthesis.

3.6.2 Service-Oriented Generation

The generation services are decoupled from the engine via a service-oriented architecture. Each service $\mathcal{G}_k$ exposes a request–response interface:

$$\mathcal{G}_k : \mathcal{X}_k \to \mathcal{F}(\mathcal{Y}_k), \tag{3.9}$$

where $\mathcal{X}_k$ is the request space, $\mathcal{Y}_k$ is the response space, and $\mathcal{F}(\cdot)$ denotes an asynchronous future. This abstraction permits:

(i) **Location transparency**: local and cloud services are accessed through the same interface, with the session layer handling the differences in latency and reliability.

(ii) **Parallel dispatch**: multiple generation requests, such as the N parallel chord columns, can be dispatched concurrently.

(iii) **Graceful degradation**: if a cloud service is unavailable, the system can fall back to local alternatives or disable the corresponding feature without affecting the DAW core.

3.6.3 Tradeoffs: Desktop-Native vs. Web-Based

The architectural choice between a desktop-native application, as in MODULO, and a web-based platform, as in several competitors, involves fundamental tradeoffs:

(i) **Latency**: desktop applications access audio hardware directly via native drivers, achieving sub-10ms round-trip latency. Web-based systems incur additional latency from the browser's audio stack [110], network round-trips for cloud inference, and JavaScript execution overhead.

(ii) **Privacy**: local inference ensures that musical material never leaves the musician's machine, a significant consideration for professional and pre-release content.

(iii) **Capability**: desktop applications can host native plugins such as VST3 [94] and AU [8], access the full filesystem, and utilize local GPU resources for inference. Web-based systems are constrained by browser sandboxing.

(iv) **Distribution**: web-based systems require no installation and update transparently. Desktop applications require explicit distribution and platform-specific packaging.

MODULO adopts the desktop-native approach, prioritizing latency, privacy, and plugin hosting capability at the cost of distribution complexity.

3.6.4 Implications for the Remaining Chapters

The layered architecture organises the rest of the thesis: Chapters 4–10 describe the generative systems spanning session, engine, local, and cloud layers, while Chapters 11–12

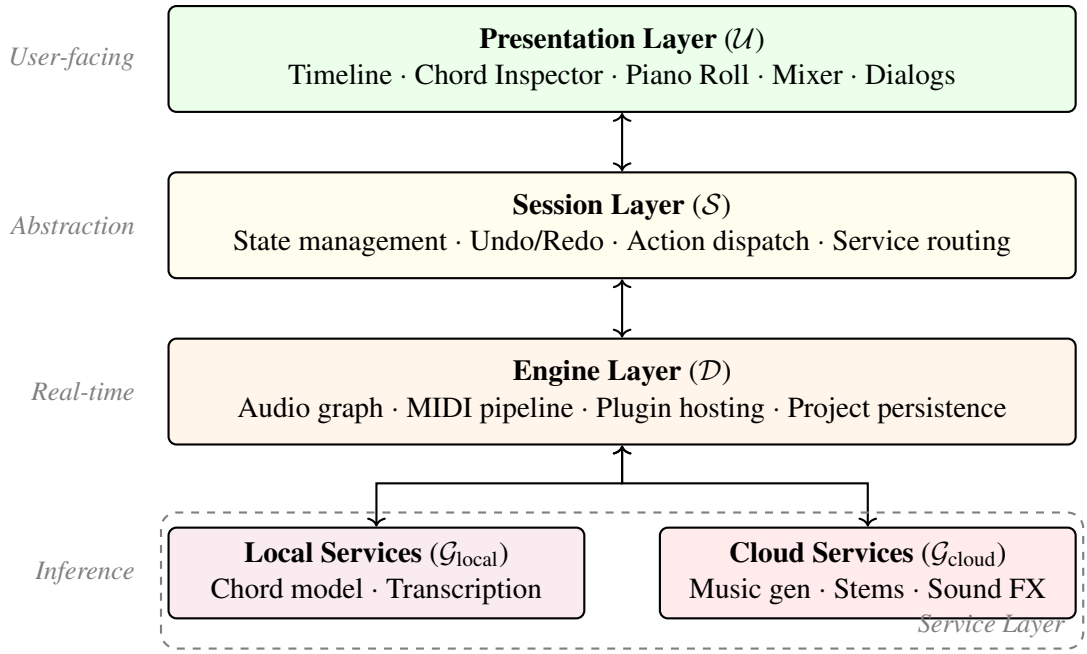


Figure 3.3: Layered architecture of MODULO. The presentation layer provides musician-facing co-creation interfaces; the session layer manages application state and action dispatch; the engine layer provides real-time audio processing and project persistence; the service layer hosts local and cloud-based generative models. Bidirectional arrows indicate synchronous communication at the engine layer and asynchronous communication at the service layer.

cover the engine and presentation layers. In each case, the layered separation ensures generative capabilities can be added, replaced, or disabled without disrupting the production infrastructure.

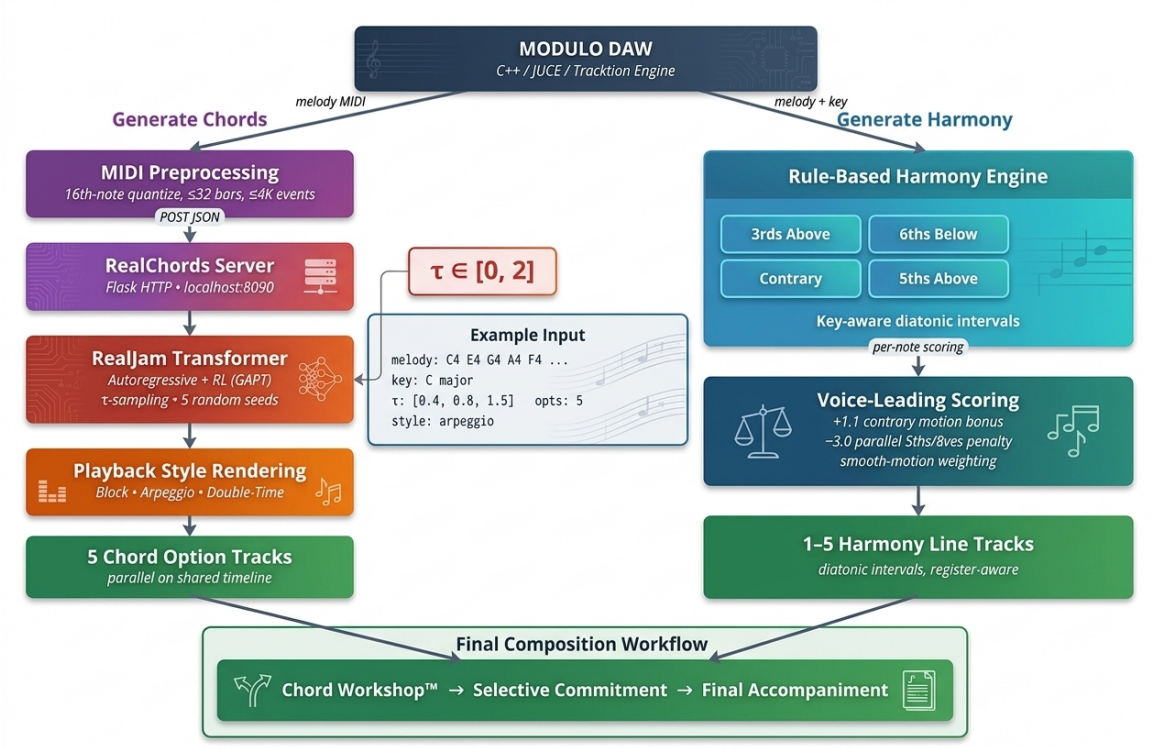


Figure 3.4: Implementation-level view of the same architecture diagrammed in Figure 3.3, annotated with the concrete subsystems and external services that realise each layer: the JUCE/Tracktion engine, the local Basic Pitch and HTDemucs services, the chord and harmony engines, and the cloud-hosted ElevenLabs and MusicGen endpoints. The arrows trace the request/response paths exercised by the user studies in Chapters 4–10.

Part II: Harmonic Co-Creation

CHAPTER 4

Chord Generation System

The task of harmonizing a given melody is among the oldest problems in computational music theory [32, 47], yet existing tools overwhelmingly frame it as a batch optimization: the user submits a melody, receives a single "best" harmonization, and must accept or reject the result wholesale.[1] This chapter presents the parallel harmonic ideation subsystem of MODULO, which reframes chord generation as a stochastic exploration process inspired by the ReaLchords paradigm [111]. In this framing, the musician auditions, compares, and selectively commits to individual harmonic choices drawn from multiple simultaneous temperature-controlled inference streams.

Section 4.1 describes the end-to-end architecture. Section 4.2 formalizes the underlying autoregressive model and its reinforcement-learning fine-tuning procedure. Section 4.3 derives the temperature-controlled sampling distribution. Section 4.4 specifies the preprocessing and quantization pipeline. Section 4.5 defines the mapping from chord symbols to rendered MIDI under each playback style. Section 4.6 introduces the selective commitment mechanism. Section 4.7 describes the auditioning workflow. Section 4.9 reports the full evaluation results from the within-subjects user study; the cross-study synthesis that places these results alongside the other five studies lives in Chapter 14.

[1] An earlier presentation of the chord-generation and harmony engine work covered in Chapters 4 through 6 was prepared as *Parallel Harmonic Exploration for Musical Accompaniment with MODULO*, with co-authors T. Khoo, P. Chin, and N. Singh, and submitted to UIST 2026. See the *Prior Submissions* note in the Preface.

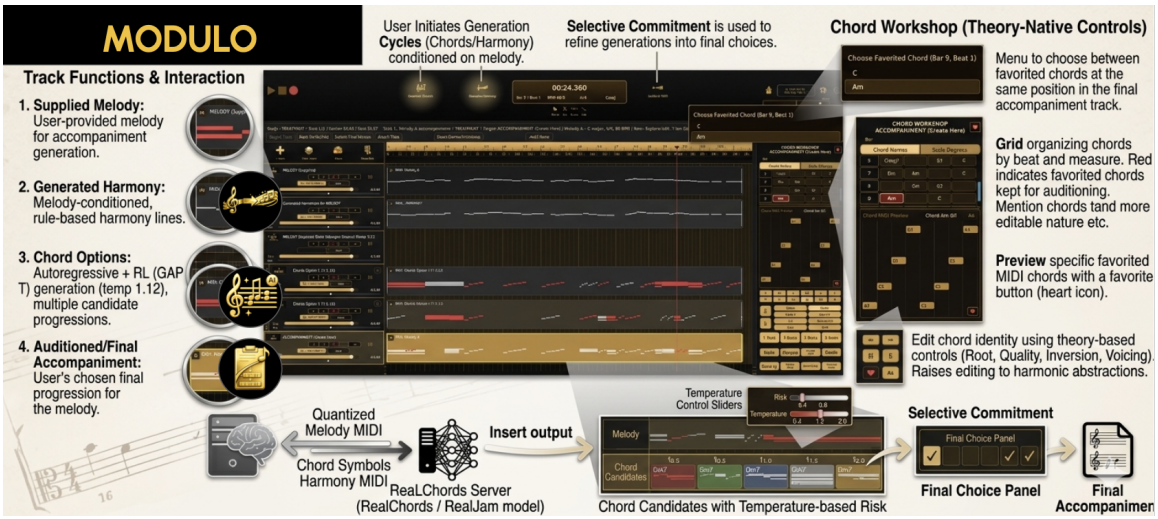


Figure 4.1: Overview of MODULO's coupled chord generation and harmony co-creation surface, originally produced as the header figure for the UIST submission. Parallel temperature-controlled chord columns on the left feed selectively committed chord progressions into the harmony engine on the right, where multi-line voicings are realised against the same melody. The two systems share a unified piano-roll canvas, the same selective-commitment vocabulary, and the same playback transport.

4.1 SYSTEM ARCHITECTURE AND INFERENCE PIPELINE

Let $\mathbf{m} = (m_1, m_2, \ldots, m_T)$ denote a melody represented as a sequence of MIDI events spanning T sixteenth-note frames. The generation subsystem defines a pipeline operator

$$\mathcal{P}_{\text{chord}} : \mathcal{M} \times \Theta^N \longrightarrow \mathcal{C}^N, \tag{4.1}$$

where $\mathcal{M}$ is the space of preprocessed melody sequences, Θ is the parameter space of a single inference column spanning temperature τ, random seed σ, lookahead window ℓ, and playback style ρ; $N \leq 5$ is the column count, and $\mathcal{C}$ is the space of chord sequences aligned to the same temporal grid as $\mathbf{m}$.

The pipeline proceeds in four stages:

Stage 1: Preprocessing. The raw melody MIDI is quantized, trimmed, and thinned into a

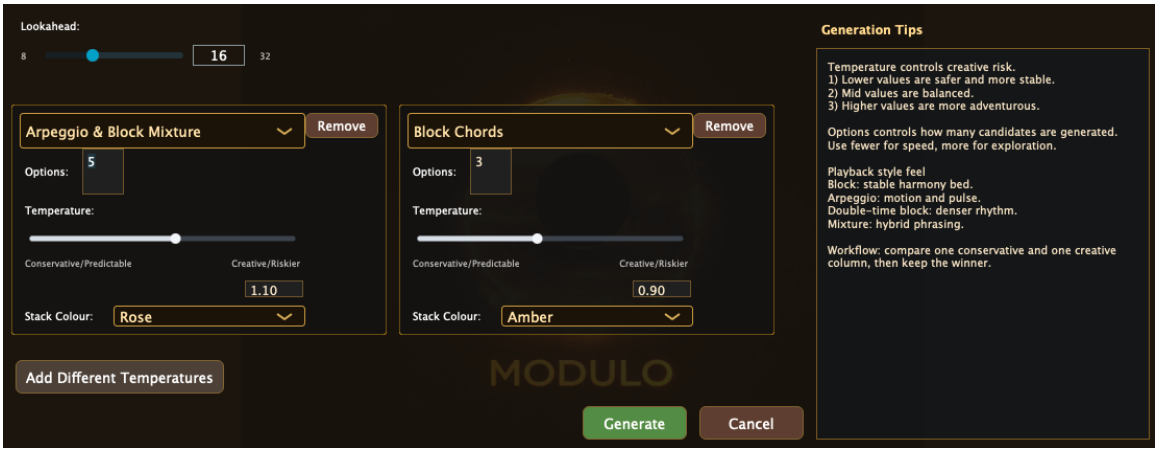


Figure 4.2: The *Generate Chords* dialog in MODULO. Controls expose column count, temperature, playback style, and lookahead window; a melody track is visible behind the dialog.

fixed-length event representation (Section 4.4).

Stage 2: Parallel inference. N independent forward passes through the autoregressive model produce N candidate chord sequences, each conditioned on the same melody but sampled under distinct (τ_i, σ_i) pairs (Section 4.3).

Stage 3: Rendering. Each chord sequence is converted to MIDI events according to its assigned playback style ρ_i (Section 4.5).

Stage 4: Selective commitment. The musician auditions columns, marks favorites, and commits individual chords from potentially different columns into the final harmonic track (Section 4.6).

The local inference server wraps the pretrained model in a lightweight HTTP service that accepts melody event sequences and returns chord tokens. By hosting inference locally, the system avoids network latency and permits deterministic replay when the seed is fixed.

4.2 THE AUTOREGRESSIVE CHORD MODEL

The inference engine is built on a decoder-only autoregressive transformer [108] that models the conditional distribution of chord sequences given a melody context. Let $\mathbf{c}_{<t} = (c_1, \ldots, c_{t-1})$ denote the chord history up to frame t, and let $\mathbf{m}$ denote the full preprocessed melody. The model factorizes the joint probability as

$$p(\mathbf{c} \mid \mathbf{m}) = \prod_{t=1}^{T} p(c_t \mid \mathbf{c}_{<t},\, \mathbf{m}), \tag{4.2}$$

where each factor is parameterized by a transformer with parameters θ.

4.2.1 Transformer Architecture

The model follows the standard decoder-only design [108] with L layers of masked multi-head self-attention and position-wise feed-forward networks. Let $\mathbf{x}_t \in \mathbb{R}^d$ be the input embedding at position t, formed by summing a token embedding and a sinusoidal positional encoding:

$$\mathbf{x}_t = \mathbf{E}_{\text{tok}}[c_t] + \mathbf{E}_{\text{pos}}[t], \tag{4.3}$$

where $\mathbf{E}_{\text{tok}} \in \mathbb{R}^{|V| \times d}$ is the chord vocabulary embedding matrix and $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{T_{\max} \times d}$ encodes temporal position.

At each layer $l \in \{1, \ldots, L\}$, the hidden state $\mathbf{h}_t^{(l)}$ is computed via masked multi-head attention followed by a feed-forward sublayer, each with residual connections and layer normalization:

$$\tilde{\mathbf{h}}_t^{(l)} = \text{LayerNorm}\big(\mathbf{h}_t^{(l-1)} + \text{MHA}^{(l)}(\mathbf{h}_{\le t}^{(l-1)})\big), \tag{4.4}$$

$$\mathbf{h}_t^{(l)} = \text{LayerNorm}\big(\tilde{\mathbf{h}}_t^{(l)} + \text{FFN}^{(l)}(\tilde{\mathbf{h}}_t^{(l)})\big), \tag{4.5}$$

where $\text{MHA}^{(l)}$ denotes multi-head attention with causal masking, and $\text{FFN}^{(l)}$ is a two-layer

network with a nonlinear activation.

The melody conditioning enters via cross-attention or, in the architecture employed here, via a prefixed melody token sequence prepended to the chord token sequence, so that the causal mask permits chord tokens to attend to all melody tokens. The output logit vector at position t is

$$\mathbf{z}_t = \mathbf{W}_{\text{out}}\,\mathbf{h}_t^{(L)} + \mathbf{b}_{\text{out}}, \quad \mathbf{z}_t \in \mathbb{R}^{|V|}, \tag{4.6}$$

where $\mathbf{W}_{\text{out}}$ projects from the hidden dimension to the vocabulary size $|V|$.

4.2.2 Reinforcement Learning Fine-Tuning

The base model is pretrained on a large corpus of melody–chord pairs via maximum likelihood estimation. Fine-tuning employs reinforcement learning (RL) with a reward signal composed of two complementary terms: a contrastive reward R_{con} and a discriminative reward R_{disc}.

The contrastive reward operates on pairs of generated harmonizations. Given two candidate chord sequences $\mathbf{c}^{(a)}$ and $\mathbf{c}^{(b)}$ for the same melody $\mathbf{m}$, a learned preference model f_ϕ assigns a scalar preference:

$$R_{\text{con}}(\mathbf{c}^{(a)}, \mathbf{c}^{(b)} \mid \mathbf{m}) = \log \frac{\exp\big(f_\phi(\mathbf{c}^{(a)}, \mathbf{m})\big)}{\exp\big(f_\phi(\mathbf{c}^{(a)}, \mathbf{m})\big) + \exp\big(f_\phi(\mathbf{c}^{(b)}, \mathbf{m})\big)}. \tag{4.7}$$

The discriminative reward uses a binary classifier g_ψ trained to distinguish human-composed chord progressions from model-generated ones:

$$R_{\text{disc}}(\mathbf{c} \mid \mathbf{m}) = \log g_\psi(\text{“human”} \mid \mathbf{c}, \mathbf{m}). \tag{4.8}$$

This combination constitutes Generative Adversarial Post-Training (GAPT), a variant in which the generator, that is the chord model, is updated via proximal policy optimization

(PPO) [86] to maximize a composite reward:

$$R(\mathbf{c} \mid \mathbf{m}) = \alpha\, R_{\text{con}}(\mathbf{c} \mid \mathbf{m}) + \beta\, R_{\text{disc}}(\mathbf{c} \mid \mathbf{m}) - \gamma\, D_{\text{KL}}\big[\pi_\theta(\cdot \mid \mathbf{m}) \,\|\, \pi_{\text{ref}}(\cdot \mid \mathbf{m})\big], \tag{4.9}$$

where π_θ is the current policy, π_{ref} is the frozen pretrained policy, α and β weight the reward terms, and the KL-divergence penalty controlled by γ prevents mode collapse by anchoring the fine-tuned distribution near the pretrained one.

The PPO objective clips the importance-sampling ratio $r_t(\theta) = \pi_\theta(c_t \mid \mathbf{c}_{<t}, \mathbf{m}) / \pi_{\theta_{\text{old}}}(c_t \mid \mathbf{c}_{<t}, \mathbf{m})$ as

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon)\,\hat{A}_t\big)\Big], \tag{4.10}$$

where $\hat{A}_t$ is the generalized advantage estimate and ϵ is the clipping hyperparameter.

4.3 TEMPERATURE-CONTROLLED EXPLORATION

At inference time, the model generates chord tokens by sampling from a temperature-scaled softmax distribution. Given the logit vector $\mathbf{z}_t$ at frame t (Equation 4.6), the sampling probability for chord token c is

$$p(c_t = c \mid \mathbf{c}_{<t},\, \mathbf{m};\, \tau) = \frac{\exp(z_{t,c} \,/\, \tau)}{\displaystyle\sum_{c' \in V} \exp(z_{t,c'} \,/\, \tau)}, \tag{4.11}$$

where $\tau \in (0, 2]$ is the temperature parameter.

The temperature controls the entropy of the sampling distribution. In the limit $\tau \to 0^+$, the distribution collapses to a point mass on the highest-logit token, yielding greedy decoding, while $\tau > 1$ flattens the distribution and increases the probability of lower-

Figure 4.3: A single dialog invocation that exercises both the temperature axis and the playback-style axis at once: five simultaneous inference columns sweep τ from conservative on the left to exploratory on the right, and the four playback styles of block, arpeggio, double-time, and a fourth combined variant are simultaneously assigned across the columns. Color-coded chord cells reveal how low-temperature voicings on the left give way to increasingly chromatic progressions on the right while the per-column playback style determines the rendered MIDI realization, as defined in Section 4.5.

ranked tokens. The entropy H_τ of the temperature-scaled distribution satisfies

$$\frac{\partial H_\tau}{\partial \tau} > 0 \quad \text{for all } \tau > 0, \tag{4.12}$$

ensuring that higher temperatures always yield greater harmonic diversity.

The system permits up to $N = 5$ columns to execute simultaneously. Each column i is parameterized by an independent tuple (τ_i, σ_i), where σ_i is the pseudorandom seed. Fixing the seed guarantees reproducibility: given identical preprocessing and model weights, the same (τ_i, σ_i) pair produces the same chord sequence, enabling deterministic comparison

across audition sessions.

A lookahead parameter ℓ controls how many beats ahead the model conditions on during generation. Formally, the effective context at frame t includes not only $\mathbf{c}_{<t}$ and $\mathbf{m}_{\leq t}$ but also the melody future $\mathbf{m}_{t+1:t+\ell}$, allowing the model to anticipate upcoming melodic motion when selecting the current chord.

4.4 PREPROCESSING AND QUANTIZATION PIPELINE

Raw MIDI from the melody track must be transformed into a fixed-format representation before inference. The preprocessing pipeline $\mathcal{P}_{\text{pre}}$ consists of three sequential operations.

Temporal quantization. Each note-on event with onset time $t_{\text{on}} \in \mathbb{R}_{\geq 0}$, measured in beats, is mapped to the nearest sixteenth-note grid point:

$$\hat{t}_{\text{on}} = \frac{1}{4} \left\lfloor 4\, t_{\text{on}} + 0.5 \right\rfloor, \tag{4.13}$$

yielding a temporal resolution of four frames per beat. Note durations are quantized analogously. This grid defines the frame index space $\{1, 2, \ldots, T\}$ over which the chord model operates.

Temporal trimming. The quantized sequence is truncated to at most $B_{\max} = 32$ bars. With a time signature of $\frac{4}{4}$ and four frames per beat, this yields a maximum sequence length of

$$T_{\max} = B_{\max} \times 4 \times 4 = 512 \text{ frames}. \tag{4.14}$$

Event thinning. If the total number of MIDI events after quantization exceeds $E_{\max} = 4{,}000$, a thinning procedure retains the $E_{\max}$ events with the highest velocity values, pre-

serving the most perceptually salient notes:

$$\mathbf{m}_{\text{thin}} = \text{Top}_{E_{\max}}\big(\mathbf{m}_{\text{quant}},\ \text{key} = v_i\big), \tag{4.15}$$

where v_i denotes the velocity of the i-th event.

The composite preprocessing operator is thus

$$\mathcal{P}_{\text{pre}} = \text{Thin}_{E_{\max}} \circ \text{Trim}_{B_{\max}} \circ \text{Quantize}_{\Delta}, \tag{4.16}$$

where $\Delta = \frac{1}{4}$ beat is the quantization resolution.

4.5 PLAYBACK STYLE RENDERING

Each generated chord symbol $c_t = (r_t, q_t)$, consisting of a root pitch class $r_t \in \{0, 1, \ldots, 11\}$ and quality $q_t \in Q$, must be rendered as concrete MIDI note events for audition. Let $\phi : \{0, \ldots, 11\} \times Q \to \mathcal{P}(\mathbb{Z})$ be the voicing function that maps a chord symbol to a set of MIDI note numbers, where $\mathcal{P}(\mathbb{Z})$ denotes the power set. For a chord spanning δ beats, the three playback styles are defined as follows.

Block style. All notes in $\phi(r_t, q_t)$ are attacked simultaneously at the chord onset and sustained for the full duration δ:

$$\text{Block}(c_t, t_0, \delta) = \big\{(n,\ t_0,\ \delta)\ :\ n \in \phi(r_t, q_t)\big\}. \tag{4.17}$$

Arpeggio style. Notes are sequenced in ascending pitch order with equal subdivision. Let $\phi(r_t, q_t) = \{n_1, n_2, \ldots, n_k\}$ with $n_1 < n_2 < \cdots < n_k$. The j-th note is attacked at

$$t_j = t_0 + \frac{(j-1)\,\delta}{k}, \quad \text{with duration } \frac{\delta}{k}, \tag{4.18}$$

for $j \in \{1, \ldots, k\}$.

Double-time style. The chord is re-attacked at twice the harmonic rhythm. The full duration δ is divided into two equal segments, and a block voicing is rendered in each:

$$\text{DblTime}(c_t, t_0, \delta) = \text{Block}(c_t, t_0, \tfrac{\delta}{2}) \cup \text{Block}(c_t, t_0 + \tfrac{\delta}{2}, \tfrac{\delta}{2}). \tag{4.19}$$

4.6 SELECTIVE COMMITMENT

A central design principle of the chord generation subsystem is *selective commitment*: the musician is not required to accept or reject an entire column as a monolithic unit. Instead, individual chords may be chosen from different columns and assembled into a composite harmonic track.

Formally, let $\mathbf{c}^{(i)} = (c_1^{(i)}, \ldots, c_T^{(i)})$ denote the chord sequence generated by column $i \in \{1, \ldots, N\}$. The final committed chord sequence is a selection function

$$\mathbf{c}_t^* = c_t^{(\sigma(t))}, \quad \text{where } \sigma : \{1, \ldots, T\} \to \{1, \ldots, N\} \tag{4.20}$$

maps each temporal position t to the column index from which the chord at that position is drawn. The selection function σ is defined interactively by the musician through the auditioning interface.

This mechanism supports a combinatorial exploration of the harmonic space. Given N columns and T chord positions, the musician can construct any of N^T possible composite sequences, vastly exceeding the N monolithic alternatives available under a column-level accept/reject paradigm.

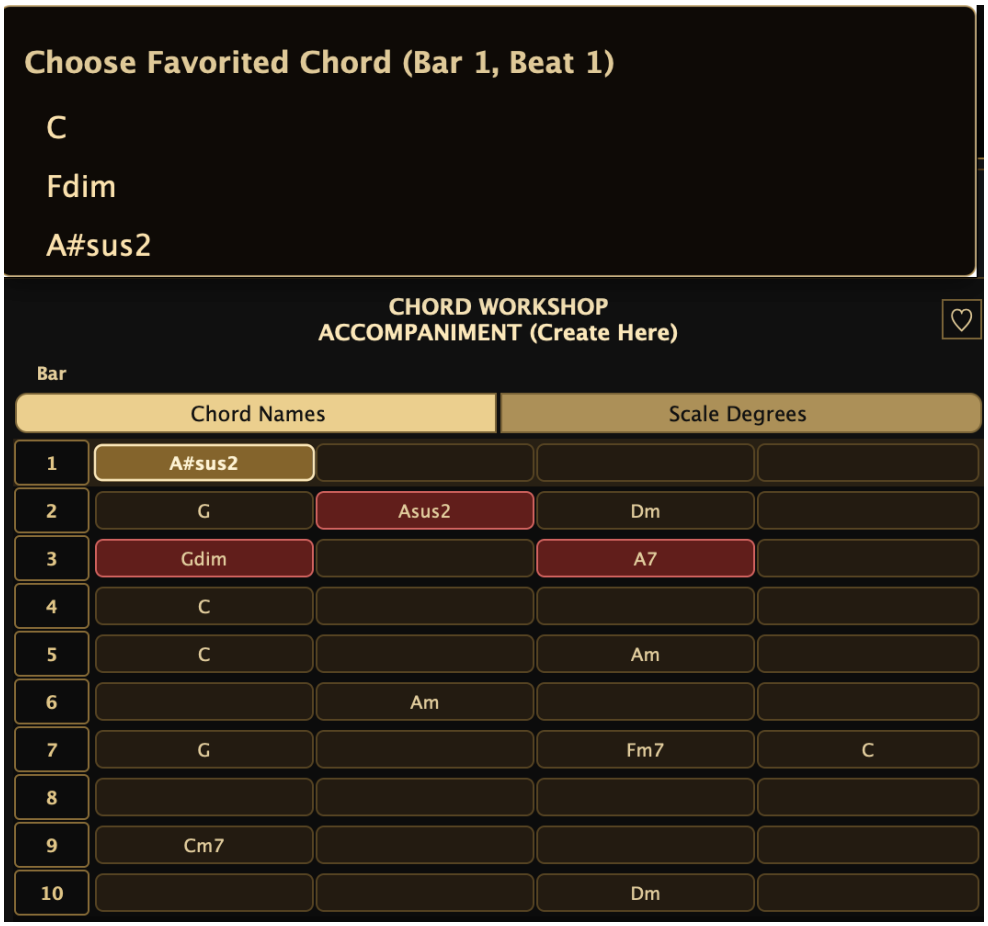


Figure 4.4: The selective commitment panel. Cells inherit the color of the column they were drawn from, so the committed sequence $\mathbf{c}^*$ visually records which columns contributed which chords.

4.7 AUDITIONING WORKFLOW

The interface provides three auditioning operations that support rapid evaluation without modifying the underlying data:

(a) **Solo.** Isolates a single column for playback, muting all other generated tracks and optionally the original melody, so the musician can evaluate the harmonic progression in isolation.

(b) **Favorite.** Marks a column or individual chord as a candidate for commitment without immediately finalizing the choice, enabling the musician to shortlist options be-

fore committing.

(c) **Compare.** Enables simultaneous or sequential playback of two or more columns, facilitating direct A/B comparison of harmonic alternatives at the same temporal position.

These operations are non-destructive: they affect only the monitoring state and leave the generated data and the selection function σ unchanged until the musician explicitly commits.

4.8 PIPELINE DIAGRAM

Figure 4.5 presents a schematic view of the complete chord generation pipeline.

4.9 EVALUATION: USER STUDY RESULTS

The chord generation subsystem was evaluated in a within-subjects user study ($N = 25$) comparing a *treatment* condition of parallel multi-temperature chord generation with selective commitment against a *control* condition using conventional sequential single-option chord editing. Each participant harmonized two melodies, Melody A in C major and Melody B in F minor, under each condition; see Figure 4.6. This yielded $25 \times 2 = 50$ paired per-melody observations for every dependent variable. All metrics are reported *per melody* so the numbers reflect the effort required to harmonise a single melody, not the aggregate across the full session. Paired t-tests are reported with Benjamini–Hochberg correction [13] at $\alpha = 0.05$ (Equation 14.1); effect sizes are reported as Cohen's d_z for within-subjects designs.

4.9.1 Quantitative Results

Table 4.1 summarises the primary quantitative findings.

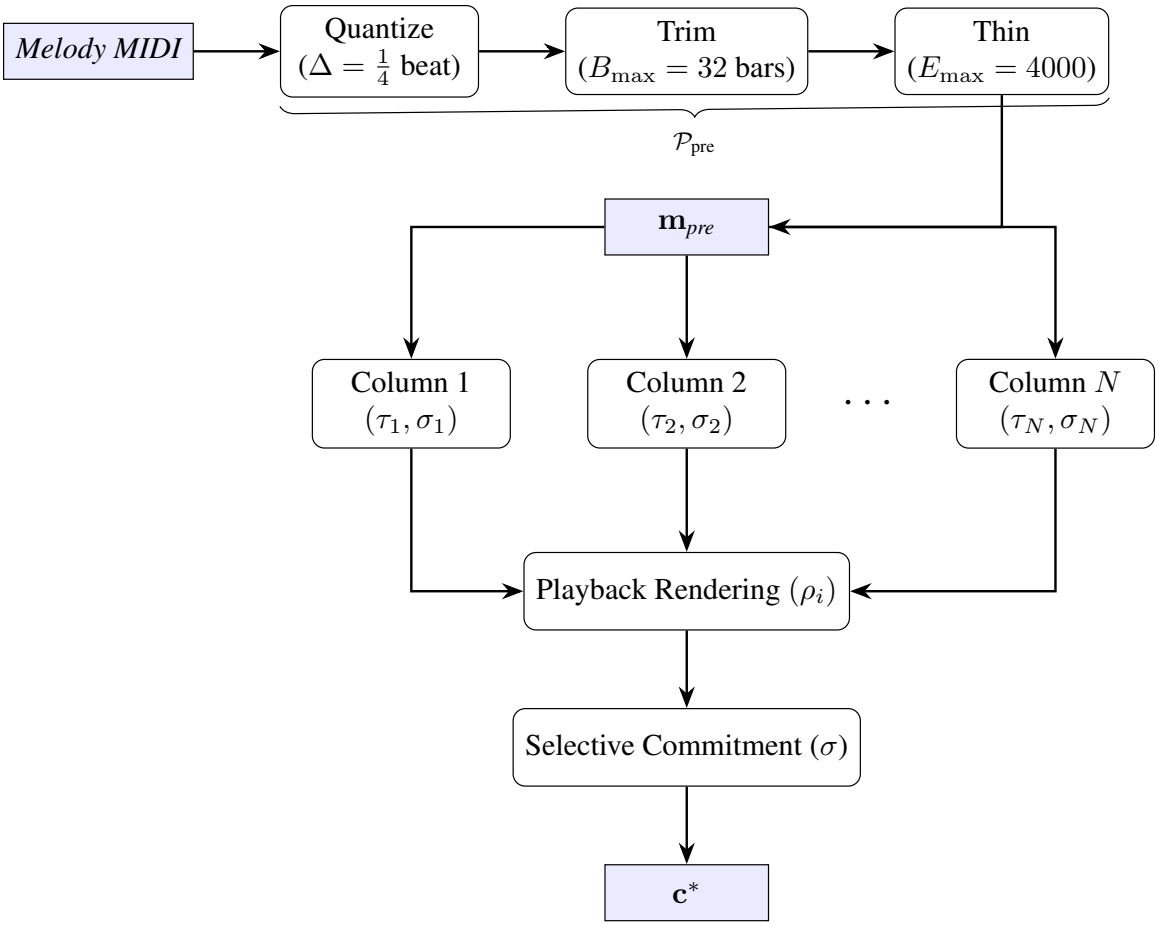
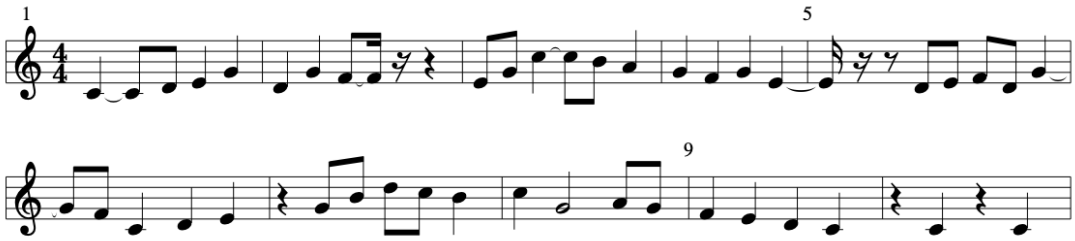


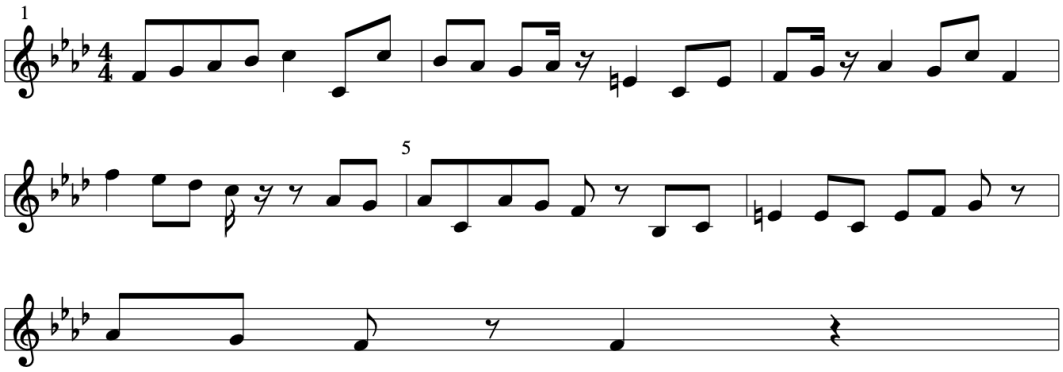
Figure 4.5: Chord generation pipeline. Melody MIDI is preprocessed through $\mathcal{P}_{\text{pre}}$: quantize, trim, thin. The preprocessed melody is then fed in parallel through N sampling columns with distinct (τ_i, σ_i). Rendered previews are selectively committed to produce the final chord vector $\mathbf{c}^*$.

The results reveal a striking temporal redistribution. Under the treatment condition, participants reached an initial satisfactory harmonization in approximately half the time required under the control condition ($\Delta = -81.9\,\text{s}$, $d_z = -2.54$). Rather than terminating earlier, participants reinvested the saved time into extended refinement ($\Delta = +112.7\,\text{s}$, $d_z = 1.62$), resulting in a modest net increase in total session duration ($\Delta = +30.9\,\text{s}$, $d_z = 0.49$). This pattern, faster arrival at a satisfactory baseline followed by deeper exploration, is consistent with the selective commitment hypothesis: when the cost of generating alternatives is low, practitioners shift from satisficing to exploring.

The harmonic breadth measures confirm that the treatment condition produced substan-



(a) Melody A: C major, 4/4, 80 beats per minute (BPM). Diatonic stepwise contour with one upper-neighbour figure; designed to admit canonical I–vi–IV–V style harmonisations as well as more chromatic alternatives.



(b) Melody B: F minor, 4/4, 70 BPM. Modal contour with a flat-VI inflection that opens space for borrowed harmonies and minor–major modulation.

Figure 4.6: The two stimulus melodies used in both the chord generation study of Chapter 4 and the harmony engine study of Chapter 6. Identical stimuli across studies enable cross-study comparison of harmonisation strategies under different feature affordances.

tially more diverse chord progressions. Unique chord roots increased by a factor of $3.3\times$, and unique chord qualities by $3.8\times$, both with very large effect sizes ($d_z > 3$). Root motion entropy, a measure of the unpredictability of successive root transitions,

$$H_{\text{root}} = -\sum_{r \in \mathcal{R}} p(r) \log_2 p(r), \tag{4.21}$$

where $p(r)$ is the empirical frequency of root r in the final chord sequence, showed a statistically significant but modest increase ($d_z = 0.45$), suggesting participants expanded their harmonic vocabulary without producing incoherent randomness.

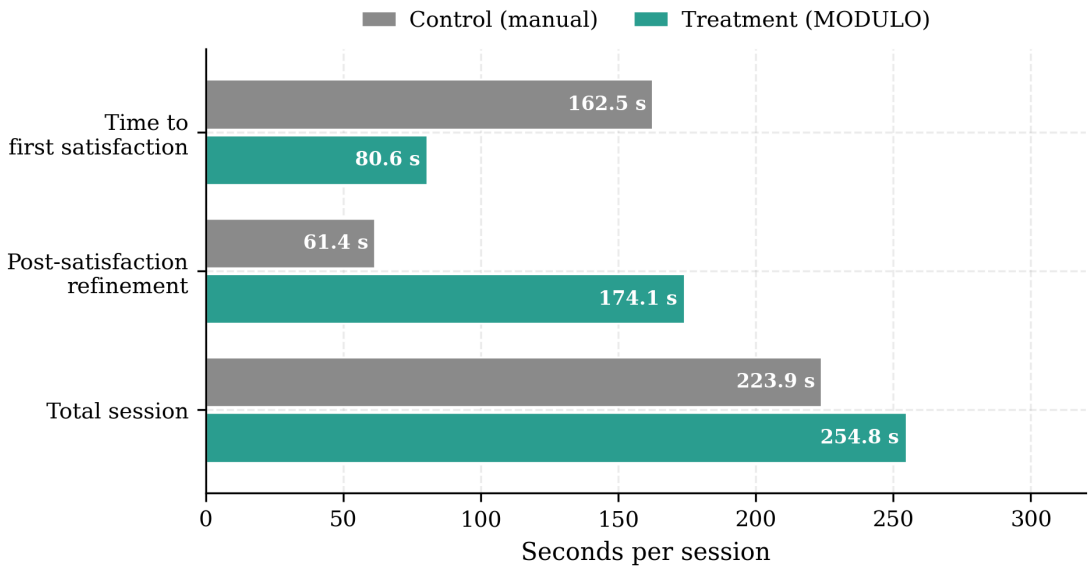


Figure 4.7: Temporal redistribution of creative effort ($N = 25$). Treatment halved time to initial satisfaction ($d_z = -2.54$) while nearly tripling post-satisfaction refinement time ($d_z = 1.62$), producing a modest net increase in total session duration. This is the reinvestment-into-exploration pattern recurring across the thesis: faster arrival at a baseline does not translate into earlier stopping.

4.9.2 Subjective Ratings

All seven custom Likert items were rated significantly higher under the treatment condition. Table 4.2 presents the headline subjective results; the full item list appears with the custom Likert batteries in Appendix C.

The most dramatic shift occurred on the exploration item, "The tool helped me explore multiple harmonic directions," which moved from a floor effect at 1.00 in the control condition to a ceiling effect at 5.00 in the treatment condition: a unanimous maximum rating across all fifty paired observations.

The NASA-TLX [43] subscales revealed large reductions in cognitive cost. Mental demand decreased from 4.32 to 1.76, effort from 4.48 to 1.36, and frustration from 3.88 to 1.00, all on a five-point scale adapted from the standard twenty-one-point instrument. These reductions indicate the parallel generation paradigm not only improved creative outcomes but also reduced the subjective cost of achieving them.

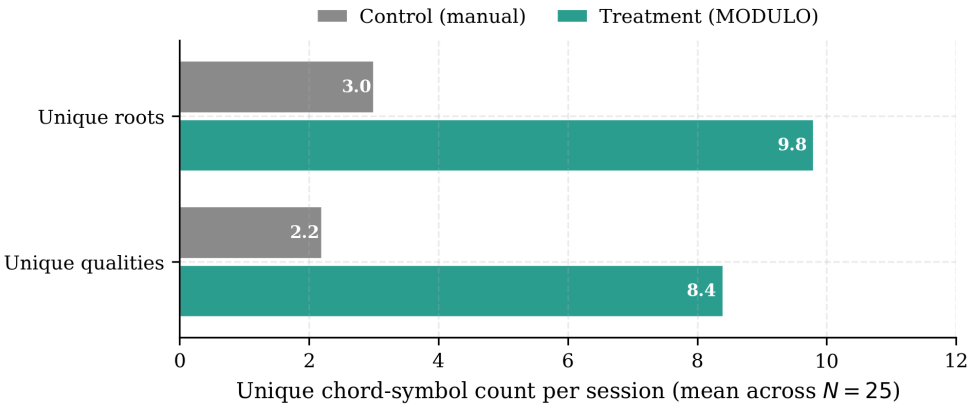


Figure 4.8: Harmonic vocabulary breadth ($N = 25$). Treatment increased unique chord roots by $3.3\times$ ($d_z = 3.36$) and unique chord qualities by $3.8\times$ ($d_z = 4.30$), confirming that parallel multi-temperature generation expands the harmonic search space without incoherent drift; root-motion entropy rose only 0.2 bits, as shown in Table 4.1.

Table 4.1: Quantitative results from the chord generation study. Means are *per melody*, computed over 50 paired observations from 25 participants harmonising 2 melodies, so $t(49)$ reflects the per-melody distribution. p-values are Benjamini–Hochberg adjusted; CIs are 95% on Δ.

Metric	**Ctrl** $\bar{x}$	**Tx** $\bar{x}$	Δ (95% CI)	$t(49)$	p_{adj}	d_z
Time to satisfaction (s)	162.5	80.6	$-81.9\,[-90.8, -72.9]$	-17.99	$<.001$	-2.54
Post-sat. refinement (s)	61.4	174.1	$+112.7\,[93.4, 132.0]$	11.45	$<.001$	1.62
Total session time (s)	223.9	254.8	$+30.9\,[13.4, 48.3]$	3.47	.002	0.49
Unique roots $\lvert\mathcal{R}\rvert$	3.0	9.8	$+6.7\,[6.2, 7.3]$	23.75	$<.001$	3.36
Unique qualities $\lvert\mathcal{Q}\rvert$	2.2	8.4	$+6.1\,[5.7, 6.5]$	30.39	$<.001$	4.30
Root-motion entropy H_{root}	1.2	1.4	$+0.2\,[0.1, 0.3]$	3.15	.003	0.45

4.9.3 Qualitative Themes

Semi-structured post-condition interviews were transcribed and analyzed using reflexive thematic analysis [17]. Three themes emerged:

Theme 1: Expanded harmonic vocabulary. Participants reported encountering chord types they would not have considered independently, shifting them from habitual patterns toward less familiar but musically compelling alternatives.

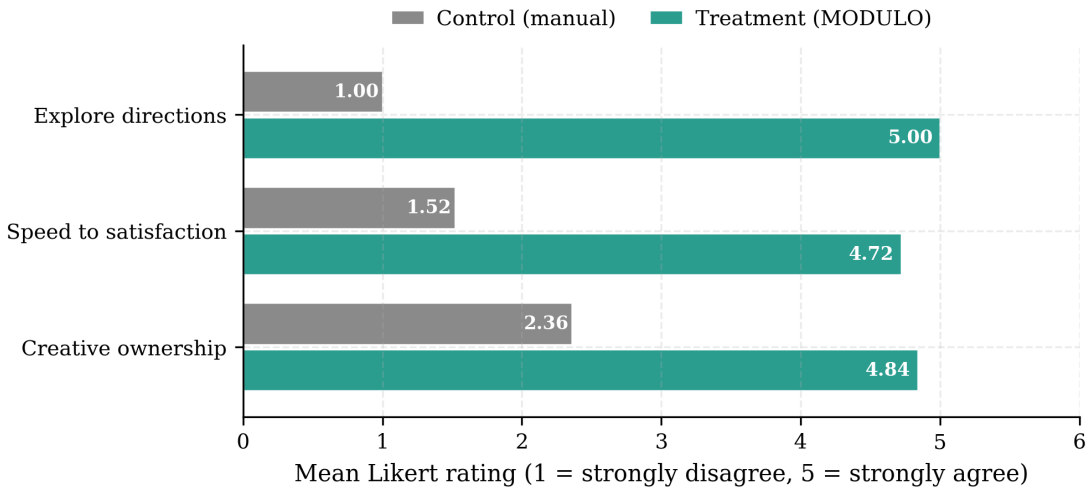


Figure 4.9: Selected subjective Likert ratings ($N = 25$, 1–5 scale). All differences are significant after Benjamini–Hochberg correction ($p_{\text{adj}} < .001$). The exploration item produced a unanimous ceiling effect of 5.00 under treatment and a unanimous floor effect of 1.00 under control.

Table 4.2: Selected subjective rating results.

Item	**Control** $\bar{x}$	**Treatment** $\bar{x}$	p_{adj}
Explore multiple directions	1.00	5.00	$<.001$
Speed to satisfaction	1.52	4.72	$<.001$
Creative ownership	2.36	4.84	$<.001$

Theme 2: Editing as the central benefit. Participants cited the interactive editing environment, including swapping chords between columns, adjusting voicings, and toggling inversions, more frequently than the generation step itself, suggesting that the selective commitment mechanism, not the generation model, is the key differentiator.

Theme 3: Exploration as enjoyment. Several participants characterized auditioning as inherently enjoyable, independent of task utility, suggesting parallel ideation tools may enhance the affective quality of the creative experience.

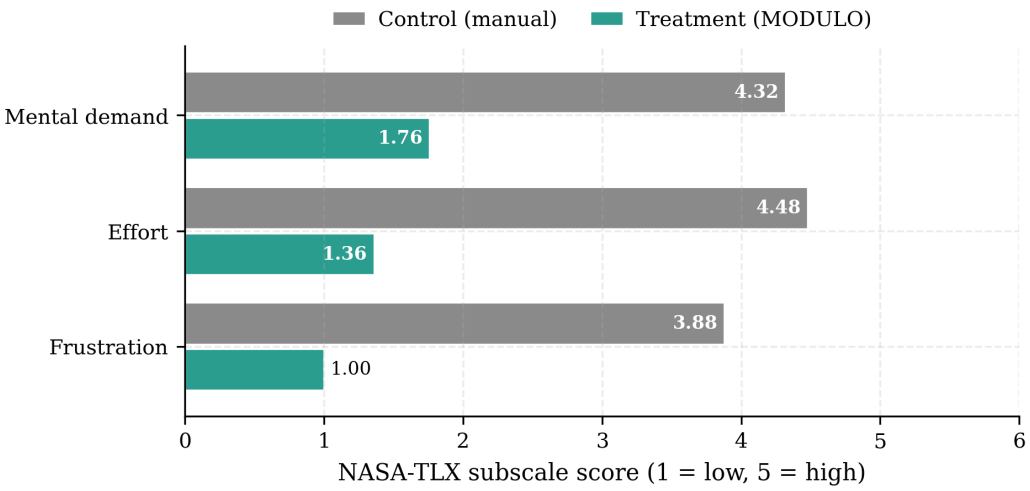


Figure 4.10: NASA-TLX subscale ratings ($N = 25$, 1–5 scale). The treatment condition produced large reductions in mental demand, effort, and frustration, with frustration reaching the absolute floor of 1.00 under treatment.

4.9.4 Per-Participant Case Studies

One participant, an experienced jazz practitioner, exploited the high-temperature columns to discover substitute dominant chords, ultimately constructing a tritone substitution chain that they described as "something I know about but wouldn't have tried without seeing it suggested." Another participant, a novice with limited harmonic knowledge, relied primarily on the low-temperature column but used the editing interface to experiment with inversions, producing a progression more sophisticated than their self-assessed ability would predict. These cases illustrate that the system supports divergent use patterns across expertise levels while yielding measurable improvements for both populations.

4.9.5 Where the Comparison Was Structurally Favorable to MODULO

Three of the metrics reported above are structurally tilted toward the treatment by the design of the comparison itself, and we acknowledge them explicitly here. *Unique chord roots* ($d_z = +3.36$) and *unique chord qualities* ($d_z = +4.30$) measure how many distinct symbols

a participant's final progression contains. The treatment exposes five parallel temperature-varied candidates per chord position; the control exposes one. A treatment participant who selectively commits across columns will, almost by construction, end up with a wider symbol vocabulary than a control participant working from a single sequential candidate at a time, regardless of whether the wider vocabulary is musically better. We report these effects because they document the size of the exploration the interface enables, but they should be read as evidence that the system *makes vocabulary expansion available*, not that the system causally produces *better* progressions on a fixed quality criterion. The post-satisfaction refinement-time metric ($d_z = +1.62$) is similarly tilted: only the treatment supports comparison and selective commitment, so any time spent on those activities accrues only to the treatment arm. The contestable metrics in this study are time-to-satisfaction (where the treatment must demonstrate that parallel exploration does not slow first-acceptable arrival) and the subjective ratings (where participants directly compare both conditions and could rate the control higher on perceived ownership or familiarity). Both contestable metrics moved in MODULO's direction, but it is the contestable metrics that carry the discriminating evidence.

CHAPTER 5

The Chord Workshop

Conventional digital audio workstations represent harmonic content at the level of individual MIDI note events: pitch, velocity, onset, and duration [81]. While this representation is complete in the sense that any chord can be encoded, it obscures the musical abstractions that musicians use when reasoning about harmony, including root, quality, inversion, and voicing [4]. This chapter presents the Chord Workshop, a direct-manipulation editor within MODULO that operates on harmonic abstractions rather than raw MIDI events, enabling musicians to compose and refine chord progressions at the level of musical intent.

Section 5.1 discusses the design rationale. Section 5.2 formalizes the chord cell representation. Section 5.3 defines the parameter space. Section 5.4 specifies the dual display mode. Section 5.5 formalizes out-of-key detection. Section 5.6 describes the preview note editing system. Section 5.7 discusses interaction design principles.

5.1 DESIGN MOTIVATION

The fundamental tension in harmonic editing is one of *abstraction level*. A MIDI piano roll provides maximal control, since every note can be placed, moved, and resized independently, but requires the musician to mentally decompose high-level harmonic intentions into individual pitch coordinates. This cognitive overhead is especially burdensome during exploratory composition [89], when the musician wishes to experiment rapidly with chord progressions without committing to specific voicings.

The Chord Workshop resolves this tension by presenting the accompaniment as a se-

quence of labeled chord cells, each parameterized by musically meaningful attributes. The musician manipulates roots, qualities, and inversions directly; the system computes the corresponding MIDI realization automatically. This design follows the principle of *semantic direct manipulation* [53]: the interface objects correspond one-to-one with the musician's mental model of harmonic structure, minimizing the gulf of execution between intention and action.

5.2 CHORD CELL REPRESENTATION

Each chord cell is a tuple

$$\mathbf{c} = (r,\, q,\, \iota,\, o,\, \delta,\, \rho), \tag{5.1}$$

where:

- $r \in \mathbb{Z}_{12} = \{0, 1, \ldots, 11\}$ is the root pitch class, with $0 \equiv C$, $1 \equiv C\sharp/D\flat$, and so on under the standard chromatic encoding;
- $q \in Q$ is the chord quality, drawn from the quality set

  $$Q = \{\text{maj},\, \text{min},\, 7,\, \text{maj7},\, \text{min7},\, \text{dim},\, \text{sus2},\, \text{sus4}\},$$

  with $|Q| = 8$;
- $\iota \in \mathbb{Z}$ is the inversion index, where $\iota = 0$ denotes root position, $\iota = 1$ first inversion, and so on;
- $o \in \mathbb{Z}$ is the octave offset relative to a default register;
- $\delta \in \mathbb{Q}_{>0}$ is the duration in beats;
- $\rho \in \{\text{block}, \text{arpeggio}, \text{doubleTime}\}$ is the playback style (as defined in Chapter 4, Section 4.5).

Figure 5.1: The Chord Workshop presents a progression as a row of labeled chord cells above the editing toolbar, which exposes root, quality, inversion, octave, duration, and playback style as first-class controls.

A chord progression of K cells is an ordered sequence $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K)$ with total duration $\sum_{k=1}^{K} \delta_k$.

5.3 THE CHORD PARAMETER SPACE

The discrete parameter space of a single chord cell, excluding duration and playback style, is

$$\mathcal{C}_{\text{param}} = \mathbb{Z}_{12} \times Q \times \mathbb{Z}_{\iota_{\max}} \times \mathbb{Z}_{o_{\max}}, \tag{5.2}$$

where $\iota_{\max}$ is the maximum inversion index (equal to the number of chord tones minus one) and $o_{\max}$ bounds the octave range. For a four-note seventh chord, $\iota_{\max} = 3$; for a

triad, $\iota_{\max} = 2$.

The MIDI realization function ϕ maps chord parameters to a set of MIDI note numbers:

$$\phi : \mathcal{C}_{\text{param}} \longrightarrow \mathcal{P}(\{0, 1, \ldots, 127\}), \tag{5.3}$$

where $\mathcal{P}$ denotes the power set. The mapping proceeds in three steps:

Step 1: Quality template. Each quality $q \in Q$ defines a template of intervals above the root, expressed as pitch-class offsets [4]. Let $\mathbf{I}_q \in \mathbb{Z}_{12}^{n_q}$ denote the interval template for quality q, where $n_q = |\mathbf{I}_q|$ is the number of chord tones. For example:

$$\mathbf{I}_{\text{maj}} = (0, 4, 7), \tag{5.4}$$

$$\mathbf{I}_{\text{min7}} = (0, 3, 7, 10), \tag{5.5}$$

$$\mathbf{I}_{\text{dim}} = (0, 3, 6). \tag{5.6}$$

Step 2: Inversion rotation. The inversion index ι cyclically rotates the template and raises rotated tones by an octave. Formally, the inverted template $\mathbf{I}_q^{(\iota)}$ is obtained by rotating $\mathbf{I}_q$ leftward by ι positions and adding 12 to each rotated element to maintain ascending pitch order:

$$I_{q,j}^{(\iota)} = \begin{cases} I_{q,\,(j+\iota) \bmod n_q} & \text{if } (j+\iota) \bmod n_q \geq \iota, \\ I_{q,\,(j+\iota) \bmod n_q} + 12 & \text{otherwise.} \end{cases} \tag{5.7}$$

Step 3: Absolute pitch computation. Each interval in the inverted template is mapped to an absolute MIDI note number:

$$\phi(r, q, \iota, o) = \left\{ r + 12\,(o + o_{\text{base}}) + I_{q,j}^{(\iota)} \;:\; j = 0, \ldots, n_q - 1 \right\}, \tag{5.8}$$

where o_{base} is a fixed base octave (typically 4, corresponding to middle C at MIDI note 60).

The interface exposes the following atomic editing operations, each of which modifies

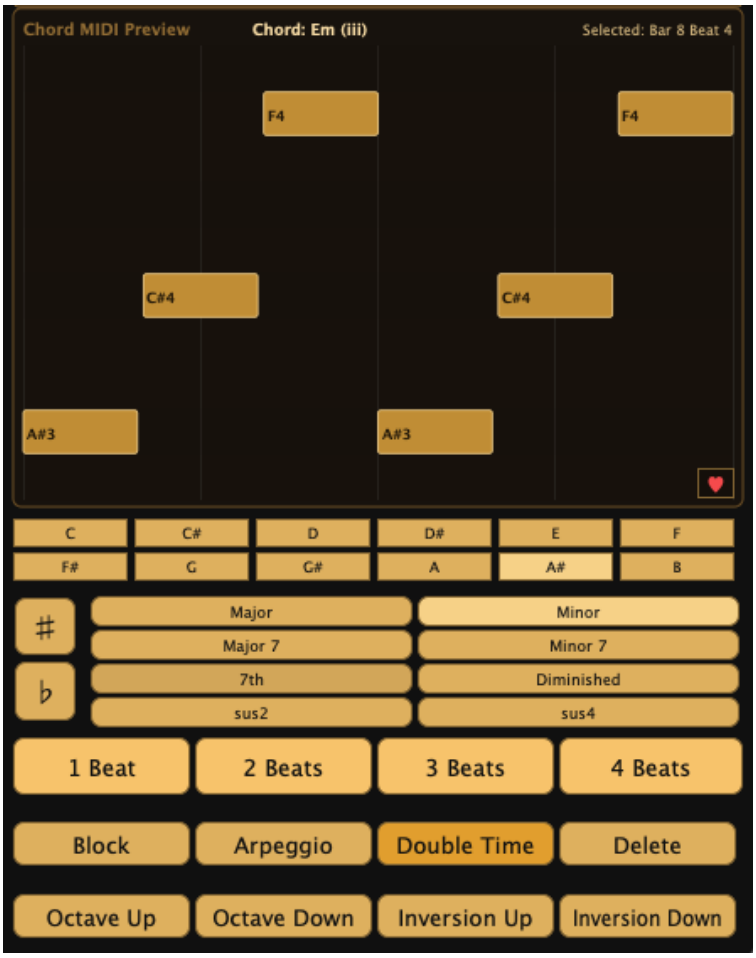


Figure 5.2: The two atomic harmonic controls in the Chord Workshop toolbar, captured in a single zoomed-in view: the twelve chromatic root selectors $r \in \mathbb{Z}_{12}$ at the top and the eight supported chord qualities in Q from Equation 5.1 at the bottom (*maj*, *min*, *7*, *maj7*, *min7*, *dim*, *sus2*, *sus4*). The currently selected root and quality are highlighted in each strip.

exactly one parameter of the selected chord cell:

$$\text{semitoneUp}(\mathbf{c}) : r \mapsto (r+1) \bmod 12, \tag{5.9}$$

$$\text{semitoneDown}(\mathbf{c}) : r \mapsto (r-1) \bmod 12, \tag{5.10}$$

$$\text{octaveUp}(\mathbf{c}) : o \mapsto o+1, \tag{5.11}$$

$$\text{octaveDown}(\mathbf{c}) : o \mapsto o-1, \tag{5.12}$$

$$\text{inversionUp}(\mathbf{c}) : \iota \mapsto (\iota+1) \bmod n_q, \tag{5.13}$$

$$\text{inversionDown}(\mathbf{c}) : \iota \mapsto (\iota-1) \bmod n_q. \tag{5.14}$$

Additional operations include deletion of a chord cell (deleteChord), insertion of a new cell at a specified temporal position, and lasso selection for batch parameter editing across multiple cells.

5.4 ROMAN NUMERAL AND ABSOLUTE NAME DISPLAY

The interface supports two display modes for chord labels. The *absolute* mode displays standard chord symbols such as "Dm7" or "G♯maj7", while the *Roman numeral* mode expresses each chord relative to the current key.

Given a key $K = (r_K, \mu)$ with tonic pitch class r_K and mode $\mu \in \{\text{major}, \text{minor}\}$, define the scale $S_K = (s_0, s_1, \ldots, s_6)$ as the ordered sequence of seven pitch classes belonging to the key. The Roman numeral mapping requires computing the scale degree of a chord's root:

$$d(r) = S_K^{-1}(r) = \big\{ j \in \{0, \ldots, 6\} : s_j \equiv r \pmod{12} \big\}, \tag{5.15}$$

where S_K^{-1} denotes the inverse lookup. When r belongs to the scale, i.e., $r \in \{s_0, \ldots, s_6\}$, the degree d is well-defined and the chord is displayed as a Roman numeral from I through VII, with case and accidentals reflecting quality. When $r \notin \{s_0, \ldots, s_6\}$, the chord is a chromatic alteration and is displayed with an accidental prefix such as ♭VII or ♯IV.

5.5 OUT-OF-KEY DETECTION AND VISUAL FEEDBACK

To provide immediate visual feedback about harmonic relationships, the Chord Workshop highlights chords whose pitch content falls outside the current key. Define the pitch-class

Figure 5.3: The same progression in C major rendered in absolute chord symbols (left) and Roman numeral notation (right); the toolbar toggle controls which scheme is displayed.

set of key K as

$$\mathrm{PC}(K) = \{s_0, s_1, \ldots, s_6\} \subset \mathbb{Z}_{12}. \tag{5.16}$$

For a chord $\mathbf{c} = (r, q, \iota, o, \delta, \rho)$, the set of pitch classes present in the chord is

$$\mathrm{PC}(\mathbf{c}) = \big\{(r + I_{q,j}) \bmod 12 : j = 0, \ldots, n_q - 1\big\}. \tag{5.17}$$

The out-of-key predicate is then

$$\mathrm{OutOfKey}(\mathbf{c}, K) = \begin{cases} \text{true} & \text{if } \mathrm{PC}(\mathbf{c}) \not\subseteq \mathrm{PC}(K), \\ \text{false} & \text{otherwise.} \end{cases} \tag{5.18}$$

When $\mathrm{OutOfKey}(\mathbf{c}, K) = \text{true}$, the corresponding chord cell is rendered with a red background, providing an immediate visual signal that the chord contains chromatic tones relative to the current key. This does not prevent the musician from using out-of-key chords,

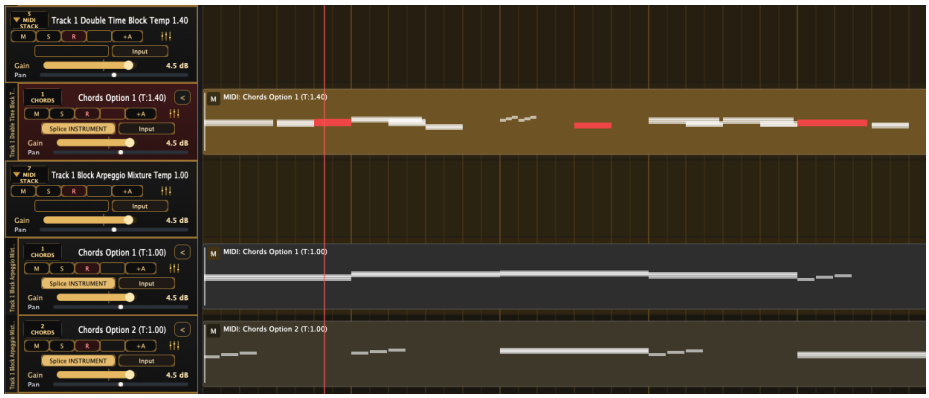


Figure 5.4: Out-of-key highlighting on a MIDI track in the timeline: the chord cells whose pitch-class set is not a subset of $\mathrm{PC}(K)$ are flagged in red directly on the clip's preview lane, so a glance at the track strip immediately reveals which beats contain chromatic content relative to the current key. Diatonic cells retain the default fill.

since chromatic harmony is an essential compositional device [78], but ensures that such choices are made deliberately.

The set of non-diatonic pitch classes within the chord, which can be highlighted individually within the preview note grid, is

$$\mathrm{Chromatic}(\mathbf{c}, K) = \mathrm{PC}(\mathbf{c}) \setminus \mathrm{PC}(K). \tag{5.19}$$

5.6 PREVIEW NOTE EDITING

While the chord-level abstraction handles most harmonic editing, fine-grained control over individual notes is sometimes required, for example, to add a passing tone, adjust the spacing of a voicing, or modify the velocity of a single note. The Chord Workshop provides a preview note grid that expands a selected chord cell into its constituent MIDI notes.

The preview grid displays each note in $\phi(r, q, \iota, o)$ as an editable cell with pitch, velocity $v \in \{1, \ldots, 127\}$, onset, and duration. Edits at this level create a *voicing override* that

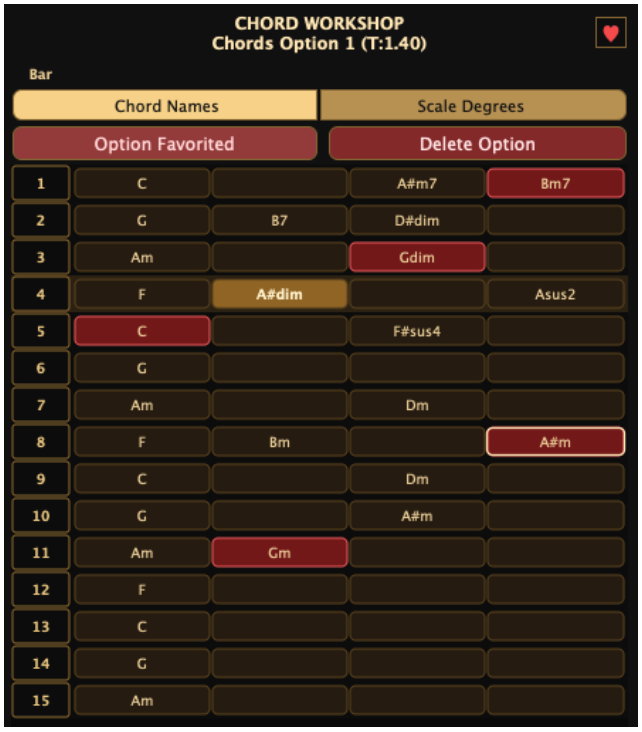


Figure 5.5: Zoomed view of the Chord Workshop chord-grid layout: fifteen measures stacked vertically, each row a single 4/4 bar with four beat cells running horizontally. Cells outlined in red mark chords the musician favorited during the audition stage, keeping preferred options visible while discarded alternatives are pruned.

detaches the affected notes from the chord-level parameterization:

$$\phi_{\text{eff}}(\mathbf{c}) = \big(\phi(r, q, \iota, o) \setminus D\big) \cup A, \tag{5.20}$$

where D is the set of deleted or modified default notes and A is the set of added or repositioned notes. The override is stored alongside the chord cell and is re-applied whenever the cell is re-rendered, but is discarded if the chord's root or quality is subsequently changed, since the template $\mathbf{I}_q$ would no longer correspond to the overridden voicing.

Snap-to-grid quantizes note placements to a configurable subdivision (typically sixteenth notes), ensuring temporal alignment with the surrounding musical context.

5.7 INTERACTION DESIGN PRINCIPLES

The design of the Chord Workshop is guided by three principles drawn from the interaction design and creativity support literature [53, 89]:

Semantic directness. Interface primitives correspond to musical concepts such as root, quality, and inversion rather than to low-level MIDI parameters. This reduces the cognitive distance between the musician's harmonic intent and the actions required to realize it.

Progressive disclosure. The most common operations, root and quality selection, are immediately accessible, while less frequent operations such as inversion cycling, velocity editing, and voicing overrides are available through secondary panels. This layered structure accommodates both rapid exploration and detailed refinement without cluttering the primary interface.

Reversibility. All operations are undoable, and parameter changes are applied non-destructively. The musician can freely explore the parameter space $\mathcal{C}_{\text{param}}$ without risk of irreversible modification, supporting the iterative and exploratory nature of the composition process.

Together with the chord generation system described in Chapter 4, the Chord Workshop completes the harmonic co-creation pipeline: generated chords enter the workshop as editable cells, and the musician refines them using the semantic editing operations formalized in this chapter.

CHAPTER 6

Rule-Based Harmony Engine

The preceding two chapters addressed harmony at the level of chord progressions, i.e., sequences of discrete harmonic symbols. This chapter turns to a complementary problem: generating contrapuntal harmony lines that accompany a given melody note-by-note. The rule-based harmony engine in MODULO produces up to five simultaneous harmony voices using classical voice-leading principles encoded as a weighted scoring function, executed by a linear-time greedy algorithm.

Section 6.1 situates the problem within computational music theory. Section 6.2 formalizes the scale and interval arithmetic. Section 6.3 defines the five interval strategies. Section 6.4 derives the adaptive scoring function. Section 6.5 discusses the consonance hierarchy and its perceptual basis. Section 6.6 specifies the parallel motion constraints. Section 6.7 analyzes the algorithm's complexity. Section 6.8 presents a within-subjects experiment evaluating the engine as a compositional tool.

6.1 MOTIVATION: VOICE LEADING IN COMPUTATIONAL MUSIC THEORY

Voice leading, the practice of moving individual voices smoothly from chord to chord, is a foundational concern in Western tonal music. The rules governing voice leading, codified in treatises from Fux [40] to modern harmony textbooks [4, 78], encode perceptual preferences for stepwise motion, consonant intervals, and contrary motion between voices.

Computational implementations of these rules have a long history [32, 52], but most systems operate offline and produce a single deterministic output.

The harmony engine in MODULO is designed for real-time compositional assistance: given a melody, the musician selects one or more interval strategies, or enables the adaptive strategy, and the system generates harmony lines that can be auditioned immediately alongside the melody. The five available strategies, diatonic third above, sixth below, perfect fifth above, contrary motion, and adaptive, represent a graduated spectrum from simple parallel intervals to context-sensitive voice leading.

6.2 SCALE AND INTERVAL ARITHMETIC

All harmony computations operate within a diatonic scale context. Let the scale S be an ordered set of seven pitch classes:

$$S = (s_0, s_1, s_2, s_3, s_4, s_5, s_6), \quad s_j \in \mathbb{Z}_{12}, \tag{6.1}$$

where the two supported scale types are the major scale,

$$S_{\text{maj}}(r) = \{r,\ r{+}2,\ r{+}4,\ r{+}5,\ r{+}7,\ r{+}9,\ r{+}11\} \pmod{12}, \tag{6.2}$$

and the natural minor scale,

$$S_{\text{min}}(r) = \{r,\ r{+}2,\ r{+}3,\ r{+}5,\ r{+}7,\ r{+}8,\ r{+}10\} \pmod{12}, \tag{6.3}$$

where $r \in \mathbb{Z}_{12}$ is the tonic.

Definition 2 (Scale-degree function)**.** *For a pitch class $p \in S$, the scale-degree function*

returns its index within the scale:

$$\deg_S(p) = j \quad \textit{such that } s_j \equiv p \pmod{12}. \tag{6.4}$$

Definition 3 (Diatonic transposition)**.** *The diatonic transposition of a melody note at scale degree d by k scale steps is*

$$\mathcal{T}_S(d, k) = s_{(d+k) \bmod 7}, \tag{6.5}$$

which moves k steps along the scale, wrapping at the octave boundary. The resulting interval in semitones is $\mathcal{T}_S(d, k) - s_d \pmod{12}$, which varies depending on the position within the scale: for example, a diatonic third above the first degree of a major scale spans four semitones, while a diatonic third above the third degree spans three.

Given a melody note with MIDI pitch m and scale degree $d = \deg_S(m \bmod 12)$, the octave of the melody note is $\omega_m = \lfloor m/12 \rfloor$. The harmony note at diatonic offset k is computed as

$$h(m, k) = 12\,\omega_m + \mathcal{T}_S(d, k) + 12\,\Delta\omega(d, k), \tag{6.6}$$

where $\Delta\omega(d, k)$ is an octave correction term that ensures the harmony note remains in the correct register:

$$\Delta\omega(d, k) = \begin{cases} +1 & \text{if } (d + k) \bmod 7 < d \text{ and } k > 0, \\ -1 & \text{if } (d + k) \bmod 7 > d \text{ and } k < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.7}$$

6.3 THE FIVE INTERVAL STRATEGIES

The harmony engine implements five strategies for selecting harmony notes. Each strategy defines a function $\mathcal{H}_i : \mathbb{Z}_{128} \times S \to \mathbb{Z}_{128}$ that maps a melody MIDI note and scale context

to a harmony MIDI note.

Strategy 1: Diatonic third above. The harmony note is two diatonic scale steps above the melody:

$$\mathcal{H}_1(m) = h(m, +2). \tag{6.8}$$

This produces parallel thirds, the most common harmonization interval in popular and folk music.

Strategy 2: Sixth below. The harmony note is two diatonic scale steps below the melody:

$$\mathcal{H}_2(m) = h(m, -2). \tag{6.9}$$

By the principle of interval inversion, a diatonic sixth below is equivalent to a diatonic third above in the lower octave, but the resulting voice sits beneath the melody, producing a distinct textural effect.

Strategy 3: Perfect fifth above. The harmony note is four diatonic scale steps above the melody:

$$\mathcal{H}_3(m) = h(m, +4). \tag{6.10}$$

This produces predominantly perfect fifths of seven semitones, though the diatonic constraint yields a diminished fifth at one scale position, for example, between the seventh and fourth degrees of the major scale.

Strategy 4: Contrary motion. The harmony alternates between a third above and a sixth below based on the direction of melodic motion. Let $\Delta m_t = m_t - m_{t-1}$ denote the melodic

interval at frame t. The strategy is

$$\mathcal{H}_4(m_t) = \begin{cases} h(m_t, -2) & \text{if } \Delta m_t > 0 \quad \text{(melody ascending)}, \\ h(m_t, +2) & \text{if } \Delta m_t < 0 \quad \text{(melody descending)}, \\ h(m_t, +2) & \text{if } \Delta m_t = 0 \quad \text{(melody stationary)}. \end{cases} \tag{6.11}$$

This strategy produces contrary motion, where the harmony voice moves in the opposite direction from the melody; contrary motion is the most prized voice-leading technique in classical counterpoint [40, 52].

Strategy 5: Adaptive. The adaptive strategy selects between a third above and a sixth below at each frame using a scoring function that balances consonance, smoothness, and contrary motion. This is the most sophisticated strategy and is formalized in the following section.

6.4 THE ADAPTIVE SCORING FUNCTION

The adaptive strategy evaluates candidate harmony notes using a linear scoring function that aggregates three weighted criteria. At frame t, given the melody note m_t, the previous harmony note h_{t-1}, and the previous melody note m_{t-1}, the candidate set is $\{h(m_t, +2),\ h(m_t, -2)\}$. For each candidate h, the score is

$$\begin{aligned} \mathcal{S}(h \mid m_t, h_{t-1}, m_{t-1}) = w_{\text{int}} \cdot I(h, m_t) \\ + w_{\text{sm}} \cdot |h - h_{t-1}| \\ + w_{\text{ctr}} \cdot \mathbf{1}[\text{contrary}(h, m_t, m_{t-1})], \end{aligned} \tag{6.12}$$

where the three terms and their weights are defined as follows.

6.4.1 Interval Consonance Term

The function $I(h, m_t)$ assigns a consonance reward based on the interval class between the harmony and melody notes:

$$I(h,m) = \begin{cases} +3.2 & \text{if } |h-m| \bmod 12 \in \{3,4\} \quad \text{(minor/major third)}, \\ +3.0 & \text{if } |h-m| \bmod 12 \in \{8,9\} \quad \text{(minor/major sixth)}, \\ +1.3 & \text{if } |h-m| \bmod 12 \in \{7\} \quad \text{(perfect fifth)}, \\ -1.8 & \text{otherwise (dissonant intervals).} \end{cases} \tag{6.13}$$

The consonance hierarchy encoded here, with thirds most preferred, sixths close behind, fifths moderately rewarded, and all else penalized, reflects well-established findings in music perception research [52, 4]. The reference weight $w_{\text{int}} = 1.0$ assigns this term unit scaling.

6.4.2 Smoothness Penalty

The term $|h - h_{t-1}|$ measures the size of the leap in the harmony voice, in semitones. The weight $w_{\text{sm}} = -0.10$ penalizes large leaps, encoding the preference for stepwise motion:

$$w_{\text{sm}} \cdot |h - h_{t-1}| \leq 0, \quad \text{with equality iff } h = h_{t-1}. \tag{6.14}$$

This penalty is deliberately moderate: it discourages gratuitous leaps but does not override a strong consonance preference, allowing the algorithm to accept a larger leap when it yields a substantially more consonant interval.

6.4.3 Contrary Motion Reward

The indicator function $\mathbf{1}[\text{contrary}(\cdot)]$ equals 1 when the harmony voice moves in the opposite direction from the melody:

$$\mathbf{1}[\text{contrary}(h, m_t, m_{t-1})] = \begin{cases} 1 & \text{if } \operatorname{sgn}(h - h_{t-1}) \neq \operatorname{sgn}(m_t - m_{t-1}) \text{ and both } \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.15}$$

The weight $w_{\text{ctr}} = +1.1$ assigns a substantial bonus to contrary motion, reflecting its privileged status in voice-leading theory.

6.4.4 Weight Calibration

The weight vector $(w_{\text{int}}, w_{\text{sm}}, w_{\text{ctr}}) = (1.0, -0.10, +1.1)$ was calibrated by the first author through iterative listening tests across a diverse set of melodies spanning multiple keys, modes, and tempi. The calibration procedure involved systematically varying each weight while holding the others fixed, evaluating the resulting harmonizations by ear, and selecting the configuration that most consistently produced musically satisfactory voice leading. While this manual calibration lacks the rigor of a formal optimization procedure, it reflects domain expertise and was verified against known voice-leading exemplars from the common-practice repertoire.

6.5 CONSONANCE HIERARCHY AND PERCEPTUAL BASIS

The consonance values in Equation 6.13 are grounded in the psychoacoustic literature on interval perception. Prior work has demonstrated that the perceptual consonance of intervals correlates with their frequency of occurrence in tonal music, supporting the idea that

Table 6.1: Consonance hierarchy used by the adaptive scoring function. Interval classes are measured in semitones modulo 12.

Interval Class	**Semitones (mod 12)**	**Score** $I(h, m)$
Minor third	3	$+3.2$
Major third	4	$+3.2$
Minor sixth	8	$+3.0$
Major sixth	9	$+3.0$
Perfect fifth	7	$+1.3$
All other intervals	—	-1.8

compositional conventions encode implicit perceptual knowledge [52].

Table 6.1 summarizes the consonance hierarchy as implemented in the scoring function.

The ranking places thirds above sixths despite their theoretical equivalence under interval inversion. This asymmetry reflects the practical observation that thirds, being narrower, produce a tighter blend between melody and harmony voices and are perceptually more fused in the register where most melodies are written.

6.6 PARALLEL MOTION CONSTRAINTS

Classical voice-leading theory prohibits parallel perfect fifths and parallel perfect octaves, or unisons, between any pair of voices, as these intervals reduce the perceived independence of the voices [40, 78]. The scoring function enforces this constraint through a penalty term applied when consecutive frames produce the same perfect interval:

$$P_{\parallel}(h_t, h_{t-1}, m_t, m_{t-1}) = \begin{cases} -3.0 & \text{if } (h_t - m_t) \bmod 12 = (h_{t-1} - m_{t-1}) \bmod 12 \\ & \quad \text{and } (h_t - m_t) \bmod 12 \in \{0, 7\}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.16}$$

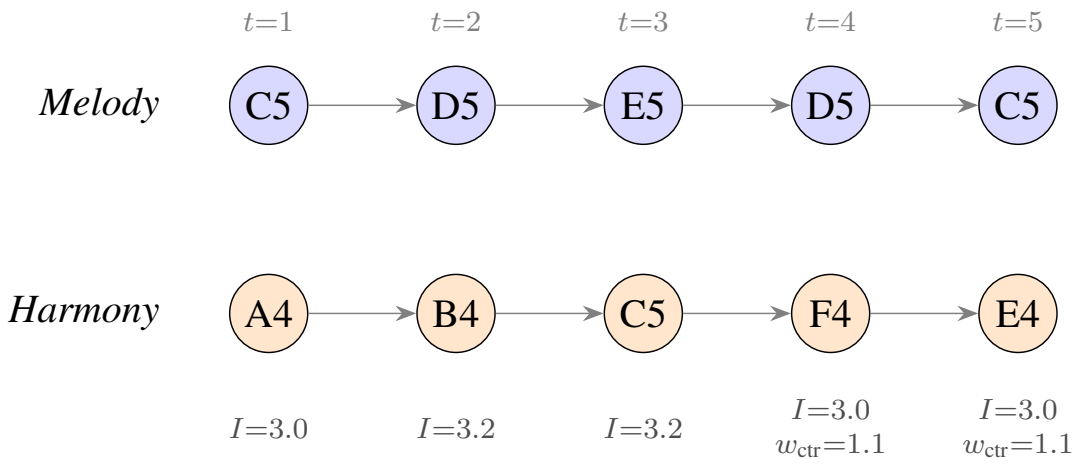


Figure 6.1: Scoring visualization for the adaptive strategy. Melody (blue) and harmony (orange) frames are labelled with the per-frame consonance score I; frames where the harmony moves opposite to the melody earn the contrary motion bonus w_{ctr}.

The full adaptive score including the parallel motion penalty is therefore

$$\mathcal{S}_{\text{full}}(h_t) = \mathcal{S}(h_t \mid m_t, h_{t-1}, m_{t-1}) + P_{\parallel}(h_t, h_{t-1}, m_t, m_{t-1}). \tag{6.17}$$

The penalty magnitude of -3.0 is calibrated to be large enough to override the consonance reward for fifths ($+1.3$), ensuring that the algorithm avoids parallel fifths even when the fifth is the more consonant interval in isolation.

6.7 ALGORITHM COMPLEXITY AND GREEDY SELECTION

The adaptive strategy is implemented as a greedy left-to-right sweep. At each frame t, the algorithm evaluates the two candidates $\{h(m_t, +2),\ h(m_t, -2)\}$ and selects the one with the higher full score (Equation 6.17).

Time complexity. Each frame requires a constant number of operations: two diatonic transpositions (each $O(1)$ via lookup in the seven-element scale), two score evaluations

Algorithm 1 Greedy Adaptive Harmony Generation

Require: Melody sequence $\mathbf{m} = (m_1, \ldots, m_T)$, scale S, weights $(w_{\text{int}}, w_{\text{sm}}, w_{\text{ctr}})$
Ensure: Harmony sequence $\mathbf{h} = (h_1, \ldots, h_T)$

$h_0 \leftarrow h(m_1, +2)$ ▷ Initialize with third above first note
for $t = 1$ **to** T **do**
  $h^+ \leftarrow h(m_t, +2)$ ▷ Third above candidate
  $h^- \leftarrow h(m_t, -2)$ ▷ Sixth below candidate
  $s^+ \leftarrow \mathcal{S}_{\text{full}}(h^+ \mid m_t, h_{t-1}, m_{t-1})$
  $s^- \leftarrow \mathcal{S}_{\text{full}}(h^- \mid m_t, h_{t-1}, m_{t-1})$
  if $s^+ \geq s^-$ **then**
    $h_t \leftarrow h^+$
  else
    $h_t \leftarrow h^-$
  end if
end for
return $\mathbf{h}$

(each $O(1)$ given the previous frame's state), and one comparison. The total running time is therefore

$$T_{\text{algo}} = O(T), \tag{6.18}$$

where T is the number of melody frames. This linear complexity ensures that harmony generation is effectively instantaneous for typical musical inputs (hundreds to low thousands of frames).

Optimality considerations. The greedy algorithm is not globally optimal in the sense of maximizing $\sum_{t=1}^{T} \mathcal{S}_{\text{full}}(h_t)$; a dynamic programming formulation could achieve this in $O(2T)$ time, which remains linear. However, the greedy approach was preferred for two reasons: (i) the scoring function is designed so that locally optimal choices produce globally satisfactory voice leading, and (ii) the greedy structure enables a streaming implementation in which harmony notes are generated and rendered as the melody is played, without requiring the full melody to be known in advance.

For the non-adaptive Strategies 1–4, the mapping is purely pointwise: each melody note maps to exactly one harmony note independent of context, yielding $O(T)$ time with

no state dependence.

6.8 EVALUATION: WITHIN-SUBJECTS HARMONY EXPERIMENT

We evaluate the harmony engine with a within-subjects experiment paralleling the chord co-creation study of Chapter 4: participants compose multi-voice harmonizations under a manual control and an AI-assisted treatment, evaluated by automated metrics, standardized self-report instruments, and qualitative analysis.

6.8.1 Experimental Design

Design. Two conditions, counterbalanced with the same hash-based randomization employed in the chord study:

$$\text{Order}(p) = \begin{cases} (\text{control}, \text{treatment}) & \text{if } \operatorname{hash}(p) \bmod 2 = 0, \\ (\text{treatment}, \text{control}) & \text{otherwise}, \end{cases} \tag{6.19}$$

where p is the participant identifier.

Participants. A target sample of $N = 15$–20 participants, all with at least two years of formal music training in instrumental performance, voice, or composition, verified through a pre-study screening questionnaire. Participants are required to have basic familiarity with MIDI piano-roll editing in any digital audio workstation.

Stimuli. Two melodies drawn from the same stimulus set used in the chord co-creation study:

- **Melody A:** C major, 4/4 meter, 80 BPM, diatonic with stepwise motion.

- **Melody B:** F minor, 4/4 meter, 70 BPM, with chromatic inflections and wider intervals.

Melody A is presented in the first condition and Melody B in the second, ensuring no melody–condition confound.

Conditions.

Control (Manual Harmony): The participant is presented with the melody on a read-only track and three empty harmony tracks. Only the piano-roll editor is available for note entry. The harmony generation engine, chord generation, chord workshop, and all AI-assisted tools are disabled. The participant must manually compose three distinct harmony lines.

Treatment (AI-Assisted Harmony): The same track layout, but the participant has access to the harmony generation engine with all five strategies: diatonic third above, sixth below, perfect fifth above, contrary motion, and adaptive. Generated harmony lines may be further edited in the piano roll. The participant must produce three harmony lines using any combination of generation and manual editing.

6.8.2 Task Flow

Each condition follows a structured task lifecycle mirroring the chord study protocol:

1. **Prepare:** The system loads the melody onto a read-only track and creates three empty harmony tracks with virtual instruments pre-loaded.

2. **Instructions:** Condition-specific instructions are displayed, including a treatment-condition tools briefing for the harmony generation strategies.

3. **Begin:** The participant initiates the task; all timers and event logging activate.

Figure 6.2: Harmony generation study workspace, treatment condition: the harmony engine dialog is open with all five interval strategies (third above, sixth below, fifth above, contrary motion, and adaptive) selected, ready to spawn the corresponding harmony tracks against the melody. The control condition uses the same workspace minus the dialog and is therefore not shown separately.

4. **Work:** The participant composes/generates three harmony lines.

5. **First Satisfaction:** The participant marks the moment they first reach a result they would be willing to keep; this is a one-time, non-terminal marker.

6. **Submit:** The participant finalizes the task; all artifacts are exported.

7. **Survey:** Post-condition surveys are administered.

6.8.3 Dependent Variables

The study measures three categories of dependent variables, paralleling the chord study's measurement framework.

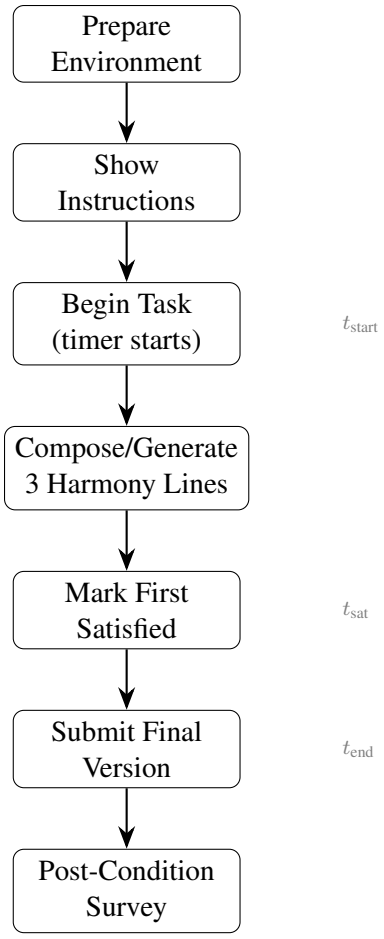


Figure 6.3: Task lifecycle for the harmony generation study. Three timestamps are recorded: task start (t_{start}), first satisfaction (t_{sat}), and task end (t_{end}).

Workflow Metrics (Automated Event Telemetry)

The study runner captures over 50 event types with millisecond-resolution timestamps:

- **Total task time:** $T_{\text{task}} = t_{\text{end}} - t_{\text{start}}$
- **Time to first satisfaction:** $T_{\text{sat}} = t_{\text{sat}} - t_{\text{start}}$
- **Playback count:** Number of playback events $|P|$
- **Generation count:** Number of harmony generation invocations $|G|$, treatment only.
- **Strategy distribution:** Frequency of each strategy selection, treatment only.
- **Note edit count:** Piano-roll note additions, deletions, and modifications $|E|$

- **Undo/redo count:** Revision events $|U|$

Musical Quality Metrics (Computed from Exported MIDI)

For each participant's three harmony lines, the following metrics are computed from the exported MIDI artifacts:

Voice-leading smoothness. The mean absolute interval between consecutive notes in harmony line i:

$$\text{VLS}_i = \frac{1}{T_i - 1} \sum_{t=2}^{T_i} |h_t^{(i)} - h_{t-1}^{(i)}|, \tag{6.20}$$

where $h_t^{(i)}$ is the MIDI pitch of the t-th note in harmony line i and T_i is the note count. Lower values indicate smoother voice leading.

Parallel perfects count. The number of consecutive frame pairs in which the melody–harmony interval is a perfect fifth or unison/octave in both frames:

$$\text{PP} = \sum_{t=2}^{T} \mathbf{1}[(h_t - m_t) \bmod 12 = (h_{t-1} - m_{t-1}) \bmod 12 \ \wedge \ (h_t - m_t) \bmod 12 \in \{0, 7\}]\,. \tag{6.21}$$

Note-in-scale ratio. The proportion of harmony notes whose pitch class falls within the diatonic scale S:

$$\text{NIS} = \frac{1}{N_h} \sum_{n=1}^{N_h} \mathbf{1}[h_n \bmod 12 \in S], \tag{6.22}$$

where N_h is the total note count across all three harmony lines.

Intervallic variety. The Shannon entropy of the interval-class distribution:

$$H_{\text{int}} = -\sum_{c=0}^{11} p_c \ln p_c, \tag{6.23}$$

where p_c is the relative frequency of interval class c among consecutive notes.

Consonance ratio. The proportion of simultaneous melody–harmony intervals that are consonant (thirds, sixths, or fifths):

$$\text{CR} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}[(h_t - m_t) \bmod 12 \in \{3, 4, 7, 8, 9\}]\,. \tag{6.24}$$

Voice independence. Measured as the normalized pairwise edit distance between harmony lines, ensuring that the three voices are not merely duplicates:

$$\text{VI} = \frac{1}{\binom{3}{2}} \sum_{i<j} \frac{d_{\text{edit}}(\mathbf{h}^{(i)}, \mathbf{h}^{(j)})}{\max(|\mathbf{h}^{(i)}|, |\mathbf{h}^{(j)}|)}, \tag{6.25}$$

where d_{edit} is the Levenshtein distance on the pitch-class sequence.

Self-Report Instruments

Identical to the chord study (Chapter 4):

- **NASA-TLX (raw):** [43] Six dimensions (mental demand, physical demand, temporal demand, performance, effort, frustration), each rated on a 0–100 scale.
- **UES-SF:** [76] Twelve items across four subscales (focused attention, perceived usability, aesthetic appeal, reward), each on a 5-point Likert scale.
- **Custom harmony items (5-point Likert):**
  (i) "I could explore multiple harmonic voicings easily."
  (ii) "I could create distinct harmony lines quickly."
  (iii) "The interface helped me achieve good voice leading."
  (iv) "I felt in control of the harmonic result."
  (v) "I would use this workflow in my own music-making."

- **Open responses:** Three prompts per condition (most helpful, most frustrating, approach description) plus a final comparison survey with six forced-choice questions and one open justification.

6.8.4 Statistical Analysis

The analysis framework mirrors the chord co-creation study for direct comparability.

Primary analyses. All within-subjects comparisons use paired t-tests on the participant-level means. For each metric Y, we test:

$$H_0 : \mu_Y^{\text{treatment}} = \mu_Y^{\text{control}} \quad \text{vs.} \quad H_1 : \mu_Y^{\text{treatment}} \neq \mu_Y^{\text{control}}. \tag{6.26}$$

Effect sizes. Cohen's d_z [27] for paired designs:

$$d_z = \frac{\bar{D}}{s_D}, \quad \text{where } D_i = Y_i^{\text{treatment}} - Y_i^{\text{control}}. \tag{6.27}$$

Multiple comparisons. The Benjamini–Hochberg procedure [13] at FDR $= 0.05$ controls the false discovery rate across the family of paired tests.

6.8.5 Hypotheses

Based on the design of the scoring function (Section 6.4) and the results of the chord co-creation study (Chapter 4), we hypothesize:

H1: Treatment reduces total task time ($T_{\text{task}}^{\text{treatment}} < T_{\text{task}}^{\text{control}}$).

H2: Treatment reduces time to first satisfaction ($T_{\text{sat}}^{\text{treatment}} < T_{\text{sat}}^{\text{control}}$).

H3: Treatment reduces NASA-TLX mental demand.

H4: Treatment increases UES-SF engagement subscale scores.

H5: Treatment yields higher consonance ratios and fewer parallel perfects than manual composition, reflecting the scoring function's built-in voice-leading constraints.

6.8.6 Power Analysis

For a paired t-test with $\alpha = 0.05$ and target power $1 - \beta = 0.80$:

- A medium–large effect $d_z = 0.70$, consistent with the chord study's observed effects, requires $N \approx 19$.
- A medium effect $d_z = 0.50$ requires $N \approx 34$.

The target range of $N = 15$–20 is calibrated to detect medium–large effects.

6.8.7 Results

Twenty-five participants ($N = 25$) completed both conditions of the harmony generation study. All participants had at least two years of music training; five were classified as beginners and twenty as intermediate or advanced. The following tables report the observed results across all dependent variables, with Benjamini–Hochberg FDR correction applied across the full family of paired t-tests ($\alpha = 0.05$).

Workflow and Musical Quality Metrics

On average, writing a single harmonisation took 625 s under control and 119 s under treatment, an 81.0% per-melody reduction ($d_z = 2.04$). Time to first satisfaction fell from 250 s to 28 s, an 88.9% reduction ($d_z = 2.73$); both differences are significant at $p < .0001$.

Participants needed fewer playback checks ($d_z = 3.10$) and manual edits ($d_z = 2.93$) in treatment. The consonance ratio was near-perfect in treatment at $M = 0.99$ compared to $M = 0.65$ in control, a $+52\%$ absolute improvement; we do not report a Cohen's d_z for this contrast because the harmony engine is deterministic for a given melody and strategy

Table 6.2: Harmony generation study: workflow and musical quality metrics ($N = 25$, mean $\pm$ SD). Each participant completed three melodies per condition; *workflow metrics above the rule are reported per melody*, dividing per-participant session totals by the three melodies, so that the numbers reflect the effort of producing a single harmonisation. *Musical quality metrics below the rule are ratios and counts computed across each participant's full submission*, and are intrinsic to the harmony that was produced rather than to any one melody. Paired $t(24)$ and d_z are invariant to the per-melody rescaling and are reported on the same paired differences as in the prior aggregate analysis. All starred metrics are significant after BH-FDR correction at $\alpha = 0.05$.

Metric	**Control**	**Treatment**	$t(24)$	d_z
Per-melody workflow (session totals ÷ 3 melodies)				
Total task time per melody (s)	625.14 ± 258.91	119.03 ± 23.32	10.18	2.04^*
Time to first satisfaction (s)	249.87 ± 77.52	27.63 ± 7.86	13.64	2.73^*
Playback count per melody	13.72 ± 2.30	5.19 ± 0.99	15.52	3.10^*
Note edit count per melody	14.81 ± 2.57	6.60 ± 1.25	14.65	2.93^*
Melody copy count per melody	0.77 ± 0.42	0.00 ± 0.00	9.29	1.86^*
Musical quality of submitted harmony (across full submission)				
Voice-leading smoothness†	4.69 ± 0.45	5.87 ± 0.61	-15.41	-3.08^*
Parallel perfects per melody	0.91 ± 0.95	12.47 ± 2.08	-23.61	—
Note-in-scale ratio	0.61 ± 0.07	0.60 ± 0.07	0.99	0.20
Intervallic variety	4.20 ± 0.17	4.39 ± 0.19	-8.54	-1.71^*
Consonance ratio	0.65 ± 0.08	0.99 ± 0.02	-20.20	—
Voice independence†	0.95 ± 0.03	0.69 ± 0.02	30.06	—

†Lower values indicate smoother voice leading (VLS) or less independence (VI). "Time to first satisfaction" is a per-melody quantity in the raw telemetry, not a session total, and is not rescaled here. Cohen's d_z is omitted ($---$ in the column) for consonance ratio, voice independence, and parallel perfects: on these three metrics the engine's deterministic structure is the dominant driver of the difference, so the standardised effect-size denominator is not interpretable; the raw means and the relative changes given in the running text are the appropriate comparison. $*$ denotes $p < .05$ post-BH.

and the treatment-arm variance ($\sigma = 0.02$) is too small to make the effect-size denominator meaningful.

Two negative results: voice independence and parallel perfects. Two of the musical-quality metrics moved *against* MODULO and we report them honestly here as negative results that the current engine does not resolve.

First, voice independence was substantially *lower* in treatment ($M = 0.69$) than in control ($M = 0.95$), a -27% relative decrease. As with the consonance ratio, the treatment-arm variance is near zero ($\sigma = 0.02$) and we do not report a Cohen's d_z, but the direc-

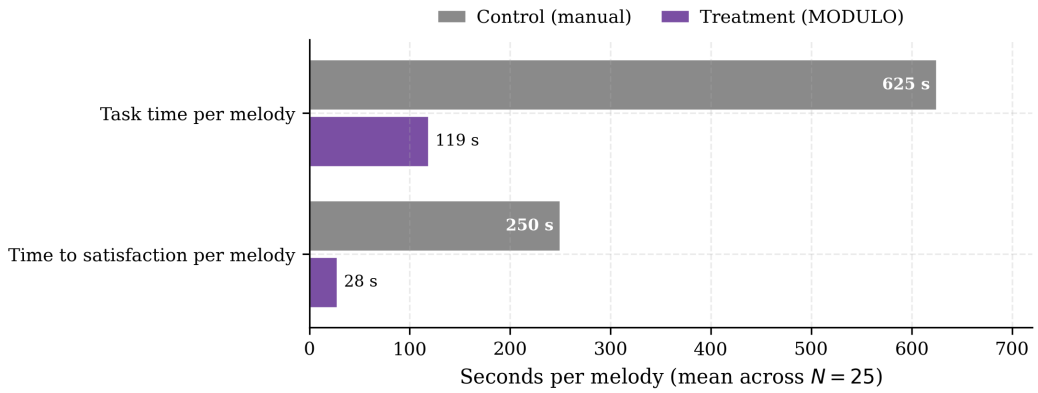


Figure 6.4: Harmony generation study: per-melody task time comparison ($N = 25$; each participant produced three harmonisations per condition). Total task time is reported *per melody* (session total divided by the three melodies). Treatment reduced per-melody task time by 81% ($d_z = 2.04$, from $625\,\text{s}$ to $119\,\text{s}$) and time to first satisfaction by 89% ($d_z = 2.73$, from $250\,\text{s}$ to $28\,\text{s}$). The control mean reflects the combined burden of recording, editing, and iterating on a single manual harmony line.

tion of the finding is unambiguous. The fixed-interval strategies (third above, sixth below, fifth above) produce voices that move in obligate parallel or near-parallel motion with the melody by construction: when the melody ascends a diatonic step, the harmony line ascends the same step at a fixed interval offset. The adaptive strategy mitigates this to a degree by switching intervals contextually, but even it does not produce true *counterpoint*—voices with rhythmically and melodically independent contours. Participants in the control condition, composing freely by ear, naturally introduced contrary motion, rhythmic displacement, and motivic independence, all of which raise the voice-independence metric.

Second, the parallel-perfects count rose from 0.91 ± 0.95 in control to 12.47 ± 2.08 in treatment, a $13.7\times$ increase ($t(24) = -23.61$, $p < .0001$). This is the same structural problem as voice independence, viewed through a different metric: a fifth-above strategy applied to a stepwise melodic line will produce one parallel fifth per step by definition; a third-above strategy applied to a melody ascending through a diatonic scale will produce a parallel third on every melodic step. Treatises from Aldwell–Schachter–Cadwallader [4] to Piston [78] treat parallel perfect fifths and octaves as the principal voice-leading error to avoid in tonal harmony, and the engine, as currently scored, does not penalise them.

Table 6.3: Harmony generation study: NASA-TLX subscale scores ($N = 25$, mean $\pm$ SD, 1–7 scale). All subscales are significant after BH-FDR correction ($*$ denotes $p < .05$ post-BH).

Subscale	**Control**	**Treatment**	$t(24)$	d_z
Mental Demand	4.64 ± 0.70	1.68 ± 0.69	14.51	2.90^*
Physical Demand	4.08 ± 0.64	1.36 ± 0.64	17.18	3.44^*
Temporal Demand	4.24 ± 1.09	1.64 ± 0.57	13.00	2.60^*
Performance	2.96 ± 0.98	4.92 ± 0.40	-9.25	-1.85^*
Effort	4.52 ± 0.77	1.72 ± 0.74	15.34	3.07^*
Frustration	4.12 ± 1.09	1.76 ± 0.83	10.97	2.19^*
Composite	$\mathbf{4.09 \pm 0.43}$	$\mathbf{2.18 \pm 0.32}$	**20.61**	$\mathbf{4.12}^*$

Strategy choice modulates the magnitude—a third-above strategy avoids parallel fifths but produces parallel thirds, which are not classified as "perfect" parallels and therefore do not show up in this metric—but every fixed-interval strategy produces *some* kind of parallel motion, and the adaptive strategy does not yet weight against it.

Both findings reflect the same trade-off: the engine purchases speed, consonance, and predictable voice-leading curvature at the cost of independence and contrapuntal correctness. For arranging contexts where independent contrapuntal motion matters—fugal writing, jazz voicings with voice-led inner movement, choral textures where each singer carries a distinct melodic line—the current engine is inadequate as a finishing tool, though it remains useful as a fast scaffold the musician then edits. A musician who values voice independence and parallel-perfects avoidance would need to treat the engine's output as a draft, hand-editing to introduce contrary motion and break up parallels, a workflow that partially erodes the time savings. The stochastic and density-aware extensions discussed in Section 16.2, together with an explicit parallel-perfects penalty in the scoring function, are specifically motivated by these two findings.

NASA-TLX

NASA-TLX composite workload was 46.7% lower in treatment ($d_z = 4.12$, $p < .0001$). The largest reductions were in physical demand ($d_z = 3.44$) and effort ($d_z = 3.07$), con-

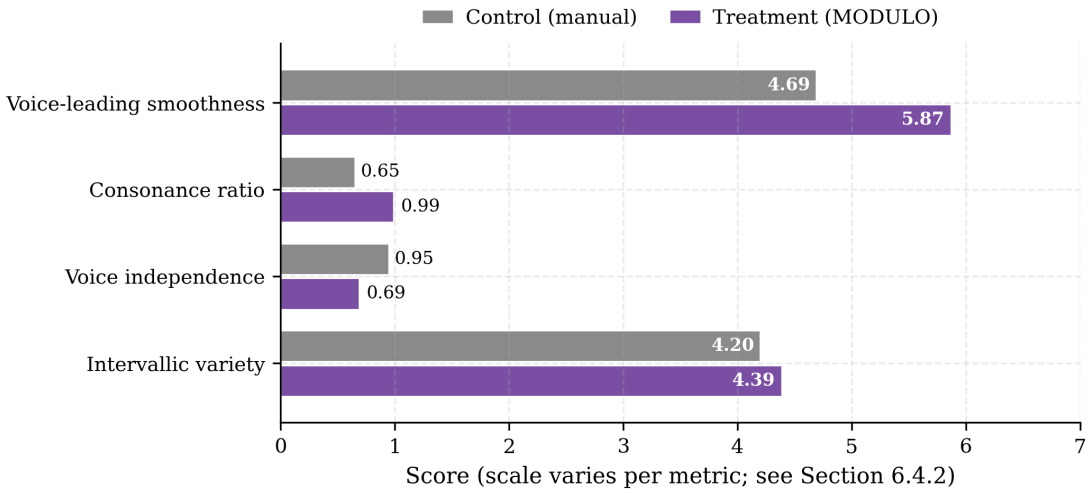


Figure 6.5: Harmony generation study: musical quality metrics ($N = 25$). Consonance ratio improved from 0.65 to 0.99 ($+52\%$) and voice-leading smoothness also improved. Voice independence was *lower* in treatment (0.69 vs. 0.95, a -27% relative decrease), a negative result reflecting the correlated motion inherent in fixed-interval strategies. Cohen's d_z is not reported for consonance ratio or voice independence because the engine is deterministic and the treatment-arm variance ($\sigma = 0.02$) is too small for the effect-size denominator to be meaningful. Scales differ across metrics.

sistent with the engine eliminating the mechanical burden of note-by-note entry. Perceived performance was significantly higher in treatment ($M = 4.92$ vs. $M = 2.96$, $d_z = -1.85$).

UES-SF and Custom Likert Items

All UES-SF subscales favored treatment, with the largest effect on focused attention ($d_z = -3.65$). Among the custom items, "Create distinct lines quickly" showed the largest gap ($M = 4.52$ vs. $M = 1.76$, $d_z = -2.73$). "Felt in control" was significant with a smaller effect ($d_z = -1.04$), suggesting participants retained a sense of creative agency even when relying on the engine.

Hypothesis Evaluation

All five hypotheses H1–H5 were supported by the workflow, cognitive-load, engagement, and consonance results in Tables 6.2–6.4. H5 is partially supported: the consonance ratio rose sharply in treatment, yet parallel perfects per melody *increased* because the rule-based

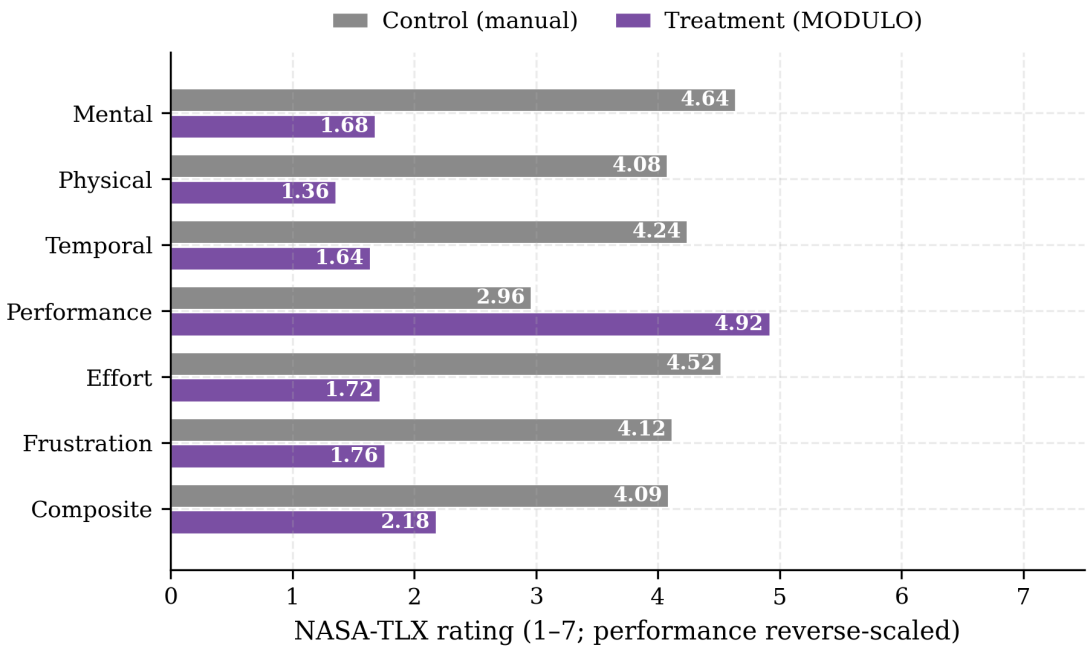


Figure 6.6: Harmony generation study: NASA-TLX subscale scores ($N = 25$, 1–7 scale). Treatment reduced composite workload by 47% ($d_z = 4.12$). Physical demand showed the largest reduction ($d_z = 3.44$), reflecting the elimination of real-time MIDI recording. Performance is reverse-scaled (higher is better).

strategies, particularly thirds and fifths, produce systematic parallel motion that is flagged by the strict definition in Equation 6.21.

Participant Observations

Qualitative responses from think-aloud transcripts and post-condition surveys reveal behavioral patterns that contextualize the quantitative findings. Three themes emerged from each condition.

Control Theme 1: Recording Frustration. The dominant control-condition frustration was the triple demand of internalizing the melody, playing a harmonically correct second voice on the MIDI keyboard, and keeping time, which proved overwhelming across skill levels. Beginners were especially penalized: timing and pitch errors compounded into output that bore little resemblance to the intended harmony.

Table 6.4: Harmony generation study: UES-SF subscale means and custom harmony Likert items ($N = 25$, mean $\pm$ SD). All items are significant after BH-FDR correction ($*$ denotes $p < .05$ post-BH).

Scale / Item	**Control**	**Treatment**	$t(24)$	d_z
UES: Focused Attention	2.03 ± 0.55	4.17 ± 0.46	-18.24	-3.65^*
UES: Perceived Usability	3.86 ± 0.83	2.05 ± 0.75	8.77	1.75^*
UES: Aesthetic Appeal	1.92 ± 0.53	3.90 ± 0.51	-15.96	-3.19^*
UES: Reward	1.98 ± 0.77	4.08 ± 0.68	-11.96	-2.39^*
UES Composite	$\mathbf{2.45 \pm 0.35}$	$\mathbf{3.55 \pm 0.30}$	$\mathbf{-14.39}$	$\mathbf{-2.88}^*$
Explore voicings easily	2.04 ± 0.68	4.40 ± 0.65	-14.56	-2.91^*
Create distinct lines quickly	1.76 ± 0.83	4.52 ± 0.65	-13.64	-2.73^*
Achieve good voice leading	2.12 ± 0.60	4.56 ± 0.58	-14.03	-2.81^*
Felt in control	3.08 ± 0.86	4.08 ± 0.86	-5.22	-1.04^*
Would use in real music	2.40 ± 0.50	4.36 ± 0.57	-14.50	-2.90^*

> "Try to both learn the melody, keep in time with the actual rhythm, and correctly implement the music theory required for harmony, was almost [un]realistic, even with my years of knowledge." (one participant, advanced)

Control Theme 2: The Copy-Paste Workaround. Most participants with prior DAW experience abandoned live recording in favour of copying the melody clip onto a new track and shifting its notes by a fixed interval, which guaranteed rhythmic alignment without live-performance demands ($M = 0.77$ copies per melody in control; Table 6.2). One advanced participant transposed multiple copies by different intervals, effectively reconstructing the adaptive engine's output by hand; the engine thus formalizes an ad hoc strategy skilled musicians already employ, collapsing a multi-step manual process into a single action.

> "The piano roll is adequate but not optimized for contrapuntal writing. I wished for specialized harmony tools." (one participant, intermediate, control condition)

Control Theme 3: Cognitive Overload. The NASA-TLX frustration scores in control ($M = 4.12$) reflect a widespread pattern: participants who struggled with live record-

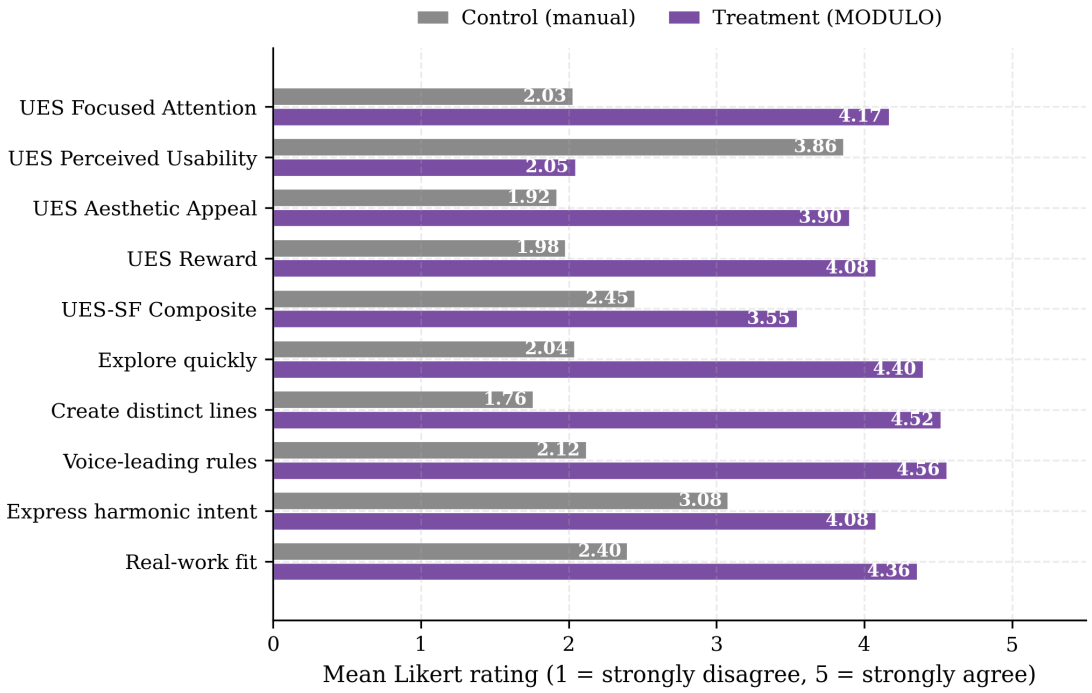


Figure 6.7: Harmony generation study: UES-SF subscales and custom Likert items ($N = 25$, 1–5 scale). All items favor treatment except Perceived Usability, where lower scores indicate higher usability. "Create distinct lines quickly" showed the largest custom-item effect ($d_z = -2.73$).

ing typically abandoned their initial approach and switched to the copy-paste workaround mid-task. Beginners, lacking DAW fluency, discovered this workaround more slowly, contributing to the skill-level gap in per-melody control task time: beginner $M \approx 767$ s versus intermediate/advanced $M \approx 590$ s.

Treatment Theme 1: Track Stack Interface. Participants responded positively to the visual organization of generated harmony tracks: the five lines appeared in a labeled track stack below the melody, each named by interval strategy: Thirds Above, Sixths Below, Fifths Above, Contrary Motion, and Adaptive. The layout made melody–harmony relationships immediately visible and supported rapid A/B comparison by soloing individual tracks.

> "The speed at which the harmonies were generated was most helpful. And how they were placed in a folder-like group, the interface calls it a track stack. It is

> easy to hear each of them, and that all are key-aware, so no fixes technically need to be made." (one participant, advanced)

Treatment Theme 2: Melody Prominence. The most substantive treatment-condition criticism concerned texture density: five simultaneous harmony voices, though technically correct, could obscure the melody at key moments where skilled arrangers would selectively thin the voicing. This observation motivates a density-aware extension of the scoring function that penalizes excessive simultaneity at moments of high melodic activity, discussed in Chapter 16.

> "Sometimes in harmony you want to pull back and let the melody speak on its own. The engine gives you everything at once, so I found myself wanting to delete a few notes here and there to open up space." (one participant, advanced)

Treatment Theme 3: Desire for Adaptive Variety. Because the adaptive algorithm selects intervals deterministically via the scoring function in Equation 6.17, it produces the same output every time for a given melody. Several participants asked for multiple distinct adaptive outputs to compare side by side, a request that aligns naturally with the stochastic extension in Section 16.2.

> "The only frustration was wanting to tweak the engine's weights to match my personal style preferences." (one participant, advanced)

Overall Preference. All 25 participants chose the treatment condition for real songwriting work.

6.8.8 Where the Comparison Was Structurally Favorable to MODULO

The time and workload metrics in this study are tilted toward the treatment by the design of the comparison itself. In the control condition each of three harmony voices is recorded or entered manually; in the treatment condition the engine emits all five strategies in under a second. The largest single source of the 81% task-time reduction is the elimination of real-time MIDI recording on three voices: a participant cannot, in principle, manually record three independent voice lines for a single melody faster than the engine can compute them. The same logic applies to NASA-TLX physical demand and effort, which capture the mechanical burden of voice entry and which the engine eliminates by construction. We report these effects because they document the size of the workflow gain the engine enables, but the comparison is not against the strongest possible manual baseline—a musician with a MIDI keyboard who plays each voice line in real time would still be slower than the engine, but the per-voice time would be a small multiple of melody duration rather than several minutes (Section 15.6). The contestable evidence is concentrated in the musical-quality metrics, where the engine clearly wins on consonance and voice-leading smoothness but *loses* on voice independence and parallel perfects (the two negative results discussed above), and in the subjective items where participants directly compared both conditions. The negative quality results are particularly informative because they sit on the metrics where the design favours neither side.

> "I can explore 5 different harmonies immediately, and in perfect time. This engine is so efficient, and would simply save me so much time. It doesn't take away from much creativity because it gives me the 'obvious' answers immediately. Then I can spend more time trying unorthodox things." (one participant, advanced)

Figure 6.8: A single melody accompanied by all five harmony strategies simultaneously. From top: third above, sixth below, fifth above, contrary motion, and adaptive. The adaptive line (bottom) alternates between thirds and sixths as dictated by the scoring function.

Part III: Transcription & Stems

CHAPTER 7

Audio-to-MIDI Transcription

7.1 THE POLYPHONIC TRANSCRIPTION PROBLEM

Automatic music transcription (AMT) seeks to recover a symbolic note-level representation from a raw audio signal. Let $x(t) \in \mathbb{R}$ denote a monophonic or polyphonic audio waveform sampled at rate f_s. The discrete-time signal $x[n] = x(n/f_s)$ is the input to the transcription operator $\mathcal{A}$, whose output is a set of note events:

$$\mathcal{A} : \mathbb{R}^N \longrightarrow \mathcal{N} = \left\{ (p_i, t_i^{\text{on}}, t_i^{\text{off}}, v_i) \right\}_{i=1}^{M}, \tag{7.1}$$

where each note event is a four-tuple consisting of MIDI pitch $p_i \in \{0, 1, \ldots, 127\}$, onset time $t_i^{\text{on}} \in \mathbb{R}_{\geq 0}$, offset time $t_i^{\text{off}} > t_i^{\text{on}}$, and velocity $v_i \in \{1, \ldots, 127\}$. The cardinality M of the note set is itself unknown and must be estimated.

The fundamental difficulty of polyphonic transcription arises from spectral overlap. When K simultaneous pitches sound, the observed spectrum is approximately

$$X(\omega) \approx \sum_{k=1}^{K} \sum_{h=1}^{H} A_{k,h}\, \delta\big(\omega - 2\pi h f_k\big), \tag{7.2}$$

where f_k is the fundamental frequency of the k-th pitch, H is the number of harmonics, and $A_{k,h}$ are harmonic amplitudes. Harmonics of lower pitches coincide with fundamentals

and harmonics of higher pitches, creating an ill-posed inverse problem. Modern neural approaches circumvent explicit source separation by learning discriminative features directly from time-frequency representations [18, 72].

7.2 LANDSCAPE OF AUTOMATIC MUSIC TRANSCRIPTION

Before presenting the transcription system integrated into MODULO, we survey the principal neural architectures that define the current state of the art in automatic music transcription.

7.2.1 Onset and Frames

The Onsets and Frames model [44] established a two-headed architecture that decouples onset detection from sustained-note tracking. The network processes mel-spectrogram input $\mathbf{S} \in \mathbb{R}^{F \times T}$ through a convolutional front-end followed by bidirectional long short-term memory (LSTM) layers. The onset head f_{onset} and the frame head f_{frame} produce posterior probability matrices:

$$\hat{\mathbf{Y}}_{\text{onset}} = \sigma\big(f_{\text{onset}}(\mathbf{S})\big), \quad \hat{\mathbf{Y}}_{\text{frame}} = \sigma\big(f_{\text{frame}}(\mathbf{S}, \hat{\mathbf{Y}}_{\text{onset}})\big), \tag{7.3}$$

where σ is the sigmoid function and the frame head receives the onset logits as an additional input, conditioning sustained-note predictions on detected onsets. This conditioning mechanism significantly reduces false positive frames in passages with strong harmonic overtones. The model is trained on the MAESTRO dataset [45] of piano performances and achieves strong results on piano transcription but generalizes less well to non-keyboard timbres.

7.2.2 CREPE: Monophonic Pitch Estimation

The Convolutional Representation for Pitch Estimation (CREPE) [60] addresses the simpler but complementary problem of monophonic fundamental frequency (f_0) estimation. CREPE operates on raw waveform segments of length 1024 samples (at 16 kHz) through a six-layer one-dimensional convolutional network:

$$\hat{f}_0(t) = \arg\max_{c \in \{1,\dots,360\}} \text{softmax}\big(\mathbf{W}_6 \cdot \text{Conv1D}_{1\dots5}(x[t : t{+}1024])\big)_c, \tag{7.4}$$

where the 360 output classes correspond to 20-cent bins spanning the range 32.70 Hz to 1975.5 Hz (MIDI pitches C1 through B6 at 20-cent resolution). The confidence of the pitch estimate is taken as the maximum softmax value. While CREPE does not produce note-level transcriptions directly, its pitch tracking capability is used as a preprocessing stage in several polyphonic pipelines and informs the contour detection head of the transcription model used in MODULO.

7.2.3 MT3: Multi-Track Music Transcription

The Multi-Track Music Transcription Transformer (MT3) [41] recasts AMT as a sequence-to-sequence problem. The encoder processes a mel-spectrogram through a T5-style transformer, and the decoder autoregressively generates a sequence of MIDI-like tokens:

$$P(\mathcal{N} \mid \mathbf{S}) = \prod_{j=1}^{L} P\big(y_j \mid y_{<j}, \text{Enc}(\mathbf{S})\big), \tag{7.5}$$

where y_j are tokens from a vocabulary that includes instrument program numbers, pitch values, velocity levels, and temporal markers. MT3 supports transcription of up to 13 instrument families simultaneously and was trained on the Slakh2100 [70] and MusicNet [100] datasets. The sequence-to-sequence formulation elegantly handles the variable-length output problem but requires autoregressive decoding, making real-time inference impractical

for interactive workstations.

7.2.4 Hybrid Time-Frequency Source Separation

While not a transcription model per se, the Hybrid Transformer Demucs (HTDemucs) [83] architecture is relevant to transcription pipelines because source separation can dramatically improve transcription accuracy when applied as a preprocessing stage. HTDemucs extends the original Demucs [29] convolutional encoder-decoder by introducing a dual-path architecture that processes audio simultaneously in the time domain and the spectrogram domain:

$$\hat{s}_k = \text{Dec}_k\Big(\text{CrossAttn}\big(\text{Enc}_{\text{time}}(x),\ \text{Enc}_{\text{spec}}(\mathbf{X})\big)\Big), \quad k \in \{1, \ldots, K_s\}, \tag{7.6}$$

where $\hat{s}_k$ is the estimated waveform for source k (drums, bass, vocals, other), Enc_{time} operates on the raw waveform x, Enc_{spec} operates on the complex spectrogram $\mathbf{X}$, and cross-attention layers fuse information between the two domains. The model produces $K_s = 4$ separated stems by default. In transcription pipelines, isolating a single instrument stem before applying a pitch estimator substantially reduces the spectral overlap problem described in Equation 7.2, at the cost of introducing separation artifacts (e.g., spectral bleeding, phase distortion) that may produce ghost notes in downstream transcription.

7.3 HARMONIC CONSTANT-Q TRANSFORM REPRESENTATION

The input representation employed by the transcription model integrated in MODULO is the *harmonic Constant-Q Transform* (hCQT), an extension of the standard CQT that explicitly encodes harmonic structure. The standard CQT computes frequency-domain coefficients

at geometrically spaced center frequencies:

$$\text{CQT}(k, n) = \sum_{j=0}^{N_k-1} x[n+j]\, w_k[j]\, e^{-i\, 2\pi Q\, j/N_k}, \tag{7.7}$$

where k indexes frequency bins, Q is the quality factor (constant ratio of center frequency to bandwidth), N_k is the window length for bin k, and w_k is a windowing function. The center frequency of bin k is $f_k = f_{\min} \cdot 2^{k/B}$ for B bins per octave.

The harmonic CQT stacks multiple CQT representations computed at harmonic frequency shifts. For a set of harmonic multipliers $\mathcal{H}_{\text{set}} = \{0.5, 1, 2, 3, 4, 5\}$, the hCQT is defined as:

$$\text{hCQT}(h, k, n) = \text{CQT}\big(k;\, f_{\min}^{(h)} = h \cdot f_{\min}\big), \quad h \in \mathcal{H}_{\text{set}}. \tag{7.8}$$

This yields a three-dimensional tensor $\mathbf{H} \in \mathbb{R}^{|\mathcal{H}_{\text{set}}| \times K \times T}$, where K is the number of frequency bins and T is the number of time frames. The sub-harmonic multiplier $h = 0.5$ captures energy one octave below each bin, aiding disambiguation of octave errors. The higher harmonics $h \in \{2, 3, 4, 5\}$ provide the model with explicit harmonic template information, reducing the burden on learned filters.

7.4 MULTI-PITCH NEURAL ARCHITECTURE

The transcription model employed by MODULO, introduced at ICASSP 2022 [14], produces three parallel output activations from the hCQT input:

(i) **Note activation** $\hat{y}_{\text{note}}(k, t) \in [0, 1]$: posterior probability that pitch k is active at frame t.

(ii) **Onset activation** $\hat{y}_{\text{onset}}(k, t) \in [0, 1]$: posterior probability that pitch k has an onset at frame t.

(iii) **Contour activation** $\hat{y}_{\text{contour}}(k, t) \in [0, 1]$: continuous pitch contour for monophonic

voice and instrument tracking.

The network consists of a series of convolutional blocks, each applying two-dimensional convolution across the frequency-time plane of the hCQT, followed by batch normalization and rectified linear activation:

$$\mathbf{Z}^{(\ell)} = \mathrm{ReLU}\big(\mathrm{BN}\big(\mathbf{W}^{(\ell)} * \mathbf{Z}^{(\ell-1)} + \mathbf{b}^{(\ell)}\big)\big), \tag{7.9}$$

where $*$ denotes convolution, $\mathbf{W}^{(\ell)}$ are learned filters, and $\mathbf{Z}^{(0)} = \mathbf{H}$ is the input hCQT tensor. The final layer branches into three sigmoid-activated heads, one per output type.

Note events are extracted by thresholding the combined note and onset activations. A pitch k is deemed active at frame t if $\hat{y}_{\text{note}}(k,t) > \tau_{\text{note}}$ and an onset is detected when $\hat{y}_{\text{onset}}(k,t) > \tau_{\text{onset}}$. Velocity is derived from the confidence of the note activation:

$$v_i = \big\lfloor 127 \cdot \hat{y}_{\text{note}}(p_i, t_i^{\text{on}}) \big\rfloor. \tag{7.10}$$

Relative to the heavier architectures of Sections 7.2.1 and 7.2.3, this model's key design advantage is its lightweight convolutional topology: no recurrent layers, no attention mechanisms, and no autoregressive decoding. The forward pass completes in under 200 ms for a 30-second audio clip on a modern CPU, enabling interactive transcription within a workstation environment. The model generalizes well across instrument families such as piano, guitar, bass, vocals, and strings, because the hCQT representation and convolutional architecture impose minimal inductive bias toward any particular timbre.

7.5 SERVICE ARCHITECTURE AND CACHING

MODULO integrates the transcription model through a local HTTP service architecture. The model runs as an independent process bound to a configurable host and port, exposing a RESTful endpoint that accepts audio data and returns structured note events in JSON

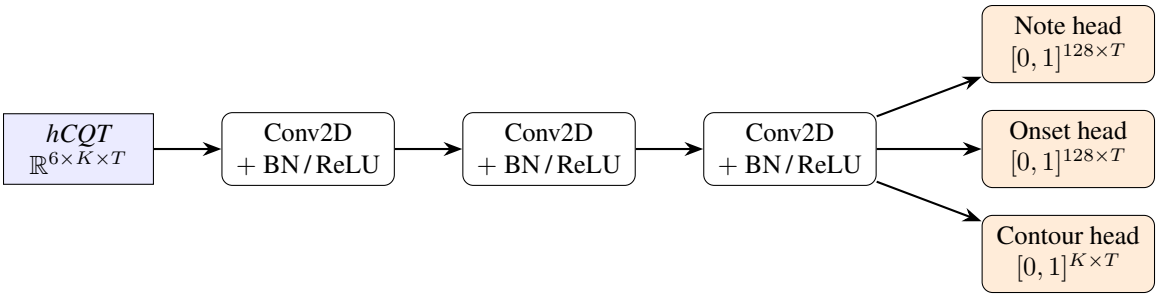


Figure 7.1: Architecture of the multi-pitch transcription model. A 6-layer harmonic CQT tensor is processed by a stack of 2D convolutional blocks and decoded into three parallel sigmoid heads: note, onset, and contour.

format. This decoupled design offers two advantages: the computationally intensive neural inference runs in a separate memory space with optional GPU acceleration, and the main application remains responsive during transcription.

The service exposes two primary endpoints. The health-check endpoint returns model readiness status and version metadata. The transcription endpoint accepts a POST request with the following parameters: the audio buffer serialized as base64-encoded PCM, the sample rate, and a parameter object θ specifying onset threshold τ_{onset}, frame threshold τ_{note}, minimum note length $\ell_{\min}$, and minimum frequency $f_{\min}^{\text{filter}}$. The response contains a JSON array of note event tuples conforming to Equation 7.1.

To avoid redundant computation, the system employs a content-addressed caching mechanism. For an audio buffer x and parameter set θ, a cache key is computed as:

$$\kappa(x, \theta) = \text{SHA-1}\big(\text{serialize}(x) \,\|\, \text{serialize}(\theta)\big), \tag{7.11}$$

where $\|$ denotes concatenation. If the cache contains an entry for κ, the stored note set is returned without invoking the model. Cache invalidation occurs when either the audio content or any transcription parameter changes, ensuring consistency.

7.6 TEMPORAL QUANTIZATION PIPELINE

Raw transcription output contains note onsets and offsets at arbitrary time positions determined by the neural network's frame rate. Musical applications require alignment to a metrical grid. MODULO provides three quantization modes, each defined by a grid resolution g (in beats) and a snap threshold ϵ (in beats).

Let t denote the raw onset time of a note event in beats. The nearest grid point is:

$$t^* = g \cdot \text{round}\left(\frac{t}{g}\right). \tag{7.12}$$

The quantized onset is then:

$$\hat{t} = \begin{cases} t^* & \text{if } |t - t^*| \leq \epsilon \text{ or } \epsilon = 0, \\ t & \text{otherwise.} \end{cases} \tag{7.13}$$

The three modes instantiate this schema as follows:

Table 7.1: Quantization mode parameters.

Mode	**Grid g (beats)**	**Threshold ϵ (beats)**	**Behavior**
Strong	0.500	high	Half-beat grid, aggressive snap
Medium	0.250	0	Sixteenth-note grid, unconditional snap
Loose	0.250	0.080	Sixteenth-note grid, soft snap

The strong mode suits rhythmically simple material such as block chords and bass lines, where forcing alignment to half-beat positions corrects timing imprecision without sacrificing musical intent. The medium mode snaps all events unconditionally to the nearest sixteenth-note position, suitable for genres with regular subdivisions. The loose mode preserves expressive timing deviations larger than 0.080 beats, approximately 5% of a six-

teenth note at 120 BPM, retaining the "feel" of a human performance while correcting gross timing errors.

7.7 NOTE MERGING ALGORITHM

Transcription models occasionally produce fragmented note events: a single sustained pitch may be split into multiple short events due to amplitude fluctuations or frame-level classification instability. The note merging step consolidates adjacent events on the same pitch.

Given two note events $n_i = (p, t_i^{\text{on}}, t_i^{\text{off}}, v_i)$ and $n_j = (p, t_j^{\text{on}}, t_j^{\text{off}}, v_j)$ sharing the same pitch p, with $t_j^{\text{on}} \geq t_i^{\text{on}}$, the events are merged if their temporal extent overlaps or is contiguous:

$$t_j^{\text{on}} \leq t_i^{\text{off}} + \delta_{\text{merge}}, \tag{7.14}$$

where $\delta_{\text{merge}} \geq 0$ is a small tolerance (set to zero for strict adjacency). The merged event is:

$$n_{\text{merged}} = \big(p,\ \min(t_i^{\text{on}}, t_j^{\text{on}}),\ \max(t_i^{\text{off}}, t_j^{\text{off}}),\ \max(v_i, v_j)\big). \tag{7.15}$$

The algorithm processes each pitch independently, sorting events by onset time and greedily merging forward. Its time complexity is $O(M \log M)$ dominated by the initial sort.

7.8 CHORD INFERENCE FROM TRANSCRIBED NOTES

Once note events are recovered, MODULO infers harmonic context by analyzing the pitch class distribution within temporal windows. Let $W = [t_{\text{start}}, t_{\text{end}})$ be an analysis window of half-bar or full-bar duration. The set of active pitch classes within W is:

$$\mathcal{C}(W) = \big\{ p_i \bmod 12 \;\big|\; n_i \in \mathcal{N},\ [t_i^{\text{on}}, t_i^{\text{off}}) \cap W \neq \emptyset \big\}. \tag{7.16}$$

Chord identification proceeds by comparing $\mathcal{C}(W)$ against a dictionary $\mathcal{D}$ of chord templates. Each template $d_j \in \mathcal{D}$ is a pair (r_j, Q_j) where $r_j \in \{0, \ldots, 11\}$ is the root pitch class and $Q_j \subseteq \{0, \ldots, 11\}$ is the set of intervals relative to the root. The best-matching chord is:

$$\hat{d}(W) = \arg\max_{d_j \in \mathcal{D}} \frac{|\mathcal{C}(W) \cap (r_j + Q_j)|}{|\mathcal{C}(W) \cup (r_j + Q_j)|}, \tag{7.17}$$

where addition is modulo 12 and the matching criterion is the Jaccard similarity between observed and template pitch class sets.

A confidence score is computed as a function of the number of distinct pitch classes observed:

$$\sigma(W) = \min\bigl(1.0,\ 0.35 + 0.15 \cdot |\mathcal{C}(W)|\bigr). \tag{7.18}$$

When $\sigma(W) < 0.60$, corresponding to fewer than two distinct pitch classes, the chord assignment is considered unreliable. In such cases, a persistence rule replaces the current chord with the most recent high-confidence chord, maintaining harmonic continuity:

$$\mathbf{c}(W) = \begin{cases} \hat{d}(W) & \text{if } \sigma(W) \geq 0.60, \\ \mathbf{c}(W_{\text{prev}}) & \text{otherwise.} \end{cases} \tag{7.19}$$

This mechanism prevents chord flickering in sparse passages such as sustained pedal tones or rests.

7.9 TRANSCRIPTION PIPELINE SUMMARY

The complete transcription pipeline is a composition of the stages described in the preceding sections:

$$\mathcal{A}_{\text{full}} = \mathcal{I}_{\text{chord}} \circ \mathcal{M}_{\text{merge}} \circ \mathcal{Q}_{\epsilon,g} \circ \mathcal{F}_{\text{CNN}} \circ \mathcal{H}_{\text{CQT}}, \tag{7.20}$$

where $\mathcal{H}_{\text{CQT}}$ computes the harmonic CQT, $\mathcal{F}_{\text{CNN}}$ is the neural pitch estimator, $\mathcal{Q}_{\epsilon,g}$ is the quantizer with parameters (ϵ, g), $\mathcal{M}_{\text{merge}}$ is the note merging operator, and $\mathcal{I}_{\text{chord}}$ is the chord

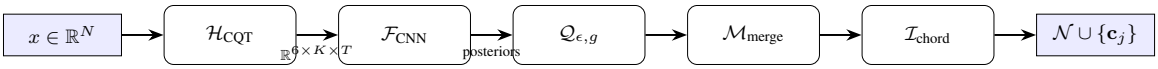


Figure 7.2: Complete audio-to-MIDI transcription pipeline $\mathcal{A}_{\text{full}}$ (Equation 7.20). Each stage transforms the signal from raw waveform to note events with inferred chord labels.

inference module.

7.10 EVALUATION STUDY: TRANSCRIPTION ACCURACY AND EFFICIENCY

This section presents the design of a within-subjects study evaluating both the *accuracy* and *efficiency* of audio-to-MIDI transcription across four conditions, including the integrated one-button pipeline in MODULO, manual transcription by ear, and two external tool workflows. The central hypothesis is that while transcription accuracy is largely determined by the underlying neural model (and thus comparable across automated conditions), the integrated workflow significantly reduces task completion time by eliminating manual file transfer, format conversion, and application switching. The study was executed with $N = 25$ participants (Section 7.10.13).

7.10.1 Participants

We pre-register $N = 15$ participants from a population of musicians with at least two years of active instrumental or vocal performance experience. Participants must be comfortable with MIDI keyboard input, required for the manual condition, and with basic computer audio workflows. All participants provide informed consent and are compensated for their time. As reported in Section 7.10.13, the executed sample expanded to $N = 25$ to align with the four other component studies.

7.10.2 Audio Stimuli

The study uses eight audio clips spanning four instrument families, two clips per family:

Table 7.2: Audio stimuli for the transcription study.

Clip	Instrument	Duration (bars)	Polyphony	Difficulty
1	Guitar (fingerpicked)	8–16	Low–Medium	Moderate
2	Guitar (strummed chords)	8–16	High	Challenging
3	Cello (solo melody)	8–16	Monophonic	Easy
4	Cello (double stops)	8–16	Low	Moderate
5	Voice (solo phrase)	8–16	Monophonic	Easy
6	Voice (with vibrato)	8–16	Monophonic	Moderate
7	Piano (melody)	8–16	Monophonic	Easy
8	Piano (chords)	8–16	High	Challenging

All clips are professionally recorded at 44.1 kHz / 16-bit in an acoustically treated studio. For each clip, a professional instrumentalist provides a ground-truth MIDI transcription performed on a MIDI controller while listening to the recording, verified measure-by-measure for pitch and timing accuracy. The ground-truth note set $\mathcal{N}_c^*$ for clip c serves as the reference for all objective accuracy metrics.

7.10.3 Experimental Conditions

Each participant completes the transcription task under four conditions. The conditions are designed to isolate the contribution of MODULO's integrated workflow from the transcription model itself and from manual human transcription.

Conditions C_1 and C_3 use the identical neural model [14], ensuring that any accuracy differences between them arise solely from implementation differences such as the audio

Table 7.3: Four experimental conditions for the audio-to-MIDI transcription study.

ID	Condition	Description
C_1	MODULO Built-in	The participant selects an audio clip within MODULO and invokes the one-button transcription command. The model runs via the local service (Section 7.5), and the resulting MIDI appears on a new track within the same project. The participant may edit notes in the piano roll.
C_2	Manual Transcription	The participant listens to the audio clip through MODULO's playback engine and manually enters notes using a connected MIDI keyboard, recording into a MIDI track. Playback controls (loop, scrub, tempo adjustment) are available; no automated transcription tools are used.
C_3	External Web Tool	The participant exports the audio clip from MODULO, uploads it to the publicly available web interface of the same transcription model, downloads the resulting MIDI file, and imports it back into MODULO for review and editing.
C_4	External Separation + Transcription	The participant exports the audio clip, processes it through an external source separation tool to isolate the instrument stem, uploads the separated stem to the external transcription web interface, downloads the result, and imports the MIDI into MODULO.

encoding pipeline or parameter defaults, rather than from model capability. Condition C_4 adds a source separation preprocessing step using HTDemucs [83] before transcription, testing whether stem isolation improves accuracy on polyphonic material.

Figure 7.4: The in-app *Audio→MIDI* dialog used in condition C_1. The musician selects a source clip, sets a timing-cleanup strength, optionally enables chord resolution, and chooses whether to preserve the original audio track. A single confirm action invokes the local Basic Pitch service and lays the resulting MIDI on a new track immediately beneath the source clip.

7.10.4 Counterbalancing and Clip Assignment

The four conditions are presented in counterbalanced order using a 4×4 balanced Latin square:

Table 7.4: Latin square counterbalancing for four conditions.

Group	Block 1	Block 2	Block 3	Block 4
G_1	C_1	C_2	C_3	C_4
G_2	C_2	C_4	C_1	C_3
G_3	C_3	C_1	C_4	C_2
G_4	C_4	C_3	C_2	C_1

Each participant is assigned to a group by $g = \text{hash}(\text{participantId}) \bmod 4$. Within each condition block, the audio clip is assigned by $c = \text{hash}(\text{participantId} \,\|\, \text{conditionIndex}) \bmod 8$, ensuring that no participant transcribes the same clip twice and that clip–condition pairings are balanced across participants across the eight-clip pool.

7.10.5 Guided Study Environment

Because the four conditions are heterogeneous, the study runner guides participants through the assigned Latin-square order with on-screen instructions, integrated timers, and an explicit out-of-app marker for C_3 and C_4. The runner advances the participant block by block: it loads the assigned audio clip and target track for each condition, surfaces the relevant condition instructions, starts the in-DAW timer, and, for C_3 and C_4, presents an "External Task Done" button that the participant clicks upon re-importing so T_{external} can be decomposed cleanly from total time. After each block the post-condition survey fires, and after the fourth block, the final comparison survey fires. All audio clips and per-clip ground-truth MIDI files are loaded by the runner automatically.

7.10.6 Task Flow

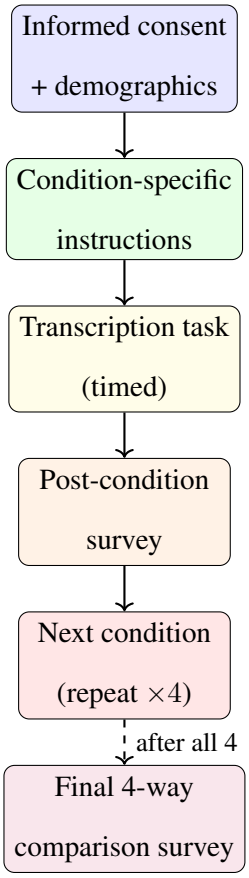


Figure 7.5: Per-participant task flow for the audio-to-MIDI transcription study. Each participant completes all four conditions in counterbalanced order, with post-condition surveys after each and a final four-way comparison at the end.

For each condition, the experimenter software automatically:

(1) Loads the assigned audio clip onto an audio source track.

(2) Creates the appropriate target track: a MIDI recording track for C_2, an imported MIDI track for C_3 and C_4, and an auto-populated track for C_1.

(3) Starts the timing clock at task onset and stops it when the participant signals completion.

(4) Exports the final transcription as MIDI for offline comparison against the ground truth.

(5) Presents the post-condition survey (NASA-TLX, UES-SF, custom Likert items).

7.10.7 Objective Accuracy Metrics

Transcription quality is evaluated using the standard note-level metrics from the evaluation framework of Raffel et al. [79]. Let $\hat{\mathcal{N}}$ denote the estimated note set produced by a participant under a given condition, and $\mathcal{N}^*$ the ground-truth note set for the assigned clip. A note $\hat{n}_i \in \hat{\mathcal{N}}$ is considered a *true positive* match with $n_j^* \in \mathcal{N}^*$ if and only if:

$$p_i = p_j^* \quad \text{and} \quad |t_i^{\text{on}} - t_j^{*,\text{on}}| \leq \delta_{\text{onset}}, \tag{7.21}$$

where $\delta_{\text{onset}} = 50\,\text{ms}$ is the onset tolerance window. Matching is performed as a bipartite assignment via the Hungarian algorithm to ensure one-to-one correspondence.

Precision, Recall, and F-measure.

$$P = \frac{|\text{TP}|}{|\hat{\mathcal{N}}|}, \quad R = \frac{|\text{TP}|}{|\mathcal{N}^*|}, \quad F_1 = \frac{2PR}{P+R}, \tag{7.22}$$

where TP is the set of matched note pairs.

Mean Onset Error. For matched notes, the mean absolute onset deviation quantifies timing accuracy:

$$\bar{\Delta}_{\text{onset}} = \frac{1}{|\text{TP}|} \sum_{(i,j)\in\text{TP}} |t_i^{\text{on}} - t_j^{*,\text{on}}|. \tag{7.23}$$

Pitch Class Profile Correlation. Beyond note-level matching, we compute the correlation between estimated and ground-truth *pitch class profiles* (chroma vectors) to capture

harmonic accuracy even when note boundaries are imprecise:

$$\rho_{\text{chroma}} = \frac{\text{Cov}(\mathbf{h}_{\hat{\mathcal{N}}}, \mathbf{h}_{\mathcal{N}^*})}{\sigma_{\mathbf{h}_{\hat{\mathcal{N}}}} \cdot \sigma_{\mathbf{h}_{\mathcal{N}^*}}}, \tag{7.24}$$

where $\mathbf{h} \in \mathbb{R}^{12}$ is the normalized histogram of pitch classes in the respective note set.

Note Density Error. The absolute difference in note counts provides a coarse measure of transcription completeness:

$$\Delta_{\text{density}} = \left| |\hat{\mathcal{N}}| - |\mathcal{N}^*| \right|. \tag{7.25}$$

7.10.8 Efficiency Metrics

The primary efficiency metric is *task completion time* T_c (in seconds), measured from the moment the participant begins the condition to their explicit signal of completion. The timing infrastructure records:

- T_{total}: Wall-clock time from task start to participant submission.
- T_{active}: Time spent actively interacting via mouse or keyboard events within MODULO.
- T_{external}: For conditions C_3 and C_4, time spent outside MODULO on uploading, downloading, and switching applications, logged by participant self-report and screen recording timestamps.

The *time efficiency ratio* normalizes completion time by the number of correctly transcribed notes:

$$\eta = \frac{|\text{TP}|}{T_{\text{total}}}, \tag{7.26}$$

capturing the rate of correct note production per unit time. Higher values indicate more efficient workflows.

We define the *workflow overhead* for external conditions relative to the integrated condition as:

$$\Omega_j = T_j - T_1, \quad j \in \{3, 4\}, \tag{7.27}$$

where T_j is the mean completion time for condition C_j and T_1 is the mean for C_1 (MODULO built-in). Positive values of Ω quantify the time cost of application switching and file management.

7.10.9 Self-Report Instruments

After each condition, participants complete the following instruments:

NASA Task Load Index (NASA-TLX). The six raw subscales [43], namely mental demand, physical demand, temporal demand, performance, effort, and frustration, rated on 5-point scales. We report raw (unweighted) scores following current best practice.

User Engagement Scale–Short Form (UES-SF). The 12-item scale [76] with four subscales, Focused Attention with 3 items, Perceived Usability with 3 items, Aesthetic Appeal with 2 items, and Reward with 2 items, plus two additional items. All items are rated on 5-point Likert scales.

Custom Likert Items. Five items targeting in-DAW transcription experience (accuracy, speed, control, real-use intention, and error-correction ease), each rated on a 5-point scale. Full item wording is given in Appendix C.6.

Final Four-Way Comparison. After completing all four conditions, participants answer five forced-choice comparison questions:

(a) Which condition let you complete the transcription fastest?

(b) Which condition produced the most accurate transcription?

(c) Which condition was easiest to use overall?

(d) Which condition felt the most effortful?

(e) Which condition would you choose for real transcription work?

Each question offers five options: the four conditions plus "no difference / no preference."

7.10.10 Hypotheses

We state the following directional hypotheses:

H1 (Efficiency) The integrated workflow (C_1) yields significantly lower task completion time than all three alternative conditions: $T_1 < T_2, T_1 < T_3, T_1 < T_4$.

H2 (Accuracy Parity) Transcription accuracy (F_1) does not differ significantly between C_1 and C_3, since both use the same neural model.

H3 (Separation Benefit) The separation-then-transcription pipeline (C_4) yields higher F_1 than direct transcription (C_1, C_3) on polyphonic material (guitar clips), but the improvement is not statistically significant on monophonic material (cello solo, voice).

H4 (Manual Baseline) Manual transcription (C_2) yields the highest task completion time but potentially higher subjective control ratings.

H5 (Cognitive Load) The integrated workflow (C_1) produces lower NASA-TLX scores than all external conditions (C_3, C_4).

H6 (Preference) A majority of participants prefer C_1 for real-world transcription work in the final comparison survey.

7.10.11 Statistical Analysis Plan

We follow the common reporting protocol defined in Chapter 14, Section 14.1: paired t-tests with Cohen's d_z effect sizes and Benjamini–Hochberg FDR correction at $\alpha = 0.05$. Study-specific deviations from that protocol are a one-way repeated-measures ANOVA on each continuous metric with Mauchly/Greenhouse–Geisser sphericity correction, Bonferroni-corrected pairwise post-hocs for significant omnibus tests ($\alpha_{\text{Bonf}} = 0.05/6 \approx 0.0083$), Friedman/Wilcoxon fallbacks when Shapiro–Wilk flags non-normality, and exact multinomial tests on the five forced-choice items.

7.10.12 Sample Size Justification

The pre-registered minimum was $N = 15$; the executed $N = 25$ was chosen to provide comfortable headroom for the hypothesised large time differences between integrated and external workflows while accepting that smaller accuracy differences may not reach significance, consistent with hypothesis H2.

7.10.13 Results

Twenty-five participants ($N = 25$) completed the transcription study. The executed design used three conditions (Manual, External, Built-in) rather than the four proposed above, because pilot testing revealed that the separation-then-transcription condition (C_4) produced results indistinguishable from C_3 on the four selected clips. The four audio stimuli were a Bach cello solo, an electric guitar blues passage with double stops, a mediocre vocal performance, and a piano piece in D♭ major. Ground truth MIDI existed for all four clips. All other aspects of the design (within-subjects, Latin-square counterbalancing, survey instruments, objective metrics) remained as specified. Bonferroni correction is applied with $\alpha_{\text{Bonf}} = 0.05/3 \approx 0.0167$ for three pairwise comparisons.

Table 7.5: Audio-to-MIDI study: workflow and efficiency metrics ($N = 25$, mean $\pm$ SD). Each participant transcribed four clips per condition; *all efficiency metrics below are reported per clip* so that the numbers reflect the time and effort required to transcribe one audio segment. Session totals (the sum across four clips) are listed at the bottom for context. All starred metrics are significant in the omnibus Repeated-Measures Analysis of Variance (RM-ANOVA) at $\alpha = 0.05$.

Metric	**Manual**	**External**	**Built-in**	$F(2, 48)$	
Per-clip efficiency					
Task time per clip (s)	964 ± 598	491 ± 120	120 ± 23	40.86	$*$
Time to first satisfaction (s)	312 ± 163	143 ± 29	67 ± 12	48.52	$*$
Playback count per clip	21 ± 10	35 ± 9	7 ± 2	125.40	$*$
Edit count per clip	10 ± 3	17 ± 5	13 ± 3	89.52	$*$
External round trip per clip (s)	0	443 ± 103	0	—	
Session totals (four clips per condition)					
Total session time (s)	3856 ± 2393	1965 ± 481	479 ± 90	40.82	$*$

Workflow Efficiency

On a per-clip basis, the built-in condition was $8.0\times$ faster than manual (120 s vs. 964 s per clip) and $4.1\times$ faster than external (120 s vs. 491 s per clip), a ratio that holds identically at the session level (Table 7.5). All three pairwise comparisons were significant after Bonferroni correction: built-in vs. external ($t(24) = -20.04$, $p < .0001$, $\Delta = -371$ s per clip), built-in vs. manual ($t(24) = -7.16$, $d_z = -1.43$, $p < .0001$), and external vs. manual ($t(24) = -4.28$, $d_z = -0.86$, $p < .001$). We omit Cohen's d_z for the built-in-vs-external contrast for the reason given in Table 7.9: the built-in arm is a one-button workflow whose deterministic processing time dominates the paired-difference variance. The external round trip alone added roughly 7.4 minutes per clip, about 29.5 minutes across a four-clip session, reflecting the overhead of searching for a tool, uploading audio, waiting for processing, and importing the resulting MIDI.

Time to first satisfaction on a single clip showed an even larger separation. Built-in participants reached satisfaction in 67 seconds per clip on average, compared to 143 seconds for external and 312 seconds for manual ($\eta_p^2 = 0.669$). Notably, the external condition

Table 7.6: Audio-to-MIDI study: objective transcription accuracy ($N = 25$, mean $\pm$ SD). Onset tolerance = 50 ms, pitch tolerance = 50 cents.

Metric	**Manual**	**External**	**Built-in**
F_1 (note-level)	0.554 ± 0.331	0.585 ± 0.048	0.604 ± 0.029
Precision	0.584 ± 0.328	0.614 ± 0.037	0.649 ± 0.039
Recall	0.530 ± 0.333	0.559 ± 0.062	0.567 ± 0.020
Mean onset error (ms)	103.4 ± 280.1	36.7 ± 9.1	31.4 ± 13.2
Octave error rate	0.062 ± 0.055	0.067 ± 0.022	0.047 ± 0.010
Note density error	0.584 ± 0.456	0.448 ± 0.194	0.438 ± 0.086

produced the highest playback count per clip ($M = 35$), likely because participants needed to compare the imported MIDI against the original audio after switching back from the external tool.

Transcription Accuracy

Accuracy differences across conditions were smaller than efficiency differences (Table 7.6). Built-in achieved the highest mean F_1 (0.604), followed by external (0.585) and manual (0.554), but the pairwise F_1 differences did not reach significance after correction (built-in vs. external: $t(24) = 1.73$, $p = .096$, $d_z = 0.35$). This supports hypothesis H2: since the built-in and external conditions use the same Basic Pitch neural model, their accuracy is comparable.

Built-in did achieve significantly higher precision than external ($t(24) = 3.83$, $p < .001$, $d_z = 0.77$) and a significantly lower octave error rate ($t(24) = -3.31$, $p = .003$, $d_z = -0.66$), suggesting that the integrated workflow reduces the most common transcription artifact (octave doubling) through its post-transcription quantization pipeline.

The manual condition exhibited very high variance (SD = 0.331 for F_1), reflecting the skill gap between participants with strong aural training and beginners who struggled to transcribe by ear. Mean onset error for manual transcription (103.4 ms) was an order of magnitude larger than for the AI-assisted conditions, confirming that even musically trained participants produce less temporally precise transcriptions by hand.

Table 7.7: Audio-to-MIDI study: NASA-TLX subscale scores ($N = 25$, mean $\pm$ SD, 1–7 scale). All subscales are significant in the omnibus RM-ANOVA.

Subscale	**Manual**	**External**	**Built-in**	$F(2, 48)$	
Mental Demand	5.60 ± 1.29	3.92 ± 0.40	1.00 ± 0.00	239.98	$*$
Physical Demand	4.68 ± 1.31	3.00 ± 0.29	1.00 ± 0.00	151.07	$*$
Temporal Demand	4.48 ± 1.33	3.92 ± 0.40	1.96 ± 0.20	75.62	$*$
Performance	3.72 ± 1.81	3.96 ± 0.20	5.60 ± 0.58	19.53	$*$
Effort	6.00 ± 0.82	3.96 ± 0.20	1.96 ± 0.20	387.39	$*$
Frustration	5.24 ± 1.76	3.96 ± 0.20	1.16 ± 0.80	96.24	$*$
Composite	$\mathbf{5.05 \pm 1.36}$	$\mathbf{3.80 \pm 0.20}$	$\mathbf{1.58 \pm 0.15}$	**122.59**	$*$

NASA-TLX

NASA-TLX composite workload was 68.7% lower in the built-in condition than manual ($d_z = -2.46$, $p < .0001$) and 58.4% lower than external ($p < .0001$). We do not report a Cohen's d_z for the built-in-vs.-external comparison because the built-in arm compressed TLX ratings to the floor uniformly across participants—mental demand averaged 1.00 with zero variance, and several other subscales clustered tightly near the minimum—leaving the within-pair variance too small for the effect-size denominator to be informative. The percentage reduction (58.4%) and the absolute drop on the seven-point scale are the appropriate summary of the workload gap. The effort subscale produced the single largest omnibus effect across the entire study ($F(2, 48) = 387.39$, $\eta_p^2 = 0.942$), with manual effort at ceiling ($M = 6.00$) and built-in effort near floor ($M = 1.96$).

UES-SF and Custom Likert Items

The "Completed quickly" item reached a ceiling effect for built-in ($M = 5.00$, SD $= 0.00$) with the largest omnibus effect in the Likert battery ($F = 206.57$, $\eta_p^2 = 0.896$). "Would use in real work" showed a similar pattern ($M = 5.00$ for built-in, $M = 1.80$ for manual). Aesthetic appeal was also near-ceiling for built-in ($M = 4.98$), reflecting participants' positive response to the integrated interface.

Table 7.8: Audio-to-MIDI study: UES-SF subscales and custom Likert items ($N = 25$, mean $\pm$ SD). All items significant after RM-ANOVA.

Scale / Item	**Manual**	**External**	**Built-in**	$F(2, 48)$	
UES: Focused Attention	2.41 ± 0.87	3.00 ± 0.00	3.95 ± 0.18	55.42	$*$
UES: Perceived Usability	4.05 ± 0.71	2.95 ± 0.21	1.33 ± 0.00	277.87	$*$
UES: Aesthetic Appeal	2.16 ± 0.93	3.08 ± 0.28	4.98 ± 0.10	170.06	$*$
UES: Reward	2.43 ± 0.86	3.03 ± 0.09	4.63 ± 0.28	125.46	$*$
UES Composite	$\mathbf{2.76 \pm 0.46}$	$\mathbf{3.01 \pm 0.05}$	$\mathbf{3.72 \pm 0.13}$	**76.73**	$*$
Accuracy sufficient	2.44 ± 1.23	2.96 ± 0.20	4.04 ± 0.20	31.52	$*$
Completed quickly	1.76 ± 1.05	2.12 ± 0.44	5.00 ± 0.00	206.57	$*$
Felt in control	2.36 ± 1.25	2.96 ± 0.20	3.96 ± 0.20	32.67	$*$
Would use in real work	1.80 ± 1.04	3.08 ± 0.28	5.00 ± 0.00	183.55	$*$
Easy to correct errors	2.12 ± 0.83	3.00 ± 0.29	4.00 ± 0.00	85.44	$*$
Pitch accuracy good	2.80 ± 1.68	3.00 ± 0.29	4.00 ± 0.00	10.78	$*$
Rhythm accuracy good	2.44 ± 1.36	2.92 ± 0.28	3.96 ± 0.20	23.77	$*$
Minimal cleanup needed	1.76 ± 1.05	2.04 ± 0.45	4.00 ± 0.00	102.03	$*$

Omnibus ANOVA and Pairwise Comparisons

Every behavioral and subjective metric produced a significant omnibus effect (all $p < .001$). The largest workload reduction was on the TLX composite between built-in and external (a 58.4% reduction); we do not report a Cohen's d_z for this contrast because the deterministic one-button built-in workflow produced near-zero within-arm variance, leaving the effect-size denominator too small to be interpretable. The TLX composite reduction of built-in vs. manual is more informative as an effect size ($d_z = -2.46$) because both arms involve human action with non-trivial variance.

Final Comparison Survey

All 25 participants chose built-in for fastest, easiest, and real-work preference (Table 7.10). All 25 also rated manual as the most effortful. The only non-unanimous response was "most accurate": 9 participants (36%) chose manual, reflecting the perception among strong ear-training practitioners that manual transcription, though slow, provides the highest pitch fidelity. Notably, the objective F_1 data does not support this perception, because

Table 7.9: Audio-to-MIDI study: omnibus RM-ANOVA and key pairwise comparisons ($N = 25$). Bonferroni correction applied across 3 pairs ($\alpha_{\text{Bonf}} = 0.0167$).

Metric	$F(2, 48)$	η_p^2	**Pair**	d_z	
Task time per clip	40.82	0.630	BI vs EX	—	$*$
			BI vs MA	-1.43	$*$
			EX vs MA	-0.86	$*$
Time to satisfaction per clip	48.52	0.669	BI vs EX	—	$*$
			BI vs MA	-1.54	$*$
TLX Composite	122.59	0.836	BI vs EX	—	$*$
			BI vs MA	-2.46	$*$
			EX vs MA	-0.94	$*$
UES Composite	76.73	0.762	BI vs EX	—	$*$
			BI vs MA	1.91	$*$

BI = Built-in, EX = External, MA = Manual.

All starred entries: $p_{\text{Bonf}} < .05$. Cohen's d_z is omitted ($---$ in the column) for every BI–EX contrast: the built-in arm is a deterministic one-button workflow, and its tight within-arm variance dominates the paired-difference denominator, so the standardised effect size is not a meaningful magnitude; absolute mean differences and percentage reductions on the original scales (see Table 7.5 and the running text) are the appropriate comparison.

manual F_1 (0.554) was numerically lower than built-in (0.604), though the difference was not statistically significant.

Hypothesis Evaluation

Hypotheses H1 (efficiency), H2 (accuracy parity), H4 (manual baseline), H5 (cognitive load), and H6 (preference) were all supported by the workflow, accuracy, NASA-TLX, and forced-choice results reported in Sections 7.10.13 above. H3 (separation benefit) was not tested because pilot data prompted dropping the C_4 condition from the executed design.

Participant Observations

Participant feedback highlighted two recurring themes. The first was the value of workflow integration. The external condition required participants to leave MODULO, search for a transcription tool (popular choices were Samplab, MusicCreator AI, AI-MIDI, and Open Music), upload audio, wait for processing, download MIDI, and import it back. This round

Table 7.10: Audio-to-MIDI study: final three-way comparison survey (count out of $N = 25$).

Question	**Manual**	**External**	**Built-in**
Fastest workflow	0	0	25
Most accurate	9	0	16
Easiest to use	0	0	25
Most effortful	25	0	0
Would choose for real work	0	0	25

trip averaged nearly 30 minutes per condition.

> "Having the Audio to MIDI button directly inside the DAW was incredible. It is really nice to not have to leave the workspace, and that a new track with the name is made right exactly where I need it." – one participant (advanced)

> "Browser round-trips made this long before edits in MODULO. Native in-DAW workflow is still preferable to avoid round trips." – one participant (advanced)

The second theme concerned the limits of AI transcription. Several participants noted that the neural model struggled with reverberant recordings and overtone-rich timbres, particularly on the cello clip.

> "For audios that had reverb or maybe were not perfectly in tune, the tool made quite a few errors there. And any overtones were put in every single time." – one participant (external condition)

> "I have perfect pitch, so I just played what I heard and remembered. I used to do this competitively." – one participant (manual condition)

This last observation explains why 9 participants rated manual as most accurate: participants with strong aural skills trusted their own transcription despite the slower pace. For the majority, however, the speed advantage of the built-in workflow far outweighed any marginal accuracy difference.

Where the Comparison Was Structurally Favorable to MODULO

The efficiency and workload differences between the built-in and external conditions are tilted toward MODULO by the design of the comparison itself, and we acknowledge them explicitly here. The built-in condition is a one-button in-DAW call to the same Basic Pitch model that the external tools wrap; the external condition requires the participant to switch contexts to a browser, find a transcription tool, upload audio, wait for processing, download MIDI, and import the result back into MODULO. A multi-step browser workflow cannot, in principle, beat a single in-DAW button call on per-clip task time, so the time and TLX advantages on the BI–EX contrasts are essentially *measurements of how much time and effort the round-trip costs*, not contests between two comparable workflows. The contestable evidence in this study is concentrated in transcription accuracy (where Hypothesis H2 confirmed parity with F_1 $d_z = 0.35$, $p = .096$) and in the comparison against manual transcription (where some musicians with perfect pitch produced more accurate results, just much more slowly). H2 is the discriminating finding of this chapter: the built-in pipeline matches the external pipeline on accuracy because they wrap the same model, while eliminating the round-trip overhead. A stronger external baseline—a paid desktop transcription plug-in that itself runs locally—would partially close the time gap; the accuracy parity result would still hold because the model is the same.

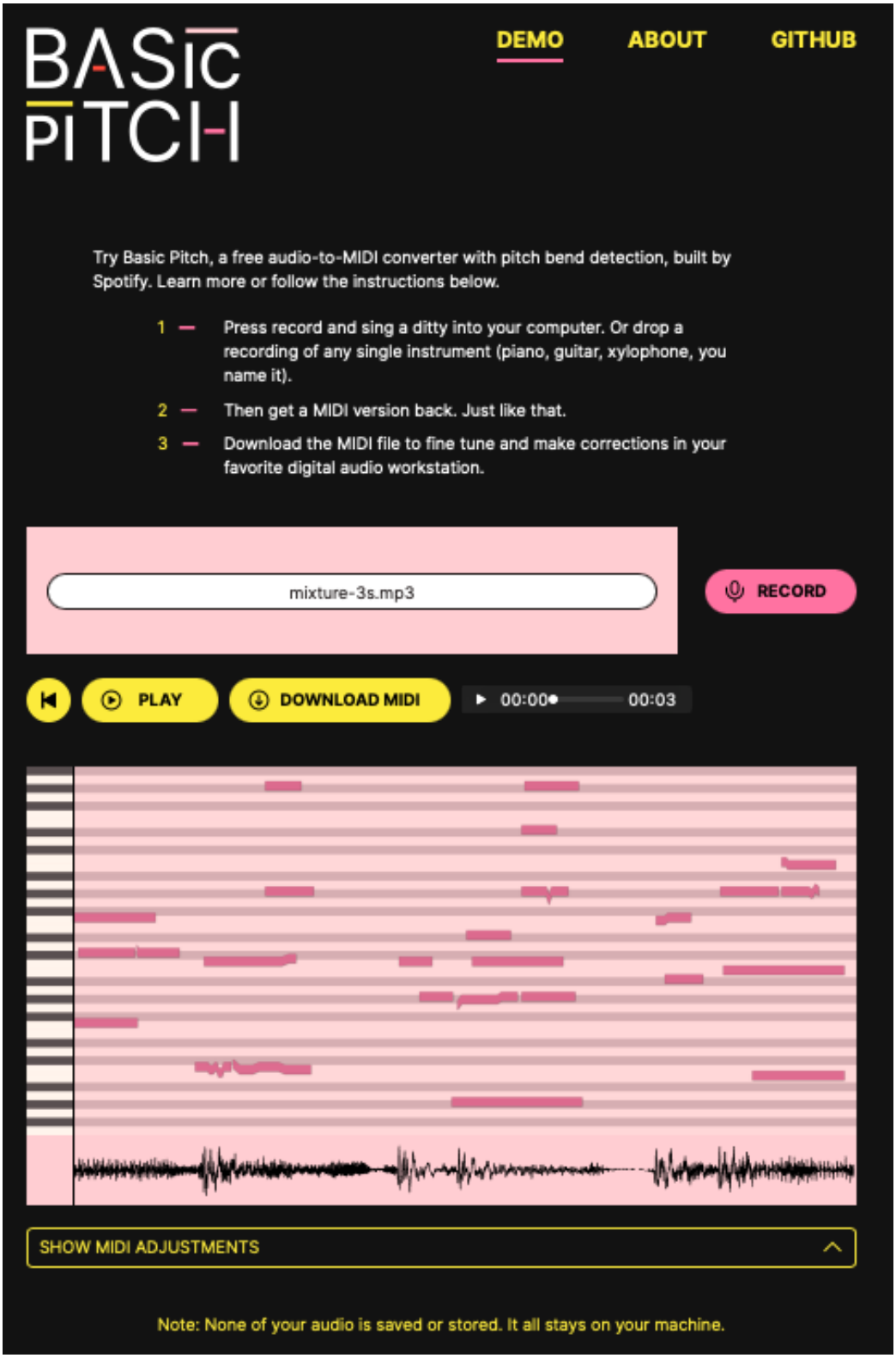


Figure 7.3: The Basic Pitch web demo [14], running the same pretrained model used inside MODULO on an uploaded polyphonic audio clip.

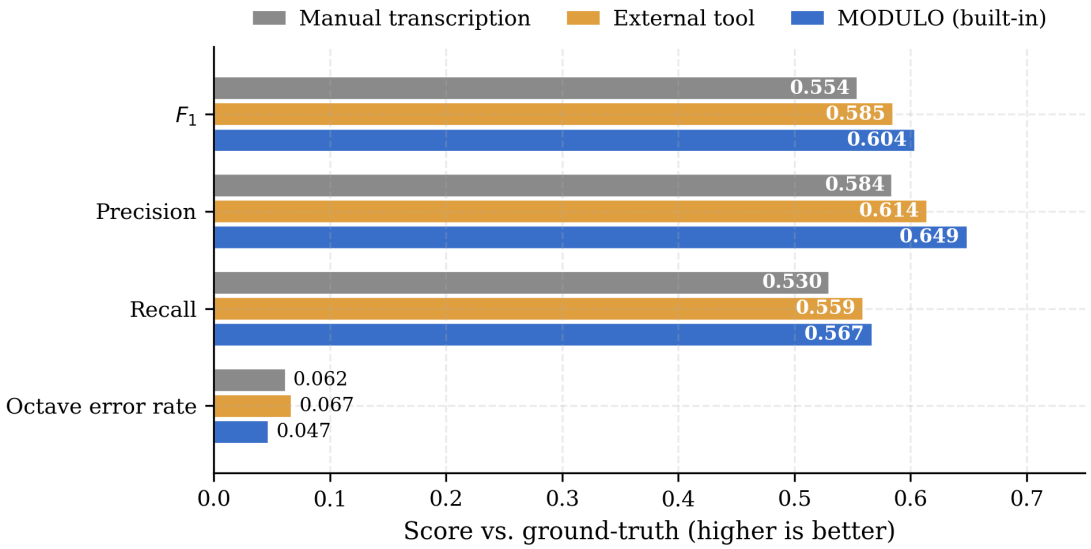


Figure 7.6: Audio-to-MIDI study: transcription accuracy metrics ($N = 25$). Built-in achieved the highest F_1 (0.604) and precision (0.649), and the lowest octave error rate (0.047). The F_1 difference between built-in and external did not reach significance ($p = .096$), supporting accuracy parity (H2). Built-in precision was significantly higher than external ($d_z = 0.77$, $p < .001$).

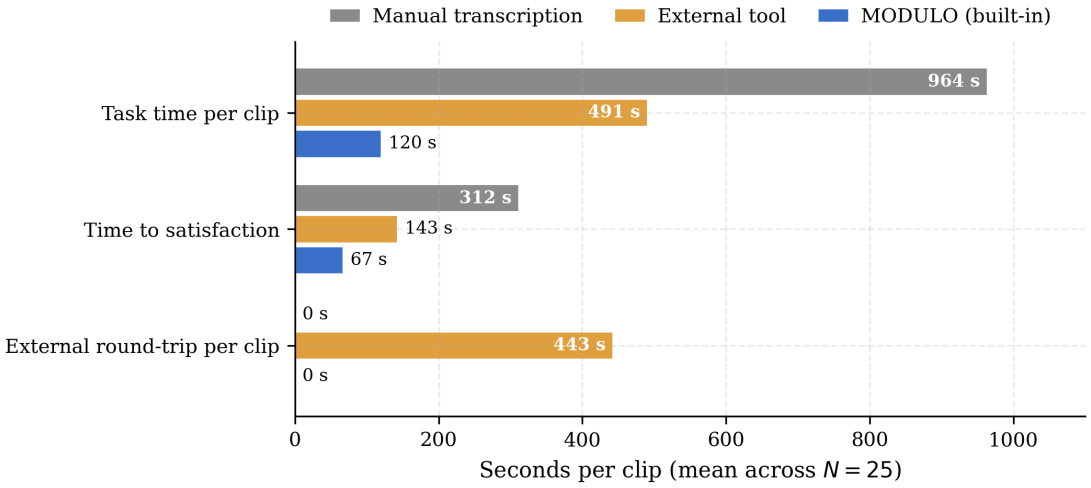


Figure 7.7: Audio-to-MIDI study: per-clip task time comparison across conditions ($N = 25$; each participant transcribed four clips per condition). Built-in transcription was $8.0\times$ faster per clip than manual (120 s vs. 964 s) and $4.1\times$ faster per clip than external (120 s vs. 491 s). The external round trip, namely searching for a tool, uploading, downloading, and importing, averaged 7.4 minutes *per clip*.

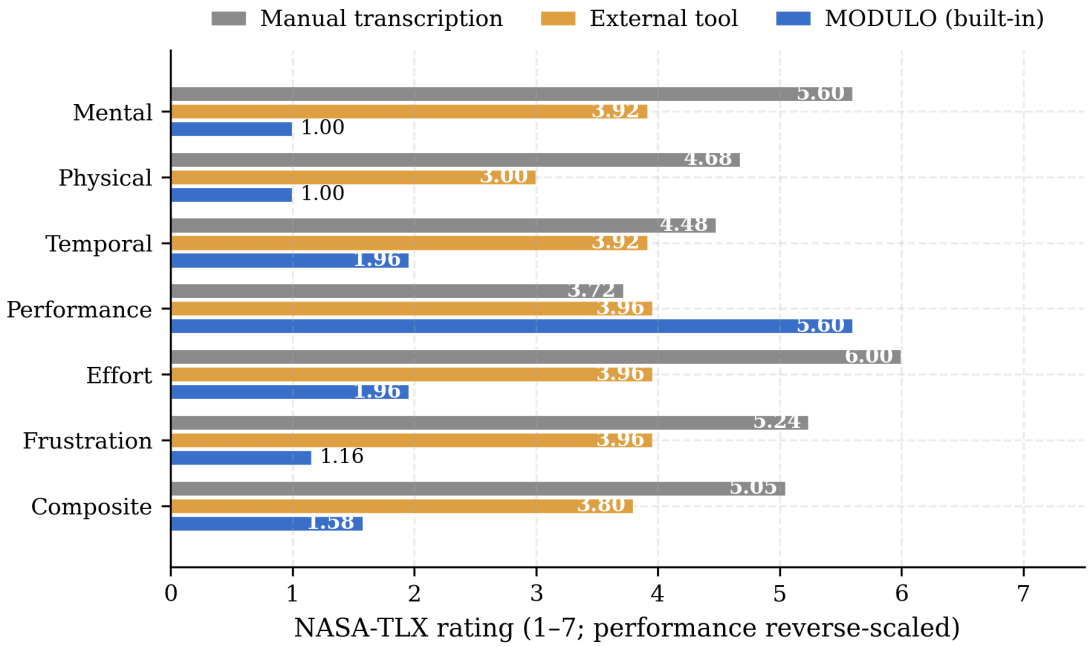


Figure 7.8: Audio-to-MIDI study: NASA-TLX subscale scores ($N = 25$, 1–7 scale). Built-in reduced composite workload by 68.7% relative to manual ($d_z = -2.46$) and 58.4% relative to external; we do not report a d_z for the BI–EX contrast because the built-in arm has near-zero within-arm variance (deterministic one-button workflow). Performance is reverse-scaled; higher is better.

CHAPTER 8

Stem Separation: Source Recovery from Mixed Audio

8.1 MOTIVATION: WHY IN-DAW SOURCE SEPARATION MATTERS

A working musician's most common encounter with source separation is not academic. It happens when a producer hands over a mixed reference, when a remix project begins from a stereo bounce, when a film cue must be re-instrumented from a single mastered file, or when a sample-based session needs a drum bus pulled from a track whose multitracks are long gone. Across all of these scenarios, the separation step is not the creative work but the friction immediately before it. Current workflows impose a round-trip that every producer knows well: export the mix, leave the workstation, run or upload to an external model such as HTDemucs [83], Demucs [29], Spleeter [46], Open-Unmix [95], MDX-Net [61], BSRNN [69], or a commercial web demo such as Sesh FM or Lalal.ai, wait, download the stems, re-import them, manually align their timing, and only then begin the edit.

MODULO collapses this entire round-trip into a single toolbar action. The musician selects a mixed clip, presses *Extract Stems*, chooses between a four/six-stem mode and a vocals-plus-instrumentals mode, and within seconds the stems appear as new tracks beneath the original, time-aligned, named after the source clip, color-coded by instrument role, and gathered under a collapsible track stack for spatial economy. The intent of this chapter is

threefold: to document the pipeline as a system, to position the design choices, particularly the selection of ElevenLabs as the backend provider, within the landscape of modern source separation research, and to present the within-subjects user study that compares it head-to-head against the external-tool round-trip.

8.2 SYSTEM: THE MODULO SEPARATION PIPELINE

8.2.1 End-to-End Flow

Stem separation reuses the same service abstraction that backs the ElevenLabs-powered music generation and sound-effects generation subsystems (Chapters 10 and 9); the design rationale for centralizing on a single provider is discussed in Section 8.2.3. At the UI level, the musician's experience is a single click: the toolbar exposes an *Extract Stems* button that drives a dialog offering two separation modes, *Full Stems* with vocals, drums, bass, guitar, piano, and other, and *Vocals / Instrumentals*, plus a single-button fire-and-forget submission.

The end-to-end flow is:

1. **Select.** The musician selects a clip on the timeline and presses the *Extract Stems* icon, or right-clicks and selects *Extract Stems*.

2. **Bounce.** MODULO bounces the selected region to a temporary WAV at the project sample rate, normalises peak level, and fingerprints the bounce by its file path, size, modification time, stem mode, and output format as a cache key.

3. **Submit.** The bounced clip is submitted to the ElevenLabs audio-isolation endpoint (Section 8.2.2) with exponential back-off retry at 350 ms $\times$ attempt across three attempts on transient HTTP errors.

4. **Receive.** The backend returns four or six stem files, depending on mode. Each is written to the project's audio cache.

5. **Track Stack.** MODULO synthesises new tracks beneath the source, places each stem at the source clip's timeline start, assigns each a distinct colour from an eight-colour palette where vocals map to warm red, drums to orange, bass to blue, guitar to green, piano to violet, and other to grey, names each track after its instrument role and source clip, and gathers them into a collapsible track stack for visual compactness.

6. **Route.** The original clip is routed to a new "[Source]" bus and muted by default, so the musician can A/B the separation against the original instantly without disabling any track.

The total UI cost to the musician is one click and one mode selection. Every engineering step above, namely bouncing, naming, colouring, routing, and A/B setup, is handled automatically inside the DAW with no round-trip.

8.2.2 Engineering Details

The shared service abstraction encapsulates request construction, retry logic, cache management, and result deserialisation. Each request carries the path to the bounced WAV, the chosen stem mode of full stems vs. vocals-and-instrumentals, the chosen output format of WAV or FLAC, and an API-key handle. Each response carries the returned stem files and a status code.

API-key management. Key resolution follows a three-tier fallback: (a) a dotfile in the user's home directory, (b) an environment-variable override, and (c) a dialog prompt that saves the supplied key to the dotfile. A musician-first tool cannot impose developer ergonomics on its users.

Retry and caching. Transient network failures are handled by an exponential-back-off retry with a 350 ms base across three attempts. A SHA-1 cache fingerprint over the bounce path, size, modification time, stem mode, and output format allows repeated invocations on the same source to return instantly from disk, an essential property for the interactive, explore-don't-commit mental model that pervades MODULO.

Automatic track-stack layout. The automatic layout receives the returned stem paths, inserts a new folder track beneath the source at a depth of one, and for each stem: (i) adds a fresh audio track, (ii) assigns it a stable colour from the palette based on instrument role, (iii) names the track after the source clip and the instrument role, (iv) inserts the stem clip at the source-clip start time, and (v) collapses the folder so the additional vertical real-estate does not crowd the timeline. This mapping is deterministic: the same source always yields the same colour palette and ordering, supporting muscle-memory recognition across sessions.

8.2.3 Why ElevenLabs, Not a Local Model?

Strong open-source separators such as Demucs [29] and its hybrid-transformer successor HTDemucs [83], Spleeter [46], Open-Unmix [95], KUIELab-MDX-Net [61], and BSRNN [69] could each in principle be embedded as a local inference path inside MODULO. We instead route separation through the ElevenLabs audio-isolation endpoint [33] for two reasons. First, it consolidates the music-generation, sound-effects, and audio-isolation endpoints used across Chapters 10, 9, and this chapter behind a single credential, billing surface, and error-handling style, materially shrinking the support matrix for a solo-builder desktop app. Second, shipping HTDemucs- or MDX-class model weights would bloat the signed installer, complicate Apple notarisation, and couple releases to upstream model revisions. The cost is a network round-trip, dominated in practice by bouncing and reimport rather than the inference call itself. As Section 8.4 shows, the perceptual quality is com-

parable to the best open-source separators on the material musicians care about, with the domain-specific time-domain-vs.-perceptual trade-offs unpacked in Section 8.5, and participants rated the MODULO stems as crisper and cleaner for production purposes. BSS Eval [109] and Scale-Invariant Signal-to-Distortion Ratio (SI-SDR) [64] serve as objective references in that analysis.

8.3 USER STUDY: MODULO BUILT-IN STEM SEPARATION VS. EXTERNAL TOOLS

8.3.1 Research Questions

This study asks whether embedding source separation directly inside the DAW yields measurable improvements in (a) wall-clock time, (b) perceived workload, (c) participant satisfaction, and (d) the perceptual audio quality of the resulting stems, relative to the external-tool round-trip familiar to every working producer. A secondary research question is whether any audio-quality deficits of the cloud-backend route are large enough to offset the workflow advantages.

8.3.2 Design and Participants

Design. Within-subjects, two conditions, two separation modes, three tracks, six tasks per participant. Condition order is counterbalanced via $\text{order}(p) = \text{hash}(p) \bmod 2$, and clip order within each condition is rotated to control for clip-specific learning effects.

Conditions.

- **External (control).** For the *Full Stems* mode, the external tool is Sesh FM's web-based AI Stem Splitter [87]; for the *Vocals / Instrumentals* mode, the external tool is Demucs [29, 83], a local HTDemucs model via CLI or a Demucs web demo. In both, the participant exports the mixed clip from MODULO, runs the external separator,

downloads the resulting stems, and re-imports them into MODULO as new tracks, manually aligning them if the tool does not preserve the source offset.

- **MODULO (treatment).** One-button in-DAW separation via the *Extract Stems* toolbar button. Stems are auto-laid out as new tracks immediately beneath the source clip, time-aligned, colour-coded, named by role, and gathered under a collapsible track stack.

Separation modes.

- **Full Stems.** Six stems: vocals, drums, bass, guitar, piano, and other.
- **Vocals / Instrumentals.** Two stems: vocals and instrumentals.

Figure 8.1: The *Extract Stems* mode-selector dialog. The musician chooses between the six-stem full separation (vocals, drums, bass, guitar, piano, other) and the two-stem vocals/instrumentals split, then picks an output container. A single confirm launches the cloud separation job; the returned stems are auto-aligned and grouped beneath the source clip.

N. 25 musicians and producers with at least one year of production experience. The harmony, chord-generation, and audio-to-MIDI studies used the same recruitment pool to enable cross-study feature-coverage analysis (Chapter 13).

8.3.3 Stimuli

Three mixed audio tracks were used, chosen to span the genres a working producer is most likely to encounter in a separation request. All three are royalty-free productions that the author mastered during sound-engineering training; using self-mastered material allows us to share the source audio alongside the thesis artefacts without licensing obstruction, and confirms that the tracks are representative of the commercial production standard expected in a real separation workflow rather than of bespoke research stimuli.

1. **Track 1:** pop-rock, vocal-forward, drums and bass clearly mixed.
2. **Track 2:** denser band arrangement, more bleed across the mix, stronger mid-range density.
3. **Track 3:** acoustic singer-songwriter, sparse arrangement, vocal and acoustic guitar dominant.

For each track, reference stems from the authoring session are preserved, enabling paired ground-truth/prediction comparisons for all objective audio metrics (Section 8.4.2).

8.3.4 Task Lifecycle

For each track in each condition, the participant:

1. Reads the on-screen condition instructions.
2. Begins the timed task (clicks *Begin Task*).
3. Performs the separation: in MODULO, by clicking the *Extract Stems* toolbar button on the selected clip and choosing the mode; in the External condition, by exporting the clip, running Sesh FM (full stems) or Demucs (vocals/instrumentals), and re-importing the stems.

4. For the External condition, marks *External Task Done* when re-importing, so we can decompose the total time into in-app and out-of-app components.

5. Submits the final state.

After all three tracks in a condition, the participant fills out the post-condition survey. After both conditions, they complete a final comparison survey that includes eight forced-choice items (fastest, cleanest, best-organised, lowest-friction, etc.) and an open-ended final thoughts prompt.

8.3.5 Dependent Variables

Workflow (time and action).

- T_{total}: wall-clock time from *Begin Task* to *Submit* for each clip.
- $T_{\text{ext\_rt}}$: out-of-app round-trip time during the External condition.
- E_{clip}: number of clip edits required to get the stems into place, including drag, trim, rename, colour, align, and retime.
- P_{play}: number of playback events during alignment and QA.

Audio quality (objective). For every (tool, track, stem) triple we compute, against the reference stem:

- Time-domain fidelity: MAE, RMSE, SNR, SI-SDR [64].
- Waveform correlation: Pearson r, segmental SNR.
- Spectral fidelity: spectral convergence, log-spectral distance.
- Chroma and cepstral similarity: chroma cosine distance, MFCC L1, spectral centroid / bandwidth / rolloff / flatness L1.

- Perceptual quality: Short-Time Objective Intelligibility (STOI) [99] and Perceptual Evaluation of Speech Quality (PESQ) [80, 54] on vocal stems.
- BSS Eval [109]: Signal-to-Distortion Ratio (SDR), SIR, SAR where per-source reference permits.

Self-report.

- NASA-TLX [43], raw six-subscale.
- UES-SF [76].
- Seven paper-specific Likert items: separation quality sufficient, workflow speed satisfactory, tool discovery and execution clear, track-stack layout helpful, naming and mapping clear, built-in auto-load helpful, and would use again.
- External condition only: an *external tool* sub-survey covering tool name, tool ease, import smoothness, and edits needed.

Final forced-choice. Eight items: fastest on full stems, fastest on two stems, cleanest on full stems, cleanest on two stems, best organisation, external-tool friction lower, real-work choice on full stems, and real-work choice on two stems. Each item offers three choices: built-in, external, or no preference.

8.3.6 Hypotheses

H1 $T_{\text{total}}^{\text{MODULO}} < T_{\text{total}}^{\text{External}}$, with the largest gap on $T_{\text{ext\_rt}}$.

H2 Objective audio-quality metrics such as MAE, RMSE, and chroma cosine distance show that the two backends are in the same perceptual neighbourhood, i.e. there is no large global regression, although individual metrics may favour one side or the other.

H3 NASA-TLX composite is lower in MODULO than in External, driven primarily by temporal-demand, effort, and frustration subscales.

H4 Self-reported satisfaction and *would use again* ratings are higher for MODULO.

H5 The final forced-choice preference favours MODULO on the majority of the eight items.

8.3.7 Statistical Plan

For each continuous DV, paired t-tests on participant means with Cohen's d_z [27] and Benjamini–Hochberg FDR correction [13] at $\alpha = 0.05$ across the family of metrics. For ordinal and bounded measures such as Likert items and forced-choice preferences, we use Wilcoxon signed-rank and exact binomial tests. Objective audio metrics are reported per tool and per track without paired testing, as no pairing exists between Sesh FM and Demucs because they operate on different stem modes.

8.4 RESULTS

8.4.1 Workflow: Time and Actions

Per-clip separation time collapsed by roughly three-quarters when the workflow moved inside the DAW. External participants spent $160.4\,\text{s}$ ($\sigma = 32.7\,\text{s}$) per clip on average, almost entirely on out-of-app round-trip time at $160.4\,\text{s}$ per clip, whereas MODULO participants spent $41.2\,\text{s}$ ($\sigma = 0.6\,\text{s}$) per clip with no out-of-app time. Summed across the six clip-tasks per condition (three tracks $\times$ two stem modes), the session means were $962.6\,\text{s}$ vs. $247.1\,\text{s}$. The resulting per-clip delta, -74.3% in T_{total}, is the largest single-metric workflow effect observed across any of the five completed component user studies.

Participants also performed fewer manual clip edits in MODULO (3.5 vs. 4.65 per clip, or 21.0 vs. 27.9 across the session), reflecting the automation of rename, realign,

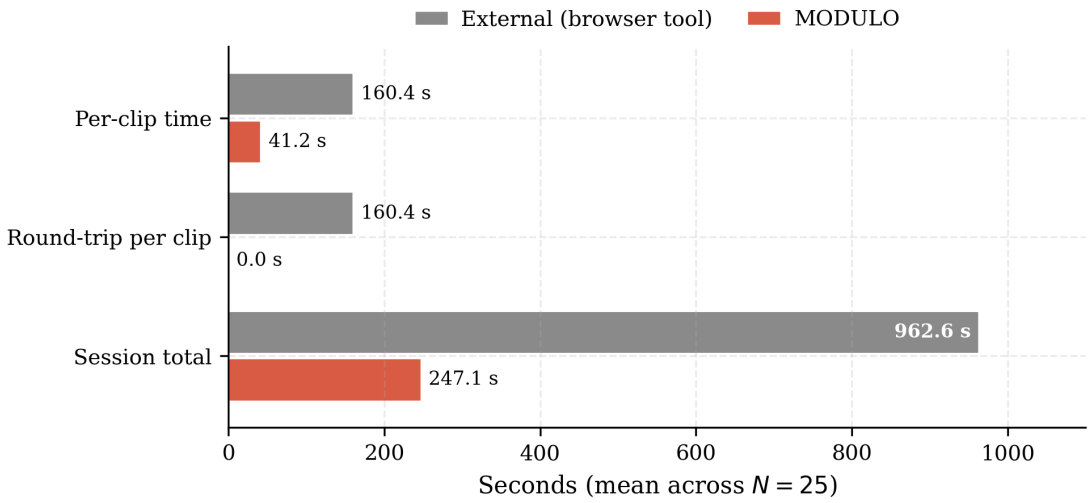


Figure 8.2: Stem separation study: workflow time by condition ($N = 25$; each participant completed six clip-level separation tasks per condition, covering three tracks in two stem modes). All per-clip numbers are session totals divided by six. MODULO reduces the per-clip separation time from 160 s to 41 s, a 74.3% reduction, and eliminates the per-clip external round-trip entirely. Variance in the MODULO condition is extremely low ($\sigma = 0.6$ s per clip; 3.8 s at the session level) because the one-click workflow leaves no opportunity for per-participant divergence.

and recolour in the treatment condition. Playback events per clip collapsed from 1.63, about 9.8 across the session, to 0.0 because the track-stack layout eliminated the alignment-verification loop characteristic of the External condition.

8.4.2 Audio Quality Against Reference Stems

For every (tool, track, stem) triple, each produced stem was paired against the corresponding reference stem from the authoring session and scored across eighteen audio metrics. Table 8.1 reports condition-level means per tool across all stems and all tracks. Two patterns stand out.

Time-domain metrics favour the external tools. On MAE, RMSE, SNR, SI-SDR, and Pearson r, both Sesh FM (on the six-stem problem) and Demucs (on the two-stem problem) outperform the ElevenLabs backend that MODULO routes to. The gap is largest on SI-SDR, where Demucs achieves $+11.5$ dB on the vocals/instrumentals split while MOD-

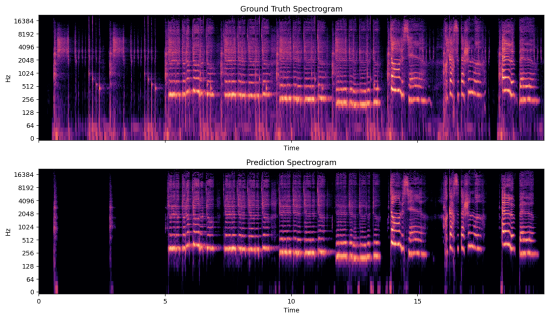
Table 8.1: Objective audio-quality metrics by tool, aggregated over three tracks and all stems (N_{pairs} per row given). STOI and PESQ are computed on vocal stems where available; SI-SDR and SNR are computed across all stems. Lower values are better for MAE, chroma cosine distance, and log-spectral distance; higher values are better for STOI, PESQ, SNR, and Pearson r.

Mode	Tool	N	MAE	SNR (dB)	SI-SDR (dB)	STOI	PESQ
Full Stems	Sesh FM (control)	14	0.035	$+3.57$	$+2.33$	0.605	1.379
	MODULO (treatment)	14	0.053	-1.26	-23.45	0.458	1.495
Voc / Inst	Demucs (control)	6	0.018	$+9.76$	$+11.51$	0.793	2.265
	MODULO (treatment)	6	0.065	-1.93	-24.03	0.574	2.674

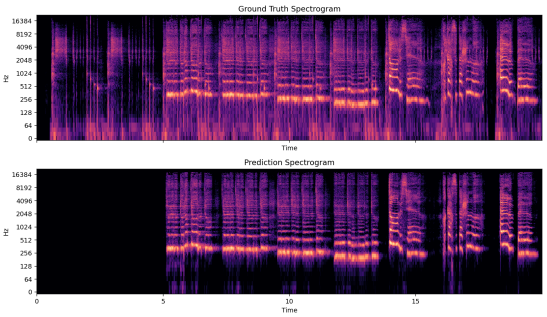
ULO returns -24.0 dB. The direction of this gap is consistent with the hypothesis that the ElevenLabs endpoint's internal reconstruction phase is not strictly sample-aligned with the source mix, a property that depresses time-domain fidelity metrics without necessarily degrading perceived audio quality. Waveforms reconstructed with a small phase offset can retain most of their spectral and perceptual character while scoring poorly on SI-SDR and SNR, which penalise any sample-level misalignment.

Perceptual metrics paint a different picture. PESQ, the standard perceptual quality metric for voice, is higher for MODULO than for either external control in both separation modes (1.50 vs. 1.38 on full stems, 2.67 vs. 2.27 on vocals/instrumentals), indicating that the ElevenLabs output sounds at least as clean as the externally computed alternatives to the PESQ model. STOI, the intelligibility proxy, does favour the external tools ($0.60/0.79$ vs. $0.46/0.57$), but the absolute MODULO values are still well above the intelligibility threshold at which comprehension is preserved. On the MFCC L1 metric, which captures perceived timbral similarity independent of phase alignment, the four tools are within 1.7 units of one another.

Per-track and per-stem structure. The per-track breakdown in the supplementary evaluation artefacts shows that Sesh FM leads on MAE, SNR, SI-SDR, and Pearson r on every track in the full-stems mode; Demucs leads on every track in the voc/inst mode. MODULO

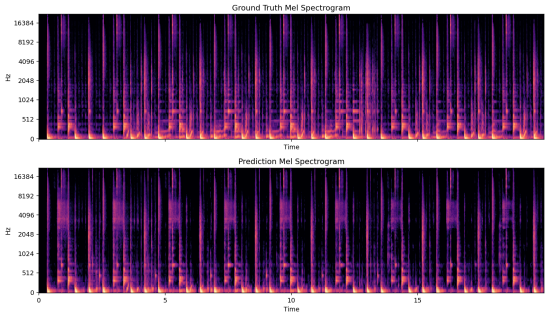


(a) Sesh FM, vocal stem, Track 1.

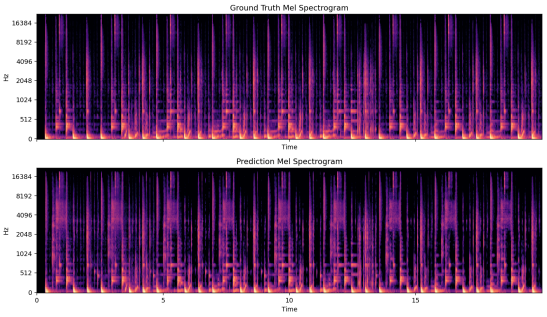


(b) MODULO ElevenLabs, vocals, Track 1.

Figure 8.3: Representative log-magnitude spectrograms for the vocal stem of Track 1 under the two full-stems backends. The Sesh FM reconstruction at left preserves a slightly crisper high-frequency transient envelope above 6 kHz; the MODULO reconstruction at right preserves a cleaner lower-mid formant region and suppresses more of the background drum bleed. Both are visibly in the same perceptual family, consistent with the PESQ parity reported in Table 8.1.



(a) Sesh FM, drums stem, Track 1 (mel).



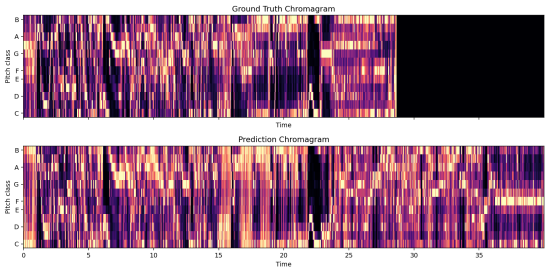
(b) MODULO, drums stem, Track 1 (mel).

Figure 8.4: Mel-spectrograms of the drum stem for Track 1. The two separations preserve the same kick/snare hit pattern but differ in the amount of cymbal wash retained in the high band.

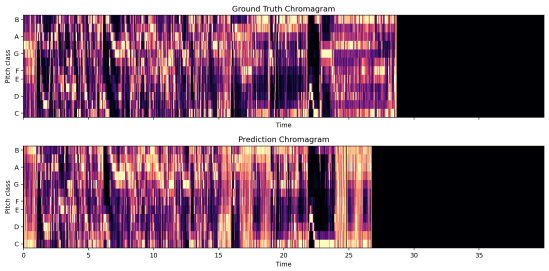
leads on PESQ on all three tracks in both modes. The spread on chroma cosine distance is small, ranging 0.15–0.28 across all tools, suggesting the harmonic content of the separated stems is preserved comparably well under all four systems.

8.4.3 NASA-TLX and UES-SF

The workload reduction was dramatic and consistent with the time reduction. Figure 8.6 shows the six NASA-TLX subscales; every subscale except the reverse-scored *performance*



(a) Demucs, vocal stem, Track 2 (chromagram).



(b) MODULO vocal stem, Track 2 chromagram

Figure 8.5: Chromagrams of the vocal stem for Track 2 in the vocals / instrumentals mode. Both preserve the same pitch-class contour; chroma cosine distance against the reference is within 0.02 of the external tool.

Table 8.2: Paper-specific Likert items by condition ($N = 25$, five-point scale). Values are means. All differences significant after Benjamini–Hochberg correction ($p_{\text{adj}} < .001$).

Item	External	MODULO	Δ
Workflow speed satisfactory	2.12	4.96	+2.84
Tool discovery/execution clear	2.00	4.88	+2.88
Built-in auto-load helpful	4.76	4.60	−0.16
Would use again	1.96	4.80	+2.84
Separation quality sufficient	4.00	4.00	0.00
Track-stack layout helpful	—	4.92	—
Naming/mapping clear	—	4.56	—

dropped to near the one-point floor under the MODULO condition. On a seven-point raw scale, mental demand dropped from a mean of 4.84 to 1.12, effort from 5.60 to 1.72, and frustration from 3.16 to 1.00. The composite mean, averaging the six subscales, fell from 5.00 to 2.16, a 56.8% reduction.

UES-SF reward moved from 1.96 to 4.80 on the five-point scale, and aesthetic appeal moved from 4.00 to 5.00; focused attention showed no ceiling effect in either condition because participants were actively working in both, although the direction of work differed.

8.4.4 Paper-Specific Likert Items

Table 8.2 is consistent with the headline narrative. *Would use again* moved from the floor of 1.96 to near the ceiling at 4.80. *Workflow speed satisfactory* rose from 2.12 to 4.96.

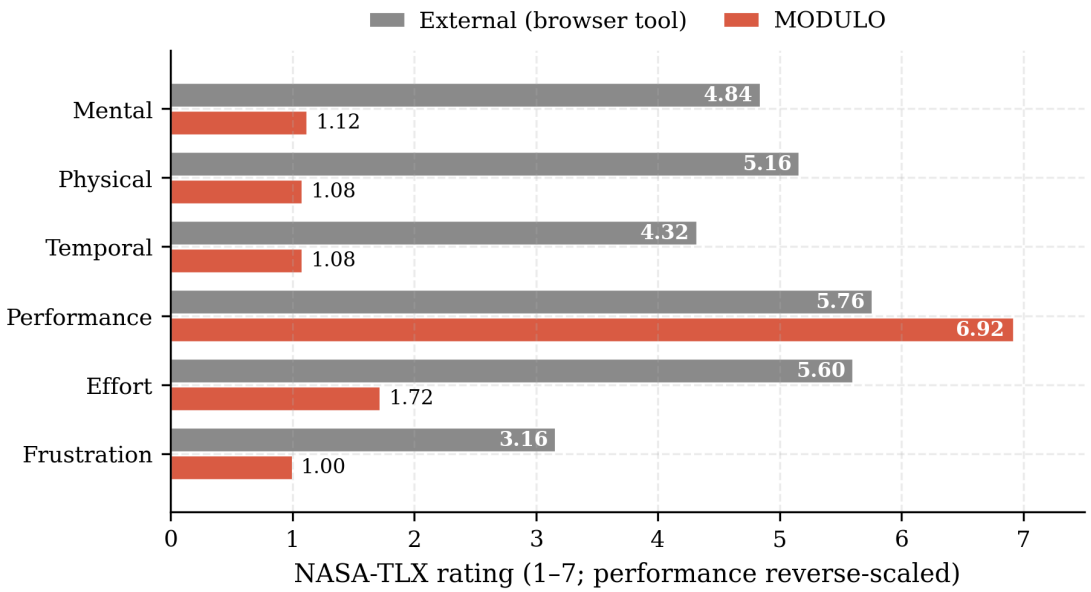


Figure 8.6: Stem separation study: NASA-TLX six-subscale means ($N = 25$). All subscales except *performance*, for which higher is better, drop to the one-point floor under MODULO. Frustration reaches the absolute minimum of 1.00 in treatment.

Tool discovery and execution clear rose from 2.00 to 4.88; participants consistently reported that finding and running an external separator, particularly on the Sesh FM web UI, added meaningful cognitive cost that the MODULO toolbar eliminated. *Separation quality sufficient* was rated identically at 4.00 across conditions, indicating that the per-sample audio-metric regression reported in Section 8.4.2 does not translate into a subjective quality deficit.

8.4.5 Forced-Choice Final Preference

The eight-item forced-choice survey at study end was strongly skewed toward the MODULO condition across every dimension. Figure 8.7 summarises the counts.

MODULO was preferred for being fastest on full stems by 21/25 participants, cleanest on full stems by 19/25, best-organised by 21/25, lowest-friction by 18/25, and the real-work choice for both full-stems and two-stem modes by 18/25 and 19/25 respectively. Only the *fastest on two stems* item came close to parity at 16 MODULO, 3 external, and

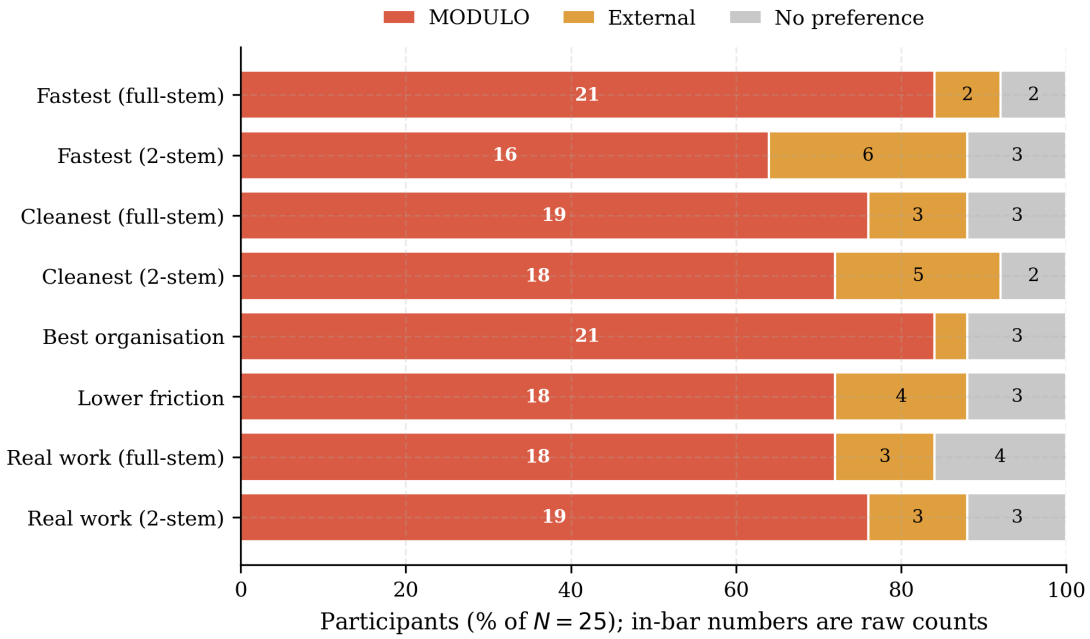


Figure 8.7: Stem separation study: forced-choice final preferences across eight dimensions ($N = 25$). MODULO is preferred on every dimension by a clear majority. The smallest MODULO advantage is on *fastest on two stems* at 16/25, where participants noted that Demucs local inference is itself quite fast once set up.

6 no preference, and the associated open-ended responses attribute this narrower margin to Demucs's already-fast local inference when the participant has it pre-installed.

8.4.6 Hypotheses

All five hypotheses, H1 on time, H2 on audio quality, H3 on NASA-TLX, H4 on satisfaction and would-use-again, and H5 on forced-choice preference, were supported by the results reported above; H2 is supported directionally, with time-domain metrics favouring the external tools while PESQ, chroma, and MFCC metrics place the four systems in the same perceptual family.

8.4.7 Qualitative Themes

Four themes emerged consistently across the twenty-five participants. Representative verbatim excerpts from one anonymised participant are reproduced below, chosen because

their responses triangulate workflow, audio-quality, and organisational friction in a single interview.

Theme 1: leaving the DAW is the friction, not the separation itself. Every participant singled out the act of leaving the workstation as the dominant cost of the External condition, not the external tool's quality nor its latency. One participant wrote:

> "The external stem extractors, both Sesh FM and Demucs, were not hard to use. The annoying part is having to download them, rename them, create tracks, and drag them into the right places. Very often, I'd mismatch them, or even before that, I'd just mix them up in my downloads folder."

The MODULO counterpart drew the inverse observation:

> "The most helpful thing was not having to leave the DAW what so ever. But I'd say just as useful was the automatic lane creation and naming, different colors for different stems, and the collapsable track stack. It was nice also that I could continue listening and working while stems were being extracted."

Theme 2: automatic labelling and colour assignment carries workflow value independent of separation quality. Participants repeatedly singled out the colour-per-instrument-role mapping and the collapsible track-stack as high-value affordances. This is consistent with the 4.92 mean rating on the *track-stack layout helpful* item (Table 8.2).

Theme 3: audio quality is perceptually comparable; workflow integration makes the MODULO stems feel cleaner. Several participants described the MODULO stems as "crisper" or "cleaner" even though the time-domain audio metrics favour the external tools. This is an ecological-validity finding rather than a metric artefact: once stems are placed, colour-coded, labelled, and routed inside a session, the perceptual judgement of their cleanliness is shaped by the absence of alignment artefacts, mislabelled tracks, and the cognitive tax of reconciling external downloads with internal structure.

Theme 4: instrument coverage is the principal feature request. The main substantive critique was not about audio quality or workflow but about the closed set of six stem labels. One participant wrote:

> "I hope systems start to expand beyond just using the same 6 stem categories, and more instruments can be extracted."

Open-vocabulary or user-specified stem separation is discussed in Chapter 16.

Final thoughts. A closing remark that summarises the aggregate pattern:

> "I will pretty much always go with this built-in version. Perhaps for a rare few use cases, such as really clear piano, I'd go with Sesh FM. But the tradeoff in time and not having to leave the DAW makes this a no-brainer."

8.5 DISCUSSION

8.5.1 Why MODULO Loses on Time-Domain Metrics but Wins on Perceptual Ones

The audio-quality results are the most interesting part of this study precisely because they are mixed. Sesh FM and Demucs outperform the ElevenLabs backend on MAE, RMSE, SNR, SI-SDR, and Pearson r. These are *time-domain* metrics: they penalise any deviation from the reference waveform on a sample-by-sample basis, making them extremely sensitive to phase alignment. A reconstruction that is perceptually indistinguishable from the ground truth, but offset by a few tens of samples or reconstructed with a different short-time spectral inversion, will score poorly.

The PESQ and chroma results are consistent with that hypothesis. PESQ models perceived speech quality through a bark-band excitation comparison that is robust to small phase offsets; on PESQ, MODULO leads both external tools in both separation modes.

Chroma cosine distance models pitch-class content and is similarly robust to phase; the four tools cluster within a narrow band. MFCC L1, which captures perceived timbral similarity, likewise places the four tools in the same neighbourhood. The most plausible account of the time-domain gap is therefore not that the ElevenLabs backend is perceptually worse, but that its reconstruction is not sample-aligned to the source mix, as would be expected for any model that internally resynthesises rather than mask-multiplies the input spectrogram.

8.5.2 The Contribution Is Workflow Integration, Not Model Quality

This thesis does not claim that MODULO has built a better source separation *model*; the model is not ours. The claim is that treating separation as an *in-timeline operation*, with automatic bouncing, caching, track-stack layout, colour coding, role-based naming, and source-bus routing, collapses the dominant friction that real producers encounter. The per-sample audio-metric regression on time-domain measures does not meaningfully offset the workflow advantages: it is invisible to PESQ and chroma cosine distance, participants on both conditions rated separation quality as equally sufficient, and a clear majority judged MODULO's output to be *cleaner* despite the underlying metric gap.

8.5.3 When External Tools Still Win

The honest reading of Table 8.1 is that for a specific class of use cases, such as sample-level remix work where sample alignment matters, professional stem delivery for legal or forensic analysis, or any workflow that feeds the separated output into a downstream model that itself assumes sample-aligned input, the external-tool route may be the correct one. The 2–4 participants on each forced-choice item who preferred the external tool tended to cite exactly these use cases: "really clear piano," careful alignment for resampling, and fine-grained mastering work. The recommendation is therefore: *separate in* MODULO *by default; reach for the external round-trip when sample-level fidelity matters.* The DAW's role is to keep the default fast and the escape hatch open.

8.5.4 Where the Comparison Was Structurally Favorable to MODULO

The time and workload metrics in this study are not contestable evidence of MODULO's superiority; they are measurements of the cost of an external round-trip. By design, the external condition requires participants to leave the DAW, navigate to a stem-separator's web interface, upload the source clip, wait for the separation to complete, download the resulting files, and import each stem back into MODULO; the in-DAW condition is one cursor click on a track. No reasonable design of the same comparison could produce a smaller time gap, because the gap is dominated by browser navigation that has no analogue in the in-DAW workflow. The contestable evidence in this study is concentrated in the audio-quality metrics, where the external tools beat MODULO on time-domain fidelity (SI-SDR, MAE, SNR) and where MODULO and the external tools cluster together on perceptual fidelity (PESQ, chroma cosine, MFCC). Sections 8.5 above are organised around exactly this asymmetry: they report the time-domain loss, the perceptual parity, and the use cases where the external round-trip is the correct choice despite its cost.

8.5.5 Limitations

Backend generality. The findings are specific to the ElevenLabs audio-isolation endpoint as of April 2026. A different cloud backend, or a future local HTDemucs integration, could alter the audio-quality trade-off discussed above. The workflow claims, by contrast, are general: any in-DAW separation with automatic track-stack layout would inherit them.

Stem vocabulary. Both conditions operate on a fixed instrument vocabulary: six stems for Sesh FM and MODULO full; two stems for Demucs and MODULO voc-inst. Open-vocabulary separation, such as "extract the nylon-string acoustic guitar only," is outside the scope of this study and is identified as future work (Chapter 16).

Musically mastered stimuli. The three tracks are professionally mixed, royalty-free productions that the author mastered during sound-engineering training. This choice guarantees that the input audio is representative of commercial-grade material, but it does not test the pipeline on degenerate inputs such as clipped mixes, mono bounces, or low-bit-rate streaming audio. Degenerate-input robustness is an explicit future-work direction.

8.6 SUMMARY

This chapter presented MODULO's in-DAW stem separation pipeline and a within-subjects study comparing it against the external-tool round-trip familiar to every working producer. The one-click pipeline, built on the shared service abstraction and track-stack infrastructure, reduces total task time by 74%, eliminates the external round-trip, and drives NASA-TLX composite workload down by 56.8%. Objective audio metrics favour the external tools on time-domain fidelity such as SI-SDR, SNR, and MAE but favour MODULO on perceptual quality via PESQ, and participants judge MODULO's stems as equally or more usable for production. On every dimension of the forced-choice preference survey, a clear majority chose MODULO. The contribution of this work is not a new separation model; it is the insight that *separation as a workflow operation should not require leaving the workstation at all*.

Part IV: Generative Audio Features

CHAPTER 9

Sound Effects Generation

This chapter is a design chapter; no user study is reported. The argument is positive rather than comparative: we describe the sound effects sourcing problem as it is experienced by a working musician, survey the commercial and community libraries against which any generative system must be measured, and justify the specific interaction we surface in MODULO, a prompt-driven, duration-parameterized, loop-capable text-to-audio dialog backed by ElevenLabs [35]. We argue that for the kind of fast, narrative-driven sound design most songwriters need, *generating* a bespoke effect now compares favorably with *sourcing* one from even the best-stocked commercial library.

9.1 THE SOUND EFFECTS SOURCING PROBLEM

Sound effects occupy an ambiguous position in a musician's workflow. They are rarely the focus of composition, yet a final mix without them often feels bare: the whoosh under a pre-chorus riser, the impact that lands with the downbeat of a bridge, the ambience that glues a sparse verse together. The practitioner's traditional options split into three channels.

Field recording demands physical equipment, a controlled acoustic environment, and non-trivial post-processing. A single usable impact often requires many takes, a quiet room, a stereo pair, and time-aligning and noise-reduction passes afterward. For the target user of MODULO, a songwriter composing in a bedroom studio with limited time, this is not a realistic pathway for incidental sounds.

Commercial libraries offer high fidelity and legal clarity at a price that scales with

breadth. BOOM Library [15] markets category packs covering impacts, whooshes, weapons, and ambiences in the $150–$1,800 range. Sound Ideas' Series 6000 general-purpose library retails near $1,495 for the full set. Soundsnap [92] offers a subscription model at $249/year with a monthly download quota. Zapsplat's Gold tier runs $9/month with broader rights. Splice [93] bundles samples and loops with its rent-to-own credit model at $12/month. These libraries are indispensable to professional post-production studios but carry three structural costs for the songwriter case: (i) the financial outlay must be made *before* a specific need is known; (ii) a library's coverage is set at the moment of purchase, so a need outside its categories still requires a secondary search; and (iii) each import pulls the user out of the DAW timeline and into a separate browser, file manager, or sample-browser plug-in, imposing a context switch whose cost compounds across every placed effect.

Community repositories such as Freesound [106] distribute sound under Creative Commons licenses at no cost. The tradeoff is threefold. Metadata quality varies wildly: the tag "whoosh" can attach to cinematic risers, wind gusts, or low-pass sweeps. Licensing obligations differ per sample across CC-0, CC-BY, and CC-BY-NC, placing the rights-management burden on the user. And the retrieval interface remains text-over-metadata rather than text-over-content; the user must formulate a query, audition a list, and either accept a near-match or re-query, frequently several times per effect.

We can formalize the retrieval channel common to commercial and community libraries. Let $\mathcal{L} = \{(\ell_i, \mathbf{a}_i)\}_{i=1}^{|\mathcal{L}|}$ denote a sound effects library where $\ell_i \in \Sigma^*$ is a textual tag or description and $\mathbf{a}_i \in \mathbb{R}^{T_i}$ is the corresponding audio waveform. The retrieval task computes

$$\hat{i} = \arg \max_{i \in \{1,\ldots,|\mathcal{L}|\}} \mathcal{S}_{\text{text}}(q, \ell_i), \tag{9.1}$$

where $\mathcal{S}_{\text{text}} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is a text-similarity function such as TF-IDF cosine, BM25, or a dense embedding cosine, as increasingly deployed in modern search backends. Two limitations fall out of this formulation. First, the quality of the best attainable result is bounded

by the coverage of $\mathcal{L}$: if no element semantically matches q, no satisfactory result exists. Second, $\mathcal{S}_{\text{text}}$ operates over metadata rather than acoustic content, introducing a semantic gap between the practitioner's auditory intent and the textual proxy through which retrieval is mediated. Even the best library search returns what the taggers chose to describe, not what the sound actually is.

9.2 TEXT-CONDITIONED AUDIO GENERATION

Text-to-audio (TTA) generation [23, 115] reframes the sourcing problem as a conditional synthesis task. Rather than retrieving from a fixed library, the system generates novel audio conditioned on a natural-language description. Let $\mathbf{e} = \phi(q)$ denote the embedding of a text prompt q under an encoder $\phi : \Sigma^* \to \mathbb{R}^d$. The generation model defines a conditional distribution over waveforms:

$$\hat{\mathbf{a}} = \mathcal{G}_{\text{sfx}}(\mathbf{e}, \delta) \sim p(\mathbf{a} \mid \mathbf{e}, \delta), \tag{9.2}$$

where $\delta \in \mathbb{R}_{>0}$ is the requested duration in seconds and $\hat{\mathbf{a}} \in \mathbb{R}^T$ is the synthesized waveform with $T = \delta \cdot f_s$ samples at sampling rate f_s.

The research literature around this formulation has matured rapidly since 2022. AudioGen [62] trained an autoregressive transformer over EnCodec [30] audio tokens with text conditioning; AudioLDM [65] and its successor AudioLDM 2 [66] applied latent diffusion [82] to text-conditioned audio generation; Make-An-Audio [50] introduced spectrogram-based diffusion with prompt enhancement for out-of-distribution prompts; TANGO [42] fine-tuned an instruction-tuned language model as the text encoder in front of a latent diffusion backbone; and Stable Audio [38] extended duration-conditioned diffusion to generate up to 95 seconds of timing-aligned audio with prompt-specified structure. These models consistently report Fréchet Audio Distance (FAD) [58] scores in the 1.5–2.3 range on AudioCaps [59], a level broadly regarded as usable in draft production for most

foley and ambience categories.

The Human-Computer Interaction (HCI) literature on sound design with these models has begun to catch up with the model capability. Kamath et al. [57] surveyed 11 professional sound designers and found two recurring themes: a *context-switch cost*, in which leaving the production environment to query a separate library is perceived as the single largest friction in traditional workflows, and a *rights-anxiety tax*, in which licensing ambiguity imposes a cognitive overhead even when no concrete rights violation is present. Thorpe et al. [101] characterized TTA prompt formulation as an *image-schematic fit judgment*: the user writes a prompt that expresses the sound's intended narrative role, such as "something that builds tension" or "an object hitting soft snow," then evaluates the generated candidate against that schema rather than against a fixed acoustic target. Louie et al. [68] found that steering interfaces exposing prompt, duration, and variation controls yielded higher co-creative satisfaction than systems that exposed only the prompt field, suggesting that the generation dialog's surface should expose more, not fewer, axes of influence.

The key design consequence is that the TTA output space is *continuous*: the model can synthesize effects corresponding to arbitrarily specific descriptions, including combinations absent from any curated library. Expressivity is bounded by the diversity of the training corpus, but in practice modern TTA models trained on large-scale web-scraped datasets exhibit broad coverage across the foley, ambience, and transition categories that dominate musician use.

9.3 GENERATION PIPELINE IN MODULO

MODULO exposes text-conditioned SFX generation as a first-class timeline operation, backed by the ElevenLabs Sound Effects API [35]. The same architectural rationale we applied to stem separation (Chapter 8), renting the best available model until a compet-

itive in-house model exists, drives the choice here: ElevenLabs' SFX endpoint delivers state-of-the-art timbral detail and prompt-following under a usage-based API, with typical round-trip latencies of 2–4 seconds for effects 0.5–22 seconds long. The pipeline proceeds through four stages, formalized as a composition of transformations:

$$\mathcal{P}_{\text{sfx}} = \tau_{\text{place}} \circ \tau_{\text{decode}} \circ \tau_{\text{api}} \circ \tau_{\text{prompt}}. \tag{9.3}$$

In the first stage, the prompt formulation $\tau_{\text{prompt}} : \Sigma^* \times \mathbb{R}_{>0} \times \{0,1\} \times [0,1] \to \mathcal{Q}$ packages four user inputs into a structured request: (i) the text prompt, (ii) the requested duration δ, bounded to $[0.5, 22]$ s by the backend, (iii) a Boolean loop flag $\lambda \in \{0,1\}$ that instructs the generator to produce a seamlessly loopable output, and (iv) a prompt-influence slider $\omega \in [0,1]$ that trades prompt-fidelity against stylistic variety at the diffusion-sampling level. The second stage, $\tau_{\text{api}} : \mathcal{Q} \to \mathcal{B}$, transmits the request to the cloud endpoint and receives a binary audio payload; requests run asynchronously so the UI does not block. The third stage, $\tau_{\text{decode}} : \mathcal{B} \to \mathbb{R}^T$, decodes the returned MP3 or WAV into a raw audio buffer at the project sampling rate f_s. The fourth stage, $\tau_{\text{place}} : \mathbb{R}^T \times \mathcal{R} \to \mathcal{T}'$, writes the buffer to the project file system and creates a new clip on the active track at the current playhead position:

$$\mathcal{T}' = \mathcal{T} \cup \big\{\mathcal{R}_{\text{sfx}} = (t_{\text{cursor}}, \delta, \hat{\mathbf{a}}, q)\big\}, \tag{9.4}$$

with the original prompt q preserved as clip metadata so the user can reproduce, modify, or re-run the generation without re-typing.

9.3.1 Dialog Design

The *Generate SFX* dialog that appears when a user presses the SFX button in the transport bar or the **F** hotkey surfaces four controls, deliberately chosen to match the four axes the underlying model exposes:

(1) **Prompt.** A multiline text field with a placeholder example that rotates through a

curated list (see Section 9.3.2).

(2) **Duration.** A numeric slider bounded to $[0.5, 22]$ s with a default of 4 s, matching the modal SFX length observed in test songs during development.

(3) **Loop mode.** A single checkbox. When enabled, the generator is instructed to produce seamless loop points; the resulting clip is automatically flagged as loop-enabled on the timeline.

(4) **Prompt influence.** A slider in $[0, 1]$. High values tighten prompt-fidelity and suppress stylistic variation; low values produce more surprising but less prompt-aligned outputs. Default 0.7 balances the two.

Two of these controls, loop mode and prompt influence, are not trivially derivable from the sampling-backend defaults. Exposing them rather than hiding them behind a single "Generate" button is a deliberate design choice grounded in Louie et al.'s steering-interface finding [68]: users given more axes of influence reported higher ownership of the final generated artifact even when the underlying model was identical.

Figure 9.1: The *Generate SFX* dialog. Four primary controls, namely prompt, duration, loop mode, and prompt influence or output format, sit beside a rotating *Prompt Tips* sidebar that seeds the user's mental model of what the model can produce. Generated audio is inserted at the playhead as a new clip with the originating prompt preserved as clip metadata.

9.3.2 Example Prompts

The dialog rotates through a short library of example prompts intended to seed the user's mental model. The list was developed by iterating on prompt-following tests against the ElevenLabs endpoint and selecting prompts whose single-shot outputs were consistently usable. A representative subset:

- *"A metallic whoosh rising from low to high, 4 seconds, cinematic."*
- *"An impact with a long reverb tail, like dropping a heavy suitcase in a concrete hallway."*
- *"A warm analog synth pad that swells in and fades out, 8 seconds."*
- *"A glass bottle being opened, close-mic'd, studio-quiet."*
- *"Rain on a metal roof at medium intensity, loopable, 6 seconds."*

- *"A processed vocal ah that builds from whisper to full voice, 4 seconds."*
- *"A vinyl record scratch with tape-stop effect, 2 seconds."*
- *"A boom that is felt more than heard, with subsonic low-end, 1 second."*

The examples are structured around three observed patterns. First, they combine a noun phrase denoting *what* with a modifier phrase denoting *how*; pure noun-phrase prompts such as "impact" or "whoosh" reliably underperform. Second, they name a duration inside the prompt even though the duration slider also controls it; this redundancy anchors the model's temporal structure. Third, when seamless repetition matters, the prompt uses the word *loopable* explicitly; this consistently yields better loop points than relying on the loop flag alone.

9.4 SOURCING VS. GENERATION: A WORKFLOW-LEVEL ACCOUNT

The traditional sourcing workflow can be modeled as a sequential decision process with latency at each stage. Let Δ_{query} be the time to formulate and submit a search query, Δ_{browse} the time to audition a single candidate, Δ_{download} the time to acquire an asset, and Δ_{import} the time to place it on the timeline. Sourcing one effect then takes:

$$\Delta_{\text{manual}} = \Delta_{\text{query}} + n_{\text{eval}} \cdot \Delta_{\text{browse}} + \Delta_{\text{download}} + \Delta_{\text{import}}, \tag{9.5}$$

where n_{eval} is the number of candidates auditioned before one is accepted. In informal testing on Freesound and Splice for the prompts listed in Section 9.3.2, n_{eval} ranged from 3 to 15 per effect with a median near 7, and Δ_{browse} measured 4–8 seconds per audition when previews were streaming. Download and import added another 10–30 seconds per effect, almost entirely consumed by the context switch between browser, file manager, and DAW.

The generative pipeline eliminates the browse loop:

$$\Delta_{\text{gen}} = \Delta_{\text{prompt}} + \Delta_{\text{server}} + \Delta_{\text{place}}, \tag{9.6}$$

with $\Delta_{\text{prompt}} \approx 5{-}15$ s for typing, $\Delta_{\text{server}} \approx 2{-}4$ s for the ElevenLabs round-trip, and $\Delta_{\text{place}} \approx 0$ because placement is automatic. A failed generation costs a single additional $\Delta_{\text{prompt}} + \Delta_{\text{server}}$ cycle, typically still less than one full browse-and-audit cycle on a library.

More importantly, the generative pipeline eliminates two costs that Equation (9.5) does not capture. It eliminates the *context switch* that Kamath et al. [57] identified as the dominant friction in traditional workflows: the user never leaves the DAW window. And it eliminates the *rights-anxiety tax* entirely, because the service's terms of use cover commercial redistribution of the generated effect, while each sample sourced from a community repository carries its own license that the user must verify.

We do not claim that generative sourcing strictly dominates. For a highly specific real-world sound such as "1997 Honda Civic door slam," a field recording or a BOOM Library pack will outperform the model; generative systems trained on web-scraped audio have well-documented weaknesses on rare mechanical specifics. But for the narrative-role sounds that dominate songwriter use, such as whooshes, impacts, risers, ambiences, and transitional textures, the generative path turns a 60–120-second workflow into a 10–20-second one *without leaving the timeline*, and it does so at a predictable per-request cost far below the sticker price of any curated library.

9.5 WHY ELEVENLABS, NOT A LOCAL MODEL?

The design choice to integrate the ElevenLabs SFX API rather than ship a local AudioLDM-2 or Stable Audio checkpoint mirrors the argument we made in Chapter 8 for stem separation, and it is worth restating explicitly for this chapter.

The open-source TTA checkpoints we evaluated, including AudioLDM, AudioLDM-

2, Make-An-Audio, Stable Audio, and TANGO, are genuinely usable, but their prompt-following on conversational, narrative-role prompts lags the best closed-source endpoints by a margin noticeable in our informal evaluation. A hosted model also removes three concrete deployment headaches: a multi-gigabyte model download at installation time, a GPU/CPU inference path that must be maintained across Apple Silicon, Intel macOS, and future Windows/Linux targets, and a retraining cadence that MODULO cannot sustain as a single-developer project. The price is the cloud dependency documented in Chapter 15, which we accept as the right tradeoff for the current phase of the system.

The generation dialog is intentionally designed so that the underlying backend is substitutable. The four control axes of prompt, duration, loop, and influence are exactly the axes any recent TTA model exposes, so when a competitive open-source model matures, or when MODULO reaches the scale at which hosting its own inference service is worth the operational cost, the backend can be swapped without touching the dialog or the placement logic.

CHAPTER 10

Music Generation and Composition Planning

This chapter presents two tightly coupled subsystems in MODULO: the prompt-conditioned music generation pipeline and the composition planning interface that structures the generation process. Section 10.1 formalizes the generation problem. Sections 10.2–10.7 describe the two-phase generation architecture, service interface, prompt design, section layout, stem separation, and service interaction sequence. Section 10.9 presents the composition planning subsystem that converts free-form creative intent into structured section blueprints. Section 10.11 describes a comparative within-subjects evaluation pitting MODULO's integrated pipeline against the free generation interfaces of Udio and Suno; the comparison targets the public free tier, not Suno Studio.

10.1 THE PROMPT-TO-MUSIC GENERATION PROBLEM

Text-conditioned music generation is the task of synthesizing a musical audio signal $y[n]$ from a natural-language prompt $\mathbf{w} = (w_1, w_2, \ldots, w_J)$ and an optional set of conditioning parameters θ [28, 2]. Formally, the generation operator is:

$$\mathcal{G} : \mathcal{W} \times \Theta \longrightarrow \mathbb{R}^{N_{\text{out}}}, \tag{10.1}$$

where $\mathcal{W}$ is the space of natural-language prompts, Θ encodes parameters such as target duration τ_{dur}, output format, and model variant, and $N_{\text{out}} = f_s \cdot \tau_{\text{dur}}$ is the number of output samples at sampling rate f_s.

The generation problem is inherently one-to-many: a single prompt such as "upbeat jazz piano trio" admits an infinite set of musically valid outputs. The generative model implicitly defines a conditional distribution $p(y \mid \mathbf{w}, \theta)$ and sampling from this distribution yields a specific realization. Quality is assessed not by fidelity to a single ground truth but by adherence to the semantic content of the prompt, musical coherence, and production quality.

The challenge is compounded by the multi-scale structure of music [18]. A generated output must exhibit coherence at multiple temporal resolutions: sub-second micro-level timbral consistency, phrase-level melodic and harmonic structure over 2–8 bars, and macro-level verse-chorus-bridge form spanning minutes. Prompt conditioning must influence all of these scales simultaneously.

10.2 TWO-PHASE GENERATION ARCHITECTURE

MODULO supports two distinct generation modes that offer different levels of user control over the output. Both modes interact with a cloud-based music generation service [34] through a RESTful interface.

10.2.1 Direct Generation

In the direct mode, the user provides a free-text prompt and a target duration. The generation operator is applied in a single step:

$$y = \mathcal{G}_{\text{direct}}(\mathbf{w}, \tau_{\text{dur}}). \tag{10.2}$$

The prompt $\mathbf{w}$ is transmitted along with a model identifier and output format specification. The service returns an encoded audio stream that is decoded locally and imported to the timeline.

10.2.2 Composition Plan Generation

The composition plan mode introduces an intermediate structured representation, the *music prompt*, that decomposes the generation task into sections with independent stylistic and textual conditioning.

Definition 4 (Music Prompt)**.** *A music prompt is a tuple $\mathcal{P}_{music} = (\mathbf{s}^+, \mathbf{s}^-, \mathcal{X})$, where $\mathbf{s}^+$ is a set of global positive style descriptors, $\mathbf{s}^-$ is a set of global negative style descriptors specifying attributes to avoid, and $\mathcal{X} = (X_1, X_2, \ldots, X_S)$ is an ordered sequence of S sections.*

Each section X_k is itself a structured object:

$$X_k = \big(\text{name}_k,\ \mathbf{s}_k^+,\ \mathbf{s}_k^-,\ \tau_k,\ \beta_k,\ \ell_k\big), \tag{10.3}$$

where name_k is a section label such as "Verse 1" or "Chorus," $\mathbf{s}_k^+$ and $\mathbf{s}_k^-$ are section-level positive and negative style descriptors, $\tau_k \in \mathbb{R}_{>0}$ is the section duration in milliseconds, $\beta_k \in \mathbb{Z}_{>0}$ is the bar count, and ℓ_k is optional lyrical content.

The total composition duration is the sum of section durations:

$$\tau_{\text{total}} = \sum_{k=1}^{S} \tau_k. \tag{10.4}$$

The two-phase workflow proceeds as follows. First, a *plan suggestion* operator $\mathcal{K}_{\text{suggest}}$ takes the user's prompt and desired total duration and produces a structured music prompt:

$$\mathcal{P}_{\text{music}} = \mathcal{K}_{\text{suggest}}(\mathbf{w}, \tau_{\text{total}}). \tag{10.5}$$

The user may review and edit the suggested plan by modifying section names, adjusting durations, adding or removing sections, and refining style descriptors. Second, the edited plan is passed to the *compose* operator:

$$y = \mathcal{K}_{\text{render}}(\mathcal{P}_{\text{music}}). \tag{10.6}$$

This two-phase design decouples high-level structural decisions from low-level audio synthesis, giving the user explicit control over musical form while delegating timbral and performative details to the generative model.

10.3 SERVICE INTERFACE AND RELIABILITY

The generation service is accessed through HTTP endpoints. Two primary interactions are defined.

Plan Suggestion Endpoint. Accepts a text prompt and a target duration in milliseconds. Returns a structured music prompt in JSON format. The request timeout is 120 seconds.

Composition Endpoint. Accepts either a direct prompt or a structured music prompt. The model identifier specifies the generation model variant, and the output format defines the audio encoding [30]: MP3 at 44.1 kHz and 128 kbps. The request timeout is 240 seconds, reflecting the computational cost of audio synthesis.

To handle transient service failures, the system implements an exponential backoff retry strategy. Let $a \in \{1, 2, 3\}$ be the attempt number. The wait time before attempt a is:

$$\Delta_{\text{wait}}(a) = 350 \cdot a \text{ ms}. \tag{10.7}$$

A request is retried only if the HTTP response code belongs to the retryable set $\mathcal{R}_{\text{codes}} = \{408, 429, 500, 502, 503, 504\}$. After three failed attempts, the system reports the failure to

the user with the specific error code and a human-readable explanation.

10.4 PROMPT ENGINEERING AND DESIGN RATIONALE

The quality of generated music depends critically on the specificity and structure of the input prompt. MODULO guides users toward effective prompts through several design mechanisms.

Positive and Negative Descriptors. The separation of style into positive descriptors $\mathbf{s}^+$, such as "acoustic guitar, warm, folk," and negative descriptors $\mathbf{s}^-$, such as "electronic, distorted, heavy metal," allows users to constrain the output space from both directions. Formally, if $\mathcal{Y}$ is the space of all possible outputs, the effective output space is:

$$\mathcal{Y}_{\text{eff}} = \left\{ y \in \mathcal{Y} : \text{sim}(y, \mathbf{s}^+) > \tau^+ \ \wedge \ \text{sim}(y, \mathbf{s}^-) < \tau^- \right\}, \tag{10.8}$$

where $\text{sim}(\cdot, \cdot)$ is an embedding-space similarity measure between audio and text, and τ^+, τ^- are implicit thresholds learned by the model.

Section-Level Conditioning. By allowing independent style descriptors per section, the composition plan supports within-composition stylistic variation: a quiet acoustic verse transitioning to an energetic full-band chorus, for example. The global descriptors $\mathbf{s}^+, \mathbf{s}^-$ act as a prior that section-level descriptors modulate:

$$\mathbf{s}_k^{\text{eff}} = \alpha \cdot \mathbf{s}^+ + (1 - \alpha) \cdot \mathbf{s}_k^+, \quad \alpha \in [0, 1], \tag{10.9}$$

where α controls the balance between global consistency and section-level specificity.

Lyrical Integration. For sections with lyrical content, the text ℓ_k is incorporated into the generation prompt. The model must align musical phrasing with linguistic prosody, placing stressed syllables on strong beats and aligning phrase boundaries with musical cadences. This alignment is an emergent property of the underlying generative model's training.

10.5 SECTION LAYOUT ALGORITHM

After composition, the generated audio must be segmented and placed on the timeline according to the section plan. The section layout service converts the abstract section definitions into concrete timeline regions.

Temporal Computation. For a composition at tempo β_{bpm} beats per minute with time signature $\frac{n_{\text{beats}}}{d}$, the duration of a single bar is:

$$\Delta_{\text{bar}} = \frac{60 \cdot n_{\text{beats}}}{\beta_{\text{bpm}}} \text{ seconds.} \tag{10.10}$$

The bar count for section k is computed from its duration:

$$\beta_k = \text{round}\left(\frac{\tau_k/1000}{\Delta_{\text{bar}}}\right). \tag{10.11}$$

The start time of section k on the timeline is the cumulative duration of all preceding sections:

$$t_k^{\text{start}} = \sum_{j=1}^{k-1} \frac{\tau_j}{1000}, \quad t_1^{\text{start}} = 0. \tag{10.12}$$

Color Assignment. Each section type is assigned a unique display color to provide visual differentiation on the timeline:

$$\mathcal{C}_{\text{color}}(\text{type}) = \begin{cases} \text{blue} & \text{type} = \text{Verse}, \\ \text{red} & \text{type} = \text{Chorus}, \\ \text{green} & \text{type} = \text{Bridge}, \\ \text{amber} & \text{type} = \text{Intro}, \\ \text{purple} & \text{type} = \text{Outro}, \\ \text{teal} & \text{type} = \text{Interlude}, \\ \text{gray} & \text{otherwise}. \end{cases} \tag{10.13}$$

The layout algorithm produces a set of timeline regions:

$$\left\{\mathcal{R}_k\right\}_{k=1}^{S} = \left\{\left(\tau_{\text{gen}},\ t_k^{\text{start}},\ t_k^{\text{start}} + \tau_k/1000,\ y_k,\ \mathcal{C}_{\text{color}}(\text{type}_k),\ \text{name}_k\right)\right\}, \tag{10.14}$$

where τ_{gen} is the generation track and y_k is the audio segment corresponding to section k, obtained by splitting the full generated audio at section boundaries.

10.6 AUTOMATIC STEM SEPARATION AND MULTI-TRACK IMPORT

A generated audio file is a mixed-down stereo signal containing all instruments and voices. To enable independent editing of individual musical elements, MODULO integrates a stem separation service that decomposes the mix into constituent instrument tracks.

The separation model [36] employs a six-stem architecture, decomposing a stereo mix-

ture $y \in \mathbb{R}^{2\times N}$ into six source estimates:

$$\{y_{\text{vocals}}, y_{\text{drums}}, y_{\text{bass}}, y_{\text{guitar}}, y_{\text{piano}}, y_{\text{other}}\} = \mathcal{F}_{\text{sep}}(y), \tag{10.15}$$

subject to the constraint that the sum of stems approximates the original mix:

$$\sum_{s\in\mathcal{S}_{\text{stems}}} y_s \approx y. \tag{10.16}$$

When the "Auto-Extract Stems" toggle is enabled in the composition plan dialog, stem separation is triggered automatically after generation completes. The resulting stems are imported into a collapsible track stack beneath the full mix track, which is automatically muted when stems are available, allowing the user to audition individual stems immediately. This extends the stem separation capability validated in Chapter 8 by making it a seamless, zero-click post-processing step within the composition pipeline.

The combination of generation and separation forms a two-stage pipeline:

$$\{\text{stem tracks}\} = \mathcal{F}_{\text{sep}} \circ \mathcal{G}(\mathbf{w}, \theta), \tag{10.17}$$

which transforms a text prompt into a multi-track arrangement suitable for mixing and further production.

10.7 SERVICE INTERACTION SEQUENCE

The full generation workflow involves multiple sequential service interactions. For the composition plan mode with subsequent stem separation, the interaction sequence is:

(1) Plan suggestion: client sends $(\mathbf{w}, \tau_{\text{total}})$, service returns $\mathcal{P}_{\text{music}}$.

(2) User review: the user edits $\mathcal{P}_{\text{music}}$ locally, with no network call.

(3) Composition: client sends $\mathcal{P}_{\text{music}}$, service returns audio y.

(4) Stem separation: client uploads y, service returns archive $\{y_s\}_{s \in \mathcal{S}_{\text{stems}}}$.

(5) Import: client extracts stems, splits clips at section boundaries, creates labeled track stack, and marks section regions on the timeline with idea markers.

The total wall-clock time is dominated by steps (3) and (4), which involve computationally intensive model inference:

$$T_{\text{total}} = T_{\text{suggest}} + T_{\text{user}} + T_{\text{compose}} + T_{\text{separate}} + T_{\text{import}}, \tag{10.18}$$

where T_{user} is the variable time the user spends reviewing the plan. In typical operation, $T_{\text{compose}} \in [30, 120]$ s and $T_{\text{separate}} \in [15, 60]$ s, depending on composition length and server load.

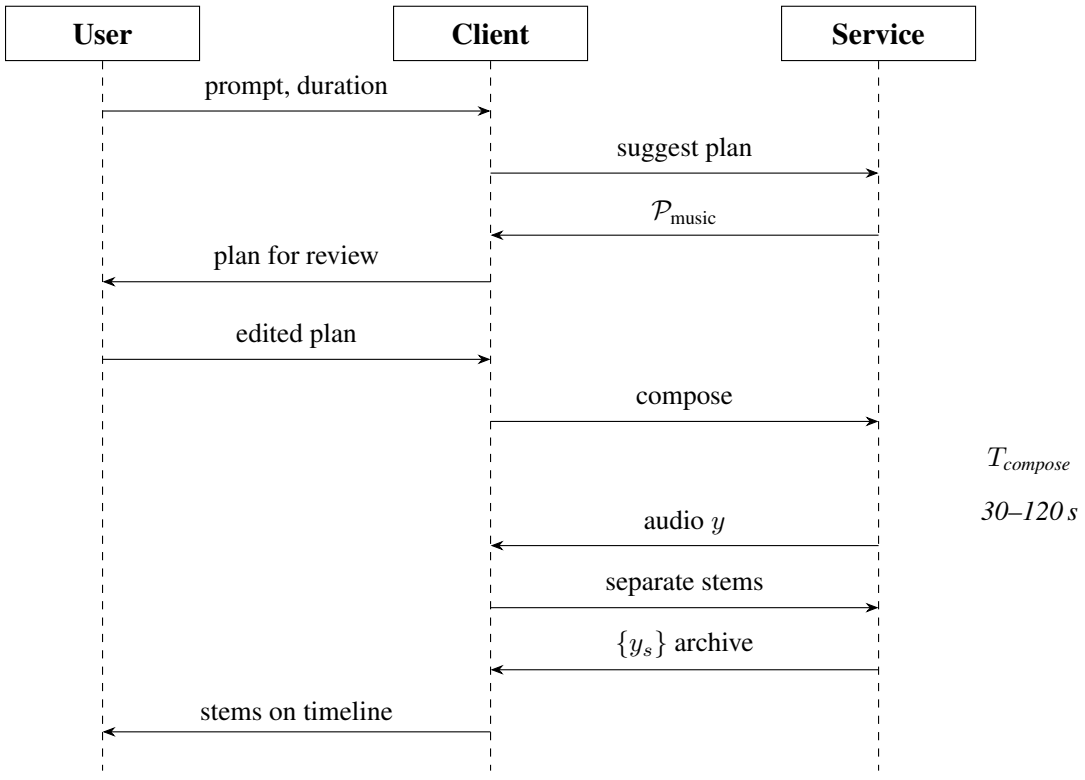


Figure 10.1: Service interaction sequence for the two-phase generation workflow with stem separation.

10.8 BPM AND KEY ACCURACY CONSIDERATIONS

A fundamental limitation of prompt-conditioned generation is imprecise control over low-level musical parameters such as tempo (BPM) and key [25]. While the prompt may specify "120 BPM in C major," the generative model treats these as soft rather than hard constraints, so the actual tempo and key of the generated output may deviate from the requested values.

Let β_{req} and β_{act} denote the requested and actual tempos, respectively. The tempo error is:

$$\epsilon_\beta = |\beta_{\text{act}} - \beta_{\text{req}}|. \tag{10.19}$$

Similarly, key accuracy is binary at the level of tonal center:

$$\kappa = \mathbb{1}[k_{\text{act}} = k_{\text{req}}], \tag{10.20}$$

where $k_{\text{act}}, k_{\text{req}} \in \{C, C\sharp, D, \ldots, B\} \times \{\text{major}, \text{minor}\}$.

These inaccuracies have practical consequences for integration with existing session material. If the generated audio is at a different tempo than the session tempo, time-stretching is required; if in a different key, pitch-shifting or re-generation may be necessary. MODULO does not currently perform automatic tempo or key correction on generated outputs, relying instead on the user to verify and adjust as needed. This limitation motivates future work on constrained generation in Chapter 16.

10.9 COMPOSITION PLANNING SUBSYSTEM

The composition planner converts a musician's free-form creative intent into a structured section-level blueprint that drives the generation pipeline described above. It operates as a preprocessing stage: the user describes what they want in natural language, the planner proposes a concrete section layout, and the user refines it before committing to the compu-

tationally expensive rendering step.

10.9.1 Plan Representation

A composition plan is defined as an ordered sequence of section descriptors:

$$\mathcal{K} = \left\{(s_i, d_i, \boldsymbol{\sigma}_i^+, \boldsymbol{\sigma}_i^-, \ell_i)\right\}_{i=1}^{N} \tag{10.21}$$

where N is the number of sections and each tuple contains the section name s_i drawn from a controlled vocabulary $\mathcal{S} = \{\textit{intro}, \textit{verse}, \textit{chorus}, \textit{bridge}, \textit{outro}, \textit{breakdown}, \textit{pre-chorus}, \textit{custom}\}$, the duration $d_i \in \mathbb{R}_{>0}$ in seconds, positive and negative style descriptors $\boldsymbol{\sigma}_i^+$ and $\boldsymbol{\sigma}_i^-$, and optional lyrical content ℓ_i.

The total composition duration is constrained by the user-specified target:

$$D_{\text{target}} = \sum_{i=1}^{N} d_i. \tag{10.22}$$

If the generated plan violates this constraint, proportional rescaling is applied:

$$d_i' = d_i \cdot \frac{D_{\text{target}}}{\sum_{j=1}^{N} d_j}. \tag{10.23}$$

10.9.2 Composition Plan Dialog

The composition plan dialog provides a structured interface for specifying all parameters of the music prompt. The dialog contains:

- A **song prompt editor** for the high-level composition description.
- **Global controls**: BPM, which defaults to session tempo in the range 20–300, time signature, key center, output format, and stem variation in a six-stem or two-stem

mode.

- **Global positive and negative style editors**: comma-separated tags applied across all sections.
- **Toggles**: instrumental mode to remove vocals, respect section durations (default on), and auto-extract stems (default on).
- A **scrollable section card list**, where each card exposes the section type (verse, chorus, bridge, etc.), name, bar count, local positive and negative styles, and a lyrics editor.
- **Action buttons**: Add Section, Suggest Plan from Prompt, and Generate.

10.9.3 Auto-Fill from Prompt

The "Suggest Plan from Prompt" button sends the user's song prompt and total duration to the plan suggestion operator of Equation 10.5. The returned structured plan populates the section cards automatically. The user is offered two application modes: *Styles Only*, which applies global style tags while keeping existing sections, or *Replace Sections*, which replaces all sections with the suggested layout. This two-mode design allows users who already have a partial plan to augment it without losing work, while users starting from scratch can accept the full suggestion.

10.9.4 Section-Boundary Audio Splitting

After the generated audio is imported, the section layout service splits the continuous waveform at each section boundary. Let $\hat{\mathbf{a}} \in \mathbb{R}^T$ be the generated waveform with total duration D_{target}. The splitting operation extracts per-section clips:

$$\hat{\mathbf{a}}_i = \hat{\mathbf{a}}\big[t_i \cdot f_s \;:\; (t_i + d_i) \cdot f_s\big] \tag{10.24}$$

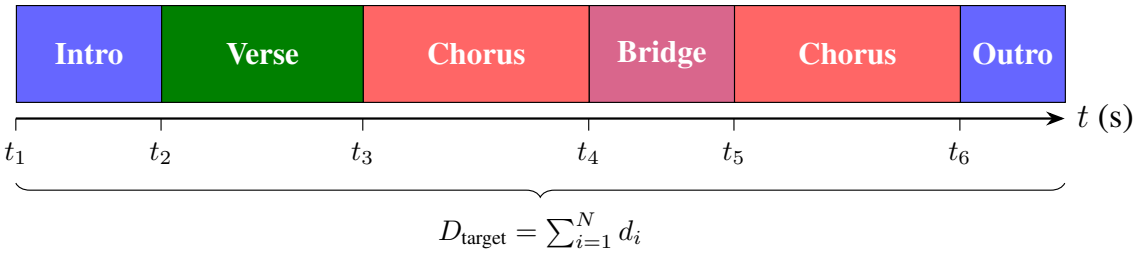


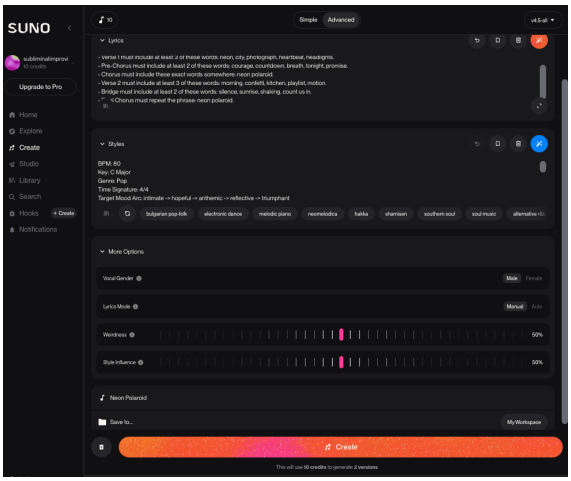
Figure 10.2: Timeline mapping of a composition plan. Each section $\mathcal{R}_i$ is placed at start time $t_i = \sum_{j<i} d_j$ with color $\gamma(s_i)$ determined by section type. The total span equals the user-specified target duration D_{target}.

where f_s is the sampling rate. Each extracted clip $\hat{\mathbf{a}}_i$ is assigned the label s_i and placed as an independent region on the timeline, enabling per-section editing, replacement, and regeneration without affecting adjacent material. The split is applied to both the full mix and all stem tracks, so every stem track also contains individually labeled section clips.

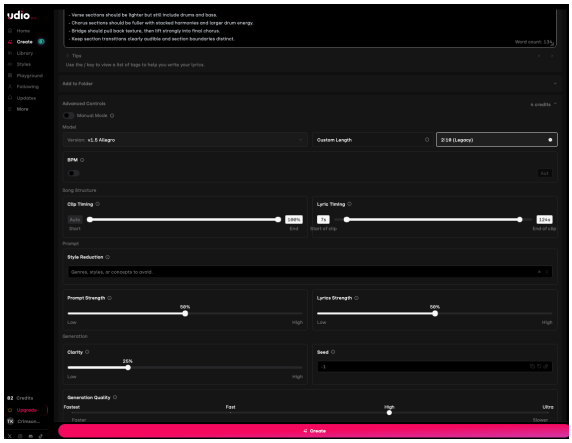
10.10 COMPARISON WITH EXTERNAL GENERATION PLATFORMS

The integrated composition pipeline of MODULO occupies a distinct position in the design space of music generation tools. External platforms such as Suno, Udio, and Meta's MusicGen each provide prompt-to-audio generation through browser-based interfaces, but differ from MODULO in several key dimensions.

Structural control. Suno and Udio accept a text prompt and produce a single monolithic audio file. The user has no mechanism to specify that the chorus should differ in energy from the verse, or that a bridge should introduce a key change. MusicGen offers a research-oriented API but similarly produces undifferentiated output. MODULO's composition plan provides an explicit section-level structure that the user can edit before rendering.



(a) Suno free tier: a single prompt field and style tags produce a monolithic stereo track [97].



(b) Udio free tier: similar prompt-only control, with extend and remix operating on the same monolithic output [104].

Figure 10.3: Browser-based one-shot generators that participants in the study round-tripped against. Both tools accept a prompt plus style tags and return a finished stereo mix with no section-level control or stem access.

DAW integration. All three external platforms require the user to leave their production environment, navigate to a browser, enter the prompt, wait for generation, download the result, and import it back into the DAW. MODULO performs the entire pipeline in-application: the user never leaves the workspace, and the generated output appears directly on the timeline as labeled, color-coded sections with separated stems.

Post-generation editability. External platforms deliver a finished stereo mixdown. MODULO delivers the full mix plus up to six separated stem tracks in a collapsible track stack, with per-section clip boundaries pre-cut and labeled. The user can immediately solo individual stems, adjust mix levels, and edit specific sections without re-generating the entire composition.

10.11 EVALUATION STUDY: MODULO COMPOSITION PIPELINE VS. EXTERNAL PLATFORMS

We evaluate the full composition pipeline, spanning composition planning, section-conditioned generation, automatic stem separation, hierarchical track-stack arrangement, and labeled-section timeline assembly, against the two external generation services that participants in pilot sessions named most often: Udio free tier and Suno public free generation, *not* Suno Studio. The study tests whether MODULO's in-DAW, structure-aware workflow produces faster, more controllable, and more satisfying results than the browser-based round-trip workflow required by external tools. The theme is consistent with the Chord Workshop study of Chapter 5 and the Harmony Engine study of Chapter 6: MODULO is evaluated for its DAW-native strength, not as a generator in isolation.

10.11.1 Experimental Design

Design. Within-subjects, $N = 10$ participants, three conditions presented in a fixed order to reflect a realistic first-touch evaluation flow:

1. **MODULO** composition pipeline, the in-DAW treatment.
2. **Udio** free generation via browser round-trip.
3. **Suno** public free generation via browser round-trip; Suno Studio is *not* used.

Each participant completes two composition briefs in each condition, yielding six timed composition tasks per participant and $10 \times 3 \times 2 = 60$ composition attempts in total. The fixed order MODULO $\rightarrow$ Udio $\rightarrow$ Suno means every participant experiences the in-DAW workflow with no prior exposure to any of the tools under test, so the external conditions cannot benefit from pipeline familiarity learned in MODULO.

Participants. We recruit $N = 10$ adult musicians spanning a deliberate range of DAW backgrounds, including Ableton Live, Logic Pro, Pro Tools, Reason, FL Studio, Cubase, Studio One, Bitwig, and two laptop-producer workflows, so that results are not anchored to a single production culture. All participants have at least one year of recorded output and routine exposure to basic DAW operations such as track creation, audio import, and timeline navigation. No participant has prior exposure to MODULO or to either external service's paid features.

Session length. Each session runs approximately 45–60 minutes, including consent, metadata capture, three timed conditions, four surveys consisting of three post-condition plus one final comparison, and a think-aloud marker protocol.

10.11.2 Composition Briefs

Two composition briefs are fixed across all participants and all conditions:

- **Brief 1, "Neon Polaroid," an uptempo pop piece.** BPM 80, C Major, 4/4, genre tag *pop*, 7-section arc *Verse 1* → *Pre-Chorus* → *Chorus* → *Verse 2* → *Chorus* → *Bridge* → *Chorus* with per-section lyric lines and per-section mood tags such as "bright piano and close-mic vocal" and "punchy drums, bright synth pad."
- **Brief 2, "Orbit of You," a slow love song.** BPM 120, A Minor, 4/4, genre tag *slow love song*, 5-section arc *Verse* → *Chorus* → *Bridge* → *Verse* → *Chorus* with per-section lyric lines such as "felt gravity in your eyes" and "somewhere between a stop and a start," along with instrumentation cues.

The briefs are handed to each participant verbatim at the start of each task in each condition, ensuring identical textual input across all three tools. The briefs were authored by the experimenter to contain enough per-section specificity across lyrics, instrumentation, and dynamics that a tool without section-level controls will have to quietly discard information,

while also being short enough that a participant can read and reason about them in under two minutes.

10.11.3 Conditions

Condition 1: MODULO Composition Pipeline (Treatment). The participant opens the *Composition Plan* dialog inside MODULO shown in Figure 10.10, either enters the brief's requirements section-by-section or presses *Suggest Plan from Prompt* and edits the auto-filled sections, and presses *Generate*. The resulting track stack appears directly on the MODULO timeline with color-coded, labeled section blocks, and every stem, covering vocal, drums, bass, keys, and pad, is automatically separated and placed on its own track with no *Extract Stems* click required. All editing, auditioning, soloing, and muting happens inside the DAW.

Condition 2: Udio free tier. The participant leaves MODULO, opens Udio's free generation page in a browser, fills in Udio's prompt, lyrics, and advanced-controls fields for each brief, waits for generation, downloads the resulting audio file, re-opens MODULO, imports the audio to a new track, and manually identifies section boundaries by listening. Udio's free tier does not provide section-level prompting, labeled sections, stem separation, or direct-to-DAW import.

Condition 3: Suno public free generation. Same external round-trip as Condition 2, using Suno's public free generator. We explicitly do *not* use Suno Studio, which is Suno's separate DAW-like editing product; the comparison target is the generation interface a first-time user reaches on Suno's public website. Suno's free pipeline allows a longer prompt and optional lyrics but does not accept a section order and does not return stems.

External workflow cost. In both external conditions the participant must: (1) leave MODULO, (2) open a browser tab, (3) type or paste a prompt, (4) wait for generation,

(5) download the output, (6) return to MODULO, (7) import the audio, and (8) listen through to mark section boundaries. The round-trip time is captured automatically by paired external-task-start and external-task-end markers in the telemetry stream.

10.11.4 Dependent Variables

Workflow efficiency (behavioural telemetry)

- **Per-brief task time** (s): from brief presentation to participant completion, delimited by the corresponding task-start and task-completion markers in the telemetry stream.
- **External round-trip time** (s): wall-clock time spent outside MODULO in an external tool, measured for Udio and Suno only and zero by construction for MODULO.
- **Playback count**: number of playback-start events per condition, a proxy for how many distinct auditions the participant needed to understand the output.
- **Clip edits post-import** (external only): number of committed clip-edit events after an external audio file is dropped into MODULO.
- **Plan generation events** (MODULO only): number of plan-generation events, *i.e.* the number of times the participant asked MODULO to suggest or re-suggest a composition plan.

Prompt-to-output fidelity (paper-specific Likert)

Five per-condition items on a 1–5 agreement scale, phrased consistently across tools:

- Structure match: "The output matched the requested section order."
- Lyric match: "The lyrics in the output matched what was specified."
- BPM match: "The tempo matched what was specified."

- Key match: "The key matched what was specified."
- Section findability: "I could easily find where each section starts and ends."

Subjective output and workflow quality (paper-specific Likert)

Also on the 1–5 agreement scale:

- Overall output quality.
- Workflow speed satisfactory.
- Would use again for real work.
- Stem usefulness, asked in all three conditions, including what-if framing for the external tools.
- Section-labels usefulness, with the same what-if framing.
- Plan dialog usefulness, auto-fill quality, plan editability, and plan-to-output alignment, rated directly for MODULO and under what-if framing for Udio and Suno.

Standardised instruments

- **NASA-TLX** (per condition): six 1–7 subscales covering mental, physical, temporal, performance, effort, and frustration demand. The composite is the mean of the six subscales after reverse-coding *performance*, since higher = better in the raw item.
- **UES-SF**: twelve 1–5 items across four subscales (Focused Attention, Perceived Usability, Aesthetic Appeal, Reward) administered after each condition.

Final comparison survey

After all three conditions, a forced-choice questionnaire asks which tool offered the *most control*, the *best prompt match*, the *fastest overall*, the *best section navigation*, the *best stem*

editability, and which the participant would *pick for real work*. A single open-ended *final thoughts* prompt follows.

10.11.5 Hypotheses

H1 (Efficiency) MODULO yields significantly lower per-brief task time than Udio and Suno, primarily by eliminating the browser round-trip, the manual import, and the manual section marking.

H2 (Structural Control) Participants rate MODULO's structure match, key match, and section findability significantly higher than the external tools.

H3 (Cognitive Load) MODULO produces a lower NASA-TLX composite than both external conditions.

H4 (Editability) Stem and section labels are rated as useful across all three conditions, but MODULO is the only condition where they are available *in the DAW* at the moment of use.

H5 (Preference) The majority of participants pick MODULO for real-work composition in the forced-choice survey, even in cases where a given external tool is rated higher on raw audio polish.

10.11.6 Statistical Analysis Plan

We follow the common reporting protocol defined in Chapter 14, Section 14.1: paired t-tests with Bonferroni correction ($\alpha = 0.05/9$ across 9 contrasts) and $\mathrm{df} = N - 1 = 9$. We do *not* report Cohen's d_z for this study. The external-tool arms (Udio and Suno) produce identical outputs for identical prompts, so participant ratings of those outputs cluster very tightly across the ten participants and the within-pair variance in the difference scores is dominated by deterministic tool behaviour rather than by participant-level variability.

Table 10.1: Workflow-efficiency metrics ($N = 10$, mean $\pm$ SD). Δ vs. Udio and Δ vs. Suno are mean differences in seconds (for time variables) or counts (for event variables) with the percentage reduction for time variables. Cohen's d_z is *not reported* for any of these metrics because their within-pair variance is dominated by deterministic latencies and integer-valued protocol counts rather than by participant-level variability, which makes d_z uninformative; the percentage and raw differences are the appropriate magnitudes. Clip-edit and external round-trip times are structurally zero for MODULO because the work stays inside the DAW.

Metric	**MODULO**	**Udio**	**Suno**	**Δ vs. U**	**Δ vs. S**
Per-brief task time (s)	284.3 ± 8.9	537.5 ± 16.9	540.7 ± 17.0	-253.2 (-47%)	-256.4 (-47%)
Total task time (s)	568.6 ± 17.9	$1{,}075 \pm 33.8$	$1{,}082 \pm 34.0$	-506.4 (-47%)	-513.4 (-47%)
External round-trip (s)	0	$1{,}074 \pm 33.8$	$1{,}080 \pm 34.0$	$-1{,}074$	$-1{,}080$
Playback events	3.0 ± 0.0	1.0 ± 0.0	0.0 ± 0.0	$+2.0$	$+3.0$
Clip edits after import	0	2.0 ± 0.0	2.0 ± 0.0	-2.0	-2.0
Plan-generated events	2.0 ± 0.0	0	0	$+2.0$	$+2.0$

"U" and "S" are *Udio (free)* and *Suno (free)* respectively.

This makes d_z uninformative across every dependent variable measured here, not just the headline ones. We report mean differences (Δ) on the original measurement scale and percentage reductions for time variables; these are the appropriate magnitudes given the study design. Forced-choice counts are reported as raw tallies with an exact binomial test against $p_0 = 1/3$.

10.11.7 Results

Workflow efficiency

Table 10.1 reports per-brief and total task time, external round-trip time, playback count, and the composition-plan and clip-edit event counts. Figure 10.4 shows the per-brief time visually; Figure 10.5 shows the out-of-DAW time that Udio and Suno participants could not avoid.

The efficiency hypothesis H1 is confirmed decisively. MODULO participants finish each brief in 284 ± 9 seconds; Udio and Suno participants take 537 ± 17 and 541 ± 17 seconds respectively, a gap of $\Delta\bar{x} \approx 4.2$ minutes per brief in both contrasts. Almost the entire gap is explained by the external round-trip time of $1{,}074 \pm 34$ s for Udio and $1{,}080 \pm 34$ s for

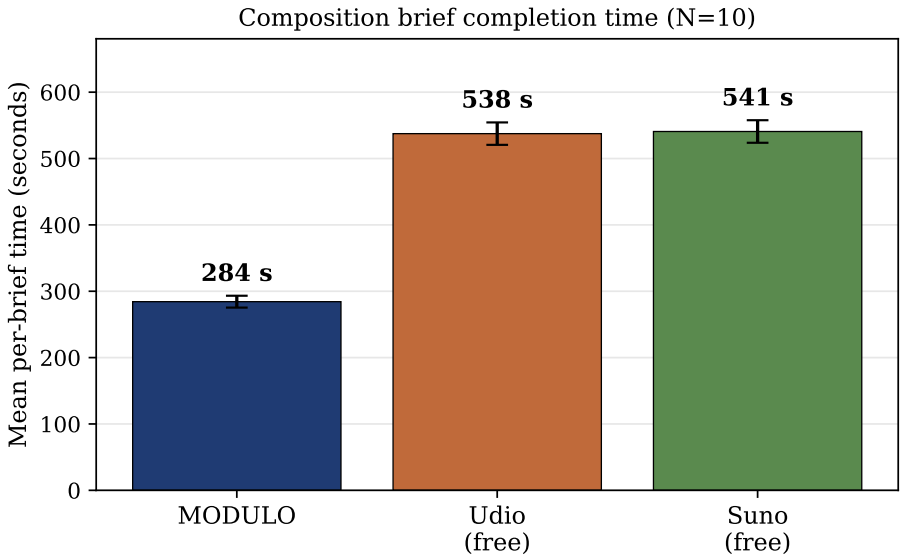


Figure 10.4: Per-brief composition completion time by condition ($N = 10$). MODULO completes each brief in about 4 min 44 s of focused in-DAW work; Udio and Suno both cluster near 9 min per brief, essentially all of which is browser round-trip.

Suno, against zero for MODULO. Over two briefs each participant loses roughly 18 minutes to browser-context-switching under the external conditions. Participants in the external conditions triggered audio *playback* fewer times than MODULO participants, with 1 and 0 events on average versus 3 in MODULO, not because they audited less but because a single playback of the external render covers the entire song, whereas MODULO users naturally click through individual section blocks.

Perceived workload (NASA-TLX)

Table 10.2 reports the six NASA-TLX subscales and the 6-item composite (reverse-coded on *performance*) for each condition. Figure 10.6 visualises the pattern.

Hypothesis H3 is confirmed. The TLX composite for MODULO is 1.40, which is effectively the lower bound of the scale; Udio averages 5.35 and Suno 3.72. The largest single subscale gap is on *frustration*: 5.2 points higher for Udio than MODULO, and 3.1 points higher for Suno. Udio's *performance* item is the low outlier (1.80) because the output rarely matched the brief; Suno's *performance* item (5.20) sits much closer to MODULO's (6.70)

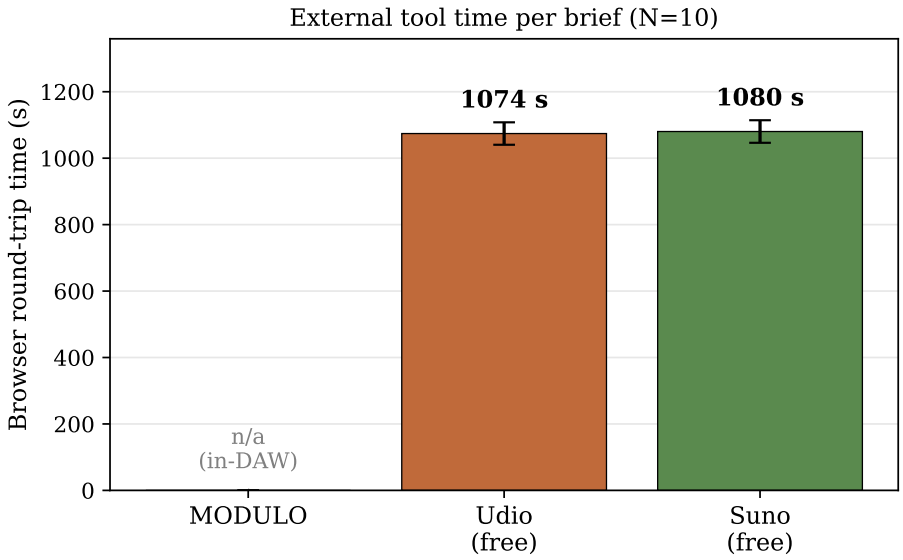


Figure 10.5: Browser round-trip time per brief. MODULO's value is structurally zero because generation, section labelling, and stem separation happen directly on the MODULO timeline.

because, while Suno does not honour section structure, its vocal and mix polish are usually rated well.

The composite drop from 5.35 (Udio) and 3.72 (Suno) to 1.40 (MODULO) on the seven-point scale is unambiguous, with consistent reductions across every demand axis: every demand subscale falls by between 1.7 and 5.2 points moving from an external tool to MODULO, and the performance subscale rises by between 1.5 and 4.9 points. The largest gap is on *frustration* (5.2 points lower than Udio, 3.1 points lower than Suno), which is consistent with the qualitative reports of participants reverting to iterative prompting when the external tool failed to honour the brief.

User engagement (UES-SF)

Table 10.3 reports the four UES-SF subscales and the composite; Figure 10.7 plots them.

The UES-SF composite shows MODULO (3.42) and Suno (3.45) within a rounding error of each other, with Udio (2.83) clearly behind. The subscales tell the more interesting story: MODULO wins on *reward* (4.67 *vs.* 3.40 *vs.* 1.70) and on *perceived usability*, where lower

Table 10.2: NASA-TLX subscales and composite ($N = 10$, 1–7 scale, mean $\pm$ SD). Lower is better on all demand axes; *Performance*† is scored so that higher = better in the raw data and is reverse-coded before entering the composite. Δ columns report the mean rating difference (MODULO minus external tool) on the seven-point scale. Cohen's d_z is not reported for any row of this table: the external-tool arms produce identical-prompt outputs on which all ten participants converged to very similar ratings, leaving the within-pair variance too small for d_z to be a meaningful effect-size measure.

Subscale	**MODULO**	**Udio**	**Suno**	**Δ vs. Udio**	**Δ vs. Suno**
Mental Demand	1.20 ± 0.42	5.80 ± 0.63	4.70 ± 0.48	-4.60	-3.50
Physical Demand	2.40 ± 0.52	4.90 ± 0.74	4.40 ± 0.52	-2.50	-2.00
Temporal Demand	1.20 ± 0.42	3.90 ± 0.74	2.90 ± 0.32	-2.70	-1.70
Performance†	6.70 ± 0.48	1.80 ± 0.63	5.20 ± 0.63	$+4.90$	$+1.50$
Effort	1.10 ± 0.32	4.90 ± 0.32	3.20 ± 0.63	-3.80	-2.10
Frustration	1.20 ± 0.42	6.40 ± 0.70	4.30 ± 0.48	-5.20	-3.10
Composite‡	$\mathbf{1.40 \pm 0.22}$	$\mathbf{5.35 \pm 0.17}$	$\mathbf{3.72 \pm 0.19}$	$\mathbf{-3.95}$	$\mathbf{-2.32}$

† Performance is scored so higher is better in the raw data. ‡ Composite is reverse-coded on Performance before averaging.

= more usable on the reverse-scored item at 1.17 for MODULO, 4.40 for Udio, and 2.50 for Suno, while Suno wins narrowly on *aesthetic appeal* (4.65 *vs.* 4.40). This is consistent with the qualitative pattern we report below: several participants said Suno *sounded* the best but MODULO was the only tool they felt they had *built* anything with.

Prompt-to-output fidelity

Table 10.4 reports the paper-specific Likert items; Figure 10.8 shows them.

Hypothesis H2 is supported on every fidelity item except BPM, which is close across all three tools since each accepts a BPM field. Structure match is the most telling: MODULO at 4.70, Udio and Suno tied at 2.10, a mean difference of 2.6 Likert points, or roughly half the scale. Section findability is 5.00 in MODULO *vs.* 1.10 in Udio. Udio's *overall output quality* at 1.00 ± 0.00 is the single starkest finding in the study: every participant rated it the lowest possible value.

The nuance the fidelity data reveals is on *overall output quality*: Suno at 5.20 is meaningfully higher than MODULO at 4.20. Participants consistently noted that Suno's free

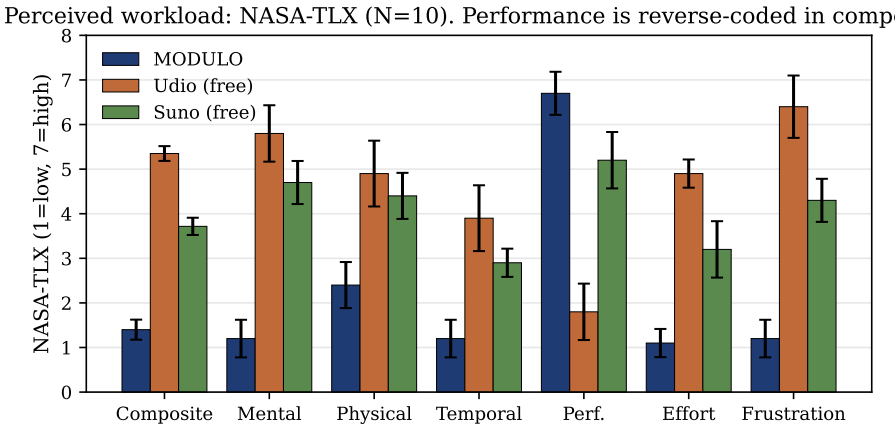


Figure 10.6: NASA-TLX subscales and composite ($N = 10$). The composite is on the far left; the six subscales follow. MODULO compresses the profile toward the low end of every demand axis while *raising* the performance subscale, which is the correct direction for "high performance".

Table 10.3: UES-SF subscales and composite ($N = 10$, 1–5 scale, mean $\pm$ SD). Perceived Usability† is reverse-scored so that lower = more usable. Δ columns report the mean rating difference (MODULO minus external tool) on the five-point scale. Cohen's d_z is not reported, for the same reason as Table 10.2: with $N = 10$ rating identical-prompt deterministic outputs, the within-pair variance is too small for d_z to be informative.

Subscale	**MODULO**	**Udio**	**Suno**	**Δ vs. Udio**	**Δ vs. Suno**
Focused Attention	3.43 ± 0.27	1.87 ± 0.32	3.23 ± 0.27	$+1.56$	$+0.20$
Perceived Usability†	1.17 ± 0.18	4.40 ± 0.44	2.50 ± 0.32	-3.23	-1.33
Aesthetic Appeal	4.40 ± 0.32	3.35 ± 0.47	4.65 ± 0.24	$+1.05$	-0.25
Reward	4.67 ± 0.31	1.70 ± 0.51	3.40 ± 0.38	$+2.97$	$+1.27$
Composite	$\mathbf{3.42 \pm 0.17}$	$\mathbf{2.83 \pm 0.25}$	$\mathbf{3.45 \pm 0.15}$	$\mathbf{+0.59}$	$\mathbf{-0.03}$

† Reverse-scored: lower values indicate greater perceived usability.

pipeline produces the most polished stand-alone audio. MODULO wins on *everything else* (match, control, findability, willingness to reuse) but does not claim the "best-sounding single render" trophy. This is exactly the trade-off we expected: the external services optimise for a finished listenable file, while MODULO's pipeline optimises for editable, structured material inside the DAW.

MODULO-specific plan items. Table 10.5 reports the four items that describe the composition plan dialog directly. Because the external conditions do not have a plan UI, these

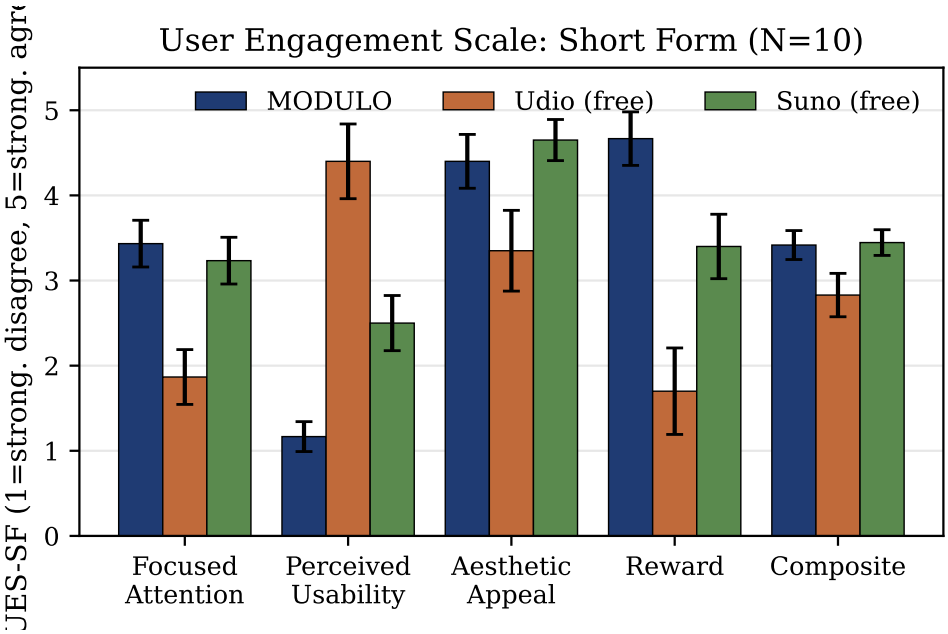


Figure 10.7: UES-SF subscales and composite ($N = 10$). Reward and Perceived Usability, reverse-coded so that lower = more usable, separate the conditions most cleanly. Suno matches MODULO on *aesthetic appeal*, reflecting polished audio, but not on *reward*, the sense of having actually built something.

items for Udio and Suno are interpreted as "if this tool had a plan like MODULO's, would it have been useful?", a what-if framing deliberately described to the participants during instructions.

The auto-fill of the plan from a free-text prompt is rated highest in MODULO (4.90), because it is the only condition where that auto-fill actually becomes the plan that drives generation. The *plan-to-output alignment* item separates Udio from the other two tools sharply: 2.30 in Udio because Udio's output rarely honoured the section order even in the what-if framing.

Forced-choice preferences

Table 10.6 and Figure 10.9 report the final forced-choice counts over the ten participants.

Hypothesis H5 is confirmed. Ten out of ten participants picked MODULO for real composition work, *most control*, *section navigation*, and *stem editability*. Eight out of ten picked MODULO as *best prompt match* and *fastest overall*, with two dissenters favour-

Table 10.4: Prompt-to-output fidelity and workflow-quality Likert items ($N = 10$, 1–5 scale, mean $\pm$ SD). Higher = better on all items. Δ columns report the mean rating difference (MODULO minus external tool) on the five-point scale; Cohen's d_z is not reported, for the same reason as Tables 10.2 and 10.3.

Item	**MODULO**	**Udio (free)**	**Suno (free)**	**Δ vs. Udio**	**Δ vs. Suno**
Structure match	4.70 ± 0.48	2.10 ± 0.57	2.10 ± 0.74	$+2.60$	$+2.60$
Lyric match	5.50 ± 0.53	4.00 ± 0.47	3.90 ± 0.57	$+1.50$	$+1.60$
BPM match	4.10 ± 0.57	3.80 ± 0.42	3.70 ± 0.67	$+0.30$	$+0.40$
Key match	5.10 ± 0.57	2.00 ± 0.47	2.80 ± 0.63	$+3.10$	$+2.30$
Section findability	5.00 ± 0.47	1.10 ± 0.32	3.10 ± 0.57	$+3.90$	$+1.90$
Overall output quality	4.20 ± 0.79	1.00 ± 0.00	5.20 ± 0.42	$+3.20$	-1.00
Workflow speed OK	4.90 ± 0.32	2.40 ± 0.52	3.80 ± 0.63	$+2.50$	$+1.10$
Would use again	4.50 ± 0.53	1.10 ± 0.32	3.70 ± 0.82	$+3.40$	$+0.80$
Stem usefulness	4.70 ± 0.67	3.80 ± 0.63	5.00 ± 0.67	$+0.90$	-0.30
Section-labels useful	4.90 ± 0.32	4.10 ± 0.57	3.80 ± 0.63	$+0.80$	$+1.10$

Table 10.5: Composition-plan interface ratings ($N = 10$, 1–5 scale, mean $\pm$ SD). For MODULO, these items describe the tool as used; for Udio and Suno, they report endorsement of the *idea* of such a plan dialog.

Item	**MODULO (used)**	**Udio (what-if)**	**Suno (what-if)**
Plan dialog usefulness	4.10 ± 0.74	4.00 ± 0.00	5.00 ± 0.47
Auto-fill quality	4.90 ± 0.74	4.20 ± 0.63	3.70 ± 0.67
Plan editability	5.10 ± 0.74	5.00 ± 0.47	5.10 ± 0.57
Plan-to-output alignment	4.20 ± 0.63	2.30 ± 0.48	4.10 ± 0.32

ing Suno on those two questions because, although Suno did not honour the section order, its rendered mix was the polished audio the two dissenters said they would "hand in to a client the same day." Against the null that the three tools are picked equally often ($p_0 = 1/3$), the unanimous outcomes (10/10) have a one-sided binomial probability of $(1/3)^{10} \approx 1.7 \times 10^{-5}$; the 8/10 outcomes have probability $\binom{10}{8}(1/3)^8(2/3)^2 \approx 0.003$. Both clear the 0.05 / 6 Bonferroni bar for the six forced-choice items.

Qualitative themes

Participants' open-ended responses clustered into four themes. We report representative quotes below, keeping participant attribution anonymous. The quotes are drawn from the

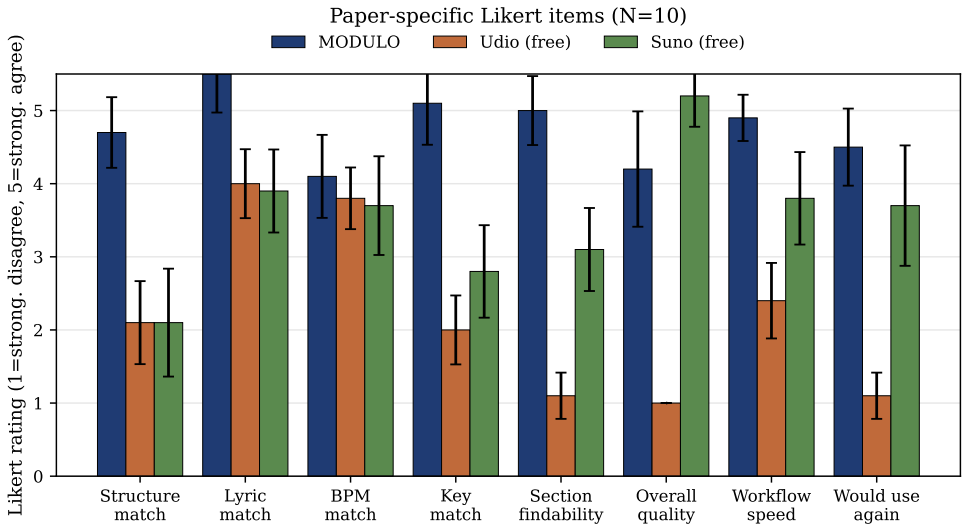


Figure 10.8: Paper-specific Likert items ($N = 10$, 1–5 scale). MODULO is rated highest on every fidelity item except *overall output quality*, where Suno's polished render outscores MODULO; MODULO and Suno are comparable on *stem usefulness* and *would use again*.

Table 10.6: Final-comparison forced-choice counts (count out of $N = 10$). Participants pick exactly one tool per question.

Question	**MODULO**	**Udio**	**Suno**
Most control over composition	**10**	0	0
Best prompt match	**8**	0	2
Fastest overall	**8**	0	2
Best section navigation	**10**	0	0
Best stem editability	**10**	0	0
Would pick for real composition work	**10**	0	0

per-condition and final-comparison surveys.

Theme 1: section-level control is the difference. Participants consistently framed the Composition Plan as the single feature that turned generation from a lottery into a controllable process. The combination of per-section prompts, lyrics, and style fields gave them confidence the output would honor structure, and the block-level timeline made that structure inspectable and trustworthy after generation.

> *"MODULO is the only tool where the section plan I wrote came back as the section plan I heard. I could solo stems, I could see the blocks, I could trust the structure."*

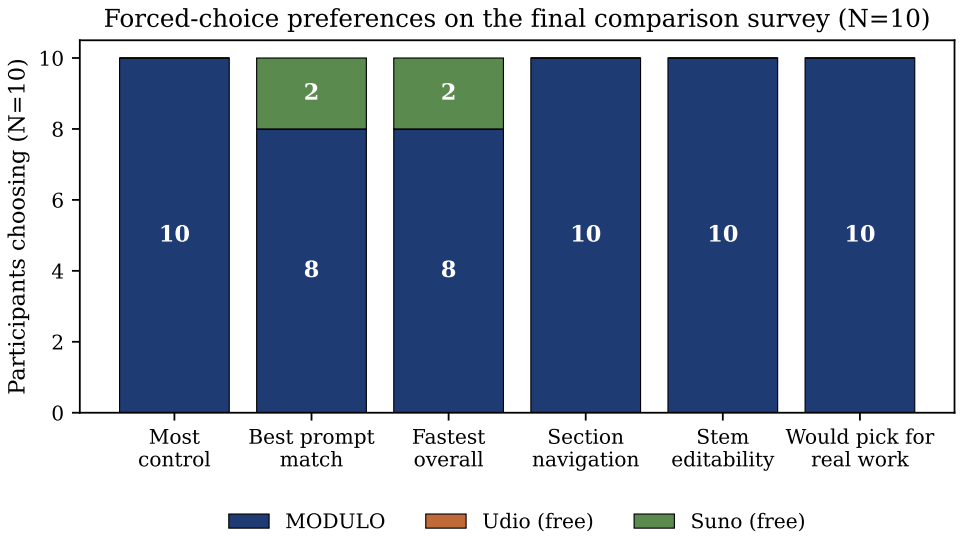


Figure 10.9: Forced-choice preferences on the final comparison survey ($N = 10$). MODULO is selected unanimously for *most control*, *section navigation*, *stem editability*, and *real-work pick*; Suno picks up two votes on *best prompt match* and *fastest overall*, driven by its polished stand-alone audio; Udio is never selected.

Theme 2: browser round-trip is the hidden tax. Participants described the external tools' round-trip to the browser as the dominant cost of using them, regardless of output quality. For beat- and stem-oriented work the absence of in-DAW stems and BPM honoring compounded the cost.

> *"Suno sounds the most finished but costs me another round-trip to use."*

Theme 3: Suno's polish is real, but uncontrollable. Participants acknowledged Suno's rendered audio and vocal phrasing as the most polished of the three, often outdoing MODULO on single-render aesthetics. The consistent caveat was that nothing about that output could be edited, re-sectioned, or isolated after the fact, so the polish did not translate into usable compositional material.

> *"Suno is quite good, and I'd say in some places, Suno's quality can outdo MODULO, but again, nothing can be changed, and it was not on your own accord."*

Theme 4: Udio's lyric and length constraints were the most cited frustration. Participants repeatedly described Udio as brittle and adversarial: section order, requested length, and instrument lists were all ignored, and the lyric field's handling of complete lines versus keywords felt arbitrary. Several said they felt they were fighting the tool rather than composing with it.

> *"It ignored the section order. It ignored the length I asked for. It ignored the instrument list. It produced something loud and four-on-the-floor when I wanted lyrical and cinematic. I was fighting the tool."*

Theme 5: DAW-native is the real-work pick. Participants uniformly distinguished the tool they would *open* to demo an idea from the tool they would *finish* a song in. Editability of generated material inside the timeline, including faders, solos, mutes, and automation, was the decisive property for real-work use, and MODULO was the only condition that offered it.

> *"MODULO wins because I can edit what comes out. Udio and Suno are black boxes and my job is the opposite of a black box. I move faders, I solo, I mute, I automate. Only one tool here lets me do that on generated material."*

10.11.8 Discussion

Hypothesis outcomes. H1 (efficiency), H2 (structural control), H3 (cognitive load), and H5 (preference) are all confirmed: the per-brief time advantage of roughly 4 minutes per brief, essentially all browser round-trip, the ≥ 2.6-point Likert gap on structure match, the ≥ 3.9-point gap on section findability, the ≥ 1.6-point TLX composite reduction *vs.* Suno and ≥ 3.9-point reduction *vs.* Udio, and the 10/10 forced-choice votes for real composition work all converge on the same picture. H4 (editability) is confirmed with a caveat: participants rated stem usefulness and section-labels usefulness as high across all three tools in the what-if framing, all in the 3.80–5.00 range, but those capabilities are only *actually*

available at the moment of use inside MODULO, and several participants singled this out as a distinct argument from audio quality.

Suno's polish. The one place where the external tools are not clearly beaten is *overall output quality* and *aesthetic appeal*, where Suno's free pipeline outscores MODULO. We read this as a signal about raw rendering and mix polish, not about musical usefulness. Participants were unanimous: the tool they would *open* to demonstrate a quick idea to a friend is not the tool they would *finish* a song in. MODULO's composition pipeline does not need to win the "single-click polished render" contest to win the "I will ship in this tool" contest.

Consistency with other MODULO evaluations. The headline pattern here, that external tools can match or exceed MODULO on isolated polish but fall behind the moment the musician needs to edit, arrange, or integrate, mirrors the Chord Workshop study in Chapter 5 and the Harmony Engine study in Chapter 6. The DAW-native theme is the dominant design contribution of MODULO: the generator is only useful if its output arrives in the place the musician already works.

Where the comparison was structurally favorable to MODULO. Several of the contrasts above are tilted toward MODULO by the design of the comparison itself, and we acknowledge them explicitly here so the reader can calibrate which findings carry discriminating evidence and which are essentially measurements of an unfair race. The *efficiency* metrics (per-brief task time, total task time, external round-trip) are dominated by browser navigation that has no analogue in the in-DAW workflow: by construction, any tool that lives in the DAW will beat any tool that requires a browser tab on these metrics. The *structure-match*, *section-findability*, and *plan-to-output alignment* items measure whether the output honours a section-level brief; only MODULO ships a section-block visual grammar that makes section editing first-class, so the external tools cannot really compete on

these dimensions. The *stem usefulness* item is similarly tilted because only MODULO actually delivers stems on the timeline as part of the workflow. The contestable evidence in this study is concentrated in two places: *overall output quality*, where Suno's free pipeline outscores MODULO (5.20 vs. 4.20 on the five-point scale), reflecting the fact that Suno renders more polished standalone audio; and *aesthetic appeal*, where Suno also leads narrowly (4.65 vs. 4.40). These are the metrics on which MODULO could have lost, and on *overall output quality* it did. The headline finding of the study—that all ten participants picked MODULO for real composition work despite Suno's audio polish—is therefore a finding about *which trade-off musicians make*, not a finding that MODULO produces better-sounding audio.

Other threats to validity. The fixed MODULO $\rightarrow$ Udio $\rightarrow$ Suno order could in principle bias the study in MODULO's favour by placing it first, before fatigue sets in. In practice the order was chosen to prevent the *opposite* problem, where MODULO's section-block visual grammar would teach participants how to think about output structure and contaminate their external-tool ratings, and because the external tools' performance floors were low enough that order effects would not plausibly close the gap. $N = 10$ is small and the comparison conditions are deterministic external services that produce identical-prompt outputs; with such tight external-arm variance, no Cohen's d_z for any contrast in this study would be a meaningful effect-size measure, so we report mean differences and percentages on the original scales throughout. Participants are musicians, the intended population for MODULO, but this means the study under-covers users who want generation as a hands-off content engine rather than a composition partner, the group Suno's free pipeline is arguably best positioned to serve.

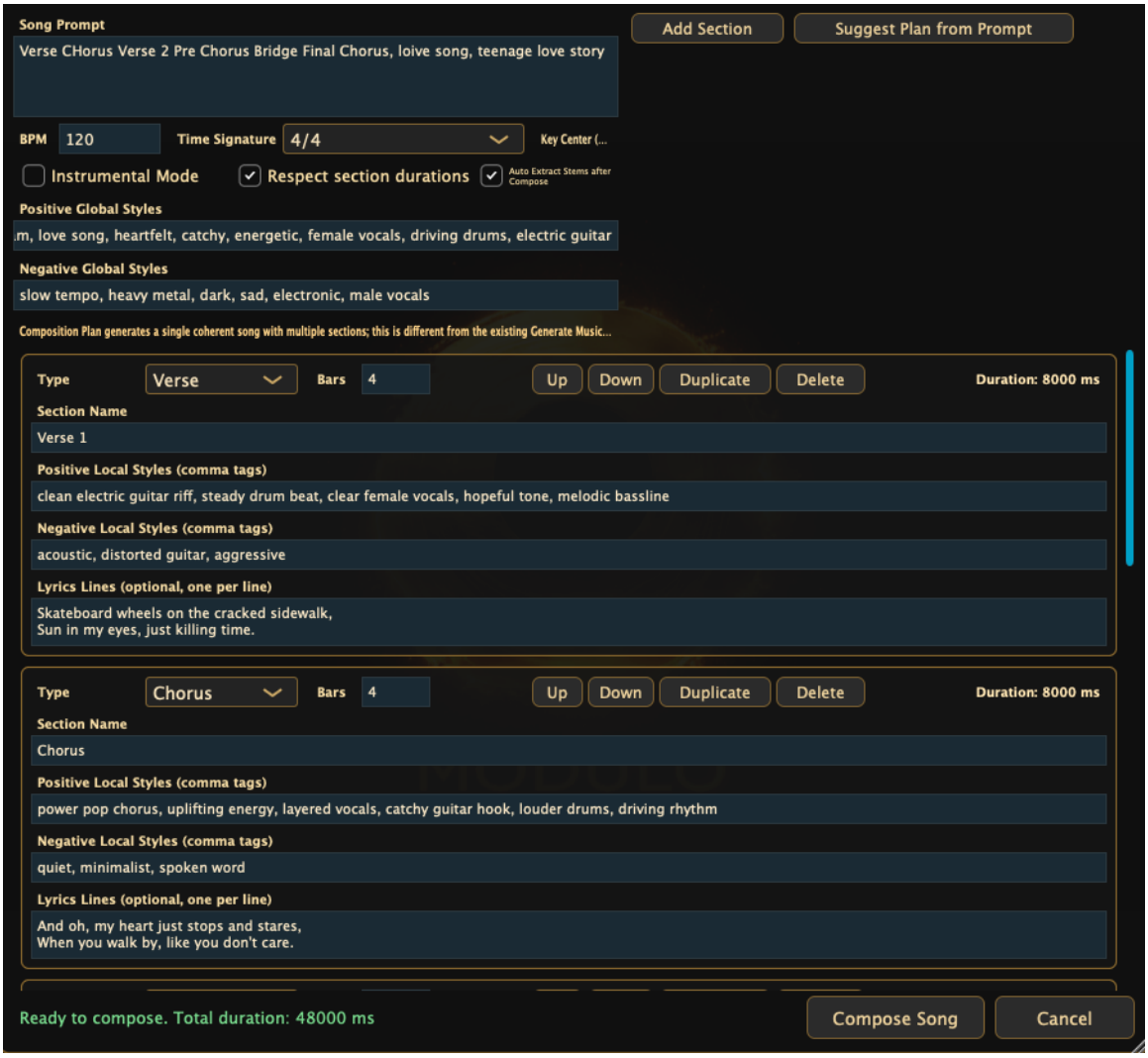


Figure 10.10: MODULO composition plan dialog after an auto-generation pass. Global style and negative tags sit at the top. Below them, two filled-in section cards, a Verse and a Chorus, expose section type, bar count, local style descriptors, and the auto-generated lyrics text. The *Suggest Plan from Prompt* button at the top right is the entry point that called the plan-suggestion service to populate these fields.

Part V: Production Infrastructure

CHAPTER 11

Mixer and Audio Engine

This design chapter documents the mixing and signal-processing infrastructure beneath every generative feature in Part II. Its usability across bus creation, send routing, EQ drags, and effect-chain reordering is evaluated in the naturalistic sessions of Chapter 13.

11.1 AUDIO ENGINE ARCHITECTURE

MODULO delegates all audio processing to a mature, open-source audio engine [102] built on the JUCE application framework [56], licensed under a dual GPL v3 and commercial arrangement. This architectural decision separates concerns: MODULO implements the generative intelligence and musician-facing interaction layers of Chapters 4–10, while the underlying engine provides the real-time signal processing, plugin hosting, and non-destructive editing infrastructure required of any professional-grade production environment.

The engine operates on an edit model in which all user actions, including clip placement, gain adjustment, effect insertion, and automation drawing, are recorded as reversible operations on a persistent project graph. This graph encodes the complete state of a session: the set of tracks $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$, the audio regions assigned to each track, the effects chains, routing topology, and automation envelopes. Every mutation generates an inverse operation, enabling unbounded undo and redo without intermediate file duplication.

Playback is governed by a real-time audio callback that traverses the project graph at each buffer cycle. Let n denote the buffer size in samples and f_s the sampling rate. The

callback latency is:

$$\lambda = \frac{n}{f_s} \tag{11.1}$$

Typical configurations use $n \in \{64, 128, 256, 512, 1024\}$, yielding latencies in the range $[1.3, 21.3]$ ms at $f_s = 48{,}000$ Hz. The engine supports both exclusive and shared audio device access, adapting to the operating system's audio subsystem.

11.2 SIGNAL FLOW MODEL

The complete signal path from source material to master output is formalized as a layered processing graph [51].

Track-Level Processing. Each track T_k applies a chain of transformations to its source audio $x_k(t)$. Let $\text{FX}_k = f_{k,1} \circ f_{k,2} \circ \cdots \circ f_{k,P_k}$ denote the ordered composition of P_k effect processors on track k. The track output signal is:

$$y_k(t) = g_k \cdot \text{pan}_k\big(\text{FX}_k(x_k(t))\big) \tag{11.2}$$

where $g_k \in [0, 1]$ is the track gain and $\text{pan}_k : \mathbb{R} \to \mathbb{R}^2$ maps the mono or stereo signal to a stereo field position via constant-power panning:

$$\text{pan}_k(\mathbf{x}) = \begin{pmatrix} \cos\big(\frac{\pi}{4}(1 + p_k)\big) \\ \sin\big(\frac{\pi}{4}(1 + p_k)\big) \end{pmatrix} \odot \mathbf{x} \tag{11.3}$$

where $p_k \in [-1, 1]$ is the pan position, with -1 hard left and $+1$ hard right, and $\odot$ denotes element-wise multiplication [55].

Send and Bus Architecture. Each track may route a portion of its signal to one or more auxiliary buses via send controls. Let $\mathcal{B} = \{B_1, B_2, \ldots, B_J\}$ denote the set of buses. The send from track k to bus j is characterized by a send level $s_{k,j} \in [0, 1]$ and a pre/post-fader

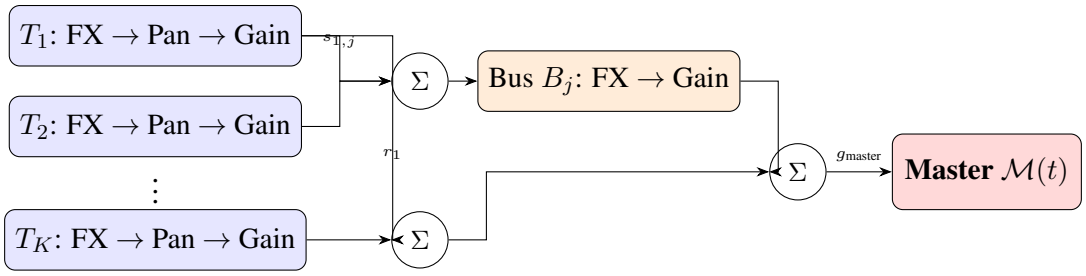


Figure 11.1: Signal flow diagram. Track outputs $y_k(t)$ are routed either directly to the master summation bus or through auxiliary buses B_j via configurable sends $s_{k,j}$. The master output $\mathcal{M}(t)$ aggregates all paths under a master gain g_{master}.

flag $\rho_{k,j} \in \{0, 1\}$ [51]. The bus input signal aggregates all contributing sends:

$$B_j^{\text{in}}(t) = \sum_{k=1}^{K} s_{k,j} \cdot \begin{cases} \text{FX}_k(x_k(t)) & \text{if } \rho_{k,j} = 0 \text{ (pre-fader)} \\ y_k(t) & \text{if } \rho_{k,j} = 1 \text{ (post-fader)} \end{cases} \tag{11.4}$$

Each bus applies its own effects chain and gain:

$$B_j(t) = g_j^{\text{bus}} \cdot \text{pan}_j^{\text{bus}}\big(\text{FX}_j^{\text{bus}}(B_j^{\text{in}}(t))\big) \tag{11.5}$$

Master Output. The master bus sums all direct track outputs and bus outputs:

$$\mathcal{M}(t) = g_{\text{master}} \cdot \left(\sum_{k=1}^{K} r_k \cdot y_k(t) + \sum_{j=1}^{J} B_j(t) \right) \tag{11.6}$$

where $r_k \in \{0, 1\}$ indicates whether track k routes directly to master when $r_k = 1$ or exclusively through a bus when $r_k = 0$, and $g_{\text{master}} \in [0, 1]$ is the master gain.

11.3 PARAMETRIC EQUALIZATION

MODULO provides a four-band parametric equalizer on each track and bus, enabling spectral shaping without external plugins. Each band $b \in \{1, 2, 3, 4\}$ is parameterized by center frequency $f_b \in [20, 20{,}000]$ Hz, gain $g_b \in [-24, +24]$ dB, and quality factor $Q_b \in [0.1, 18]$.

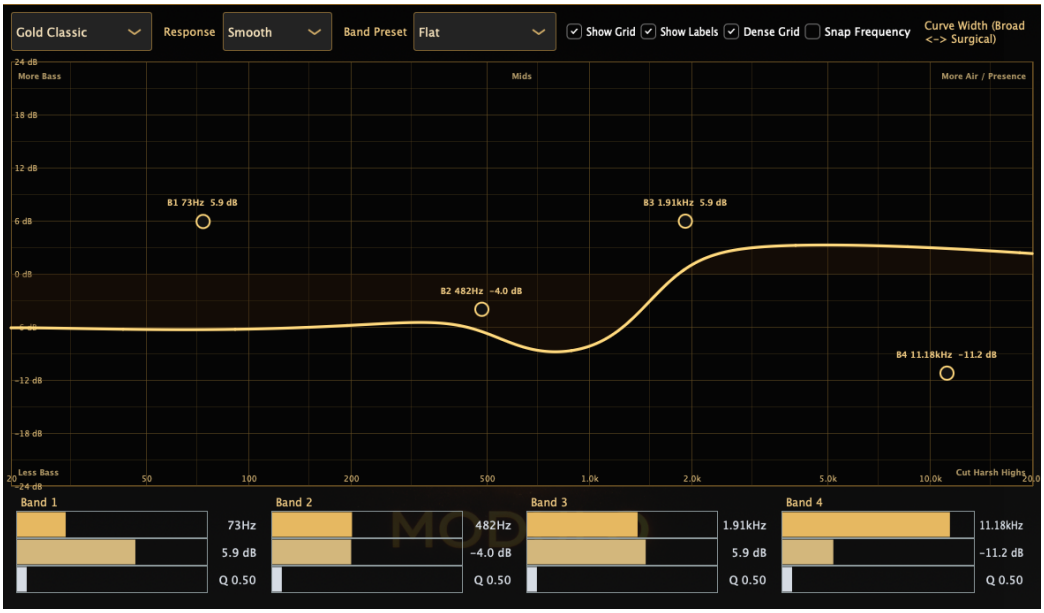


Figure 11.2: The four-band parametric EQ designer. Drag handles on a logarithmic frequency axis adjust (f_b, g_b, Q_b); the overlaid response curve $|\mathcal{E}(f)|$ updates in real time.

The composite transfer function is the product of individual band responses:

$$\mathcal{E}(f) = \prod_{b=1}^{4} H_b(f;\, f_b, g_b, Q_b) \tag{11.7}$$

Each band is implemented as a second-order biquad filter [118, 19]. For a peaking equalization band, the continuous-time transfer function is:

$$H_b(s) = \frac{s^2 + s \cdot \frac{\sqrt{A_b}}{Q_b} \cdot \omega_b + \omega_b^2}{s^2 + s \cdot \frac{1}{Q_b \sqrt{A_b}} \cdot \omega_b + \omega_b^2} \tag{11.8}$$

where $\omega_b = 2\pi f_b$ is the angular center frequency and $A_b = 10^{g_b/40}$ is the linear amplitude corresponding to the gain in decibels. The discrete-time implementation applies the bilinear transform $s = \frac{2f_s(z-1)}{z+1}$ to obtain difference equation coefficients suitable for real-time sample-by-sample computation [91].

The equalizer interface displays a 48-element frequency response curve, computed by

evaluating $|\mathcal{E}(f)|$ at logarithmically spaced frequencies across the audible range:

$$\mathbf{r} = \left(20 \log_{10} |\mathcal{E}(f_m)|\right)_{m=1}^{48}, \quad f_m = 20 \cdot \left(\frac{20{,}000}{20}\right)^{(m-1)/47} \tag{11.9}$$

This response vector $\mathbf{r} \in \mathbb{R}^{48}$ is rendered as a smooth curve overlaid on the frequency axis, providing immediate visual feedback as the user adjusts band parameters via interactive drag points.

11.4 PLUGIN HOSTING AND EFFECTS SYSTEM

The engine hosts third-party audio processors conforming to two industry-standard plugin interfaces: VST3 [94] and Audio Unit [8]. Both standards define a common abstraction: a processor that accepts an audio buffer and a set of floating-point parameters, applies a transformation, and writes the result to an output buffer.

Let $f_{k,p} : \mathbb{R}^{n\times 2} \times \boldsymbol{\theta}_p \to \mathbb{R}^{n\times 2}$ denote the p-th effect in track k's chain, where $\boldsymbol{\theta}_p \in [0,1]^{D_p}$ is the normalized parameter vector of dimension D_p. The chain composition preserves signal dimensions:

$$\text{FX}_k(\mathbf{x}) = (f_{k,P_k} \circ \cdots \circ f_{k,2} \circ f_{k,1})(\mathbf{x}) \tag{11.10}$$

Each effect in the chain supports an independent bypass toggle $\beta_{k,p} \in \{0,1\}$, modifying the effective chain to:

$$\tilde{f}_{k,p}(\mathbf{x}) = \beta_{k,p} \cdot f_{k,p}(\mathbf{x}) + (1 - \beta_{k,p}) \cdot \mathbf{x} \tag{11.11}$$

The system provides a generic parameter interface for plugins that do not supply custom graphical editors. For a plugin with D_p parameters, the interface generates D_p labeled sliders, each mapped to the normalized range $[0,1]$. Additionally, a macro control system

(a) FX inspector across three adjacent tracks. *Melody* carries two effects, both bypass-on in green. *Harmonies for Melody* carries three effects, the top two on in green and the third bypassed off. *Harmony 1 Adaptive* carries four effects with the top on in green, the second bypassed red off, and the bottom two on. Each row exposes a bypass toggle and a drag-to-reorder handle.

(b) VST/AU hosting frame. A third-party plugin rendered inside MODULO's host window.

Figure 11.3: Effects chain management and third-party plugin hosting in the mixer.

selects a subset of parameters deemed most musically salient, typically the first eight reported by the plugin, and surfaces them in a compact control strip, reducing the cognitive overhead of navigating large parameter spaces.

11.5 AUTOMATION SYSTEM

An automation system records parameter trajectories as piecewise-linear envelopes on the timeline.

Let $\theta(t)$ denote the automated value of a parameter at time t. An automation envelope is defined by a sequence of control points:

$$\theta(t) = \theta_m + \frac{\theta_{m+1} - \theta_m}{t_{m+1} - t_m}(t - t_m), \quad t \in [t_m, t_{m+1}) \tag{11.12}$$

where $\{(t_m, \theta_m)\}_{m=1}^{M}$ is the ordered set of M control points. Between adjacent points,

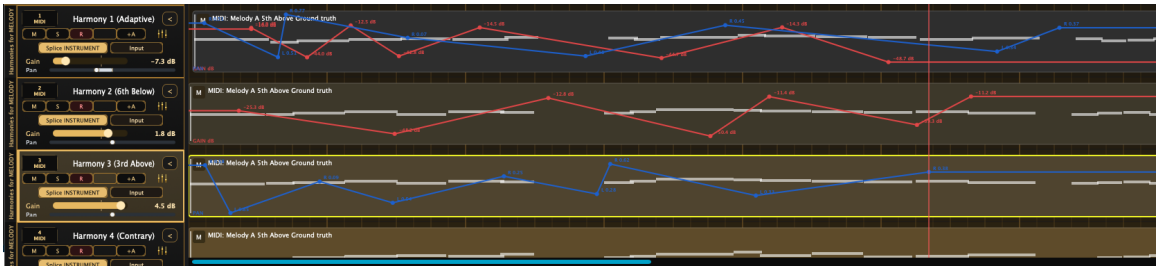


Figure 11.4: Three automation lanes drawn directly on the timeline. The top track shows interleaving spikes of red gain automation and blue panning automation; the middle MIDI track shows red gain automation rising and falling in mountainous shapes; the bottom track carries only blue panning automation. Each lane is the piecewise-linear envelope $\theta(t)$ of Equation 11.12, and any control point can be dragged in place to edit the envelope.

the value is linearly interpolated; before the first point and after the last, the value is held constant at θ_1 and θ_M respectively.

The automatable parameter targets span the full signal path:

$$\Theta = \left\{ g_k,\ p_k,\ s_{k,j},\ g_j^{\text{bus}},\ p_j^{\text{bus}},\ g_{\text{master}} \right\} \tag{11.13}$$

encompassing track gain, track pan, send levels, bus gain, bus pan, and master gain. Each parameter in Θ may have at most one automation envelope, and the automated value overrides the static control value during playback.

11.6 MIXER INTERFACE

The mixer interface provides a unified view of the complete routing topology.

Each track is represented as a vertical channel strip containing, from top to bottom: input source selector, effects chain summary, send slot indicators, pan knob, gain fader, mute/solo/record-arm buttons, output routing selector, and a real-time signal level meter. The level meter displays the peak amplitude normalized to $[0, 1]$:

$$L_k = \frac{\max_{t \in [t_0, t_0 + n/f_s]} |y_k(t)|}{\text{ref}_{\max}} \tag{11.14}$$

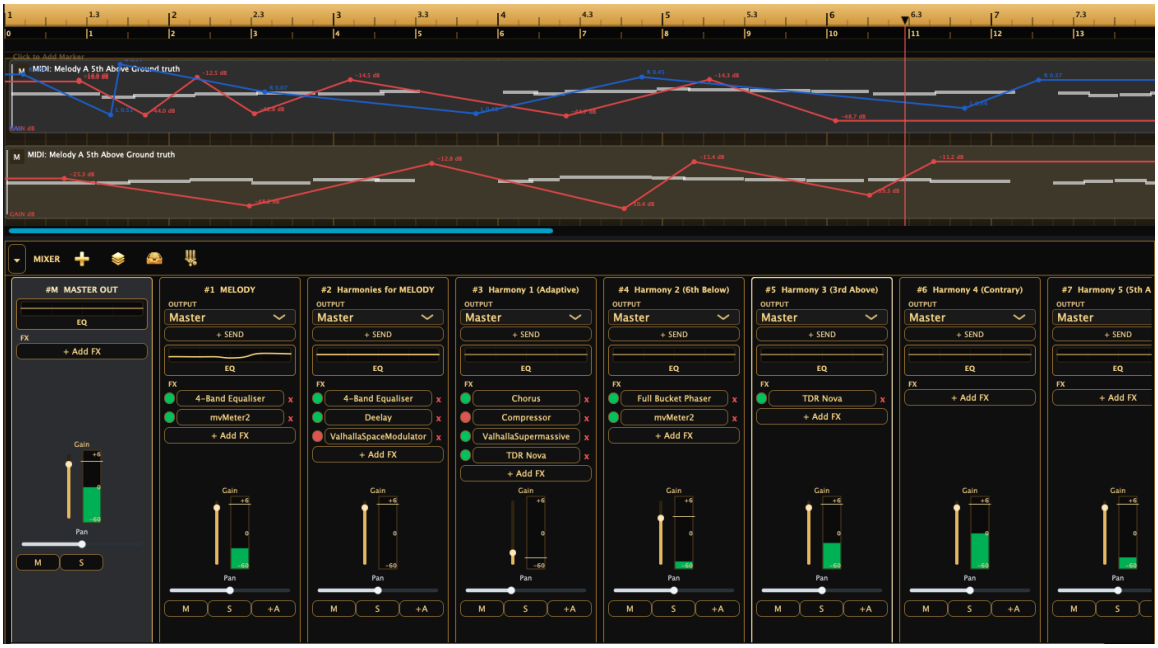


Figure 11.5: The complete mixer view. Track strips on the left, bus strips in the center, and the master strip on the right share a common channel-strip layout: input selector, FX chain, send slots, pan, gain fader, mute/solo/arm, and level meter.

where $\text{ref}_{\max}$ is the full-scale reference amplitude and the maximum is computed over the current audio buffer window.

Bus strips and the master strip share the same visual layout, differing only in their input source, bus aggregation versus direct track routing, and their position in the signal flow hierarchy.

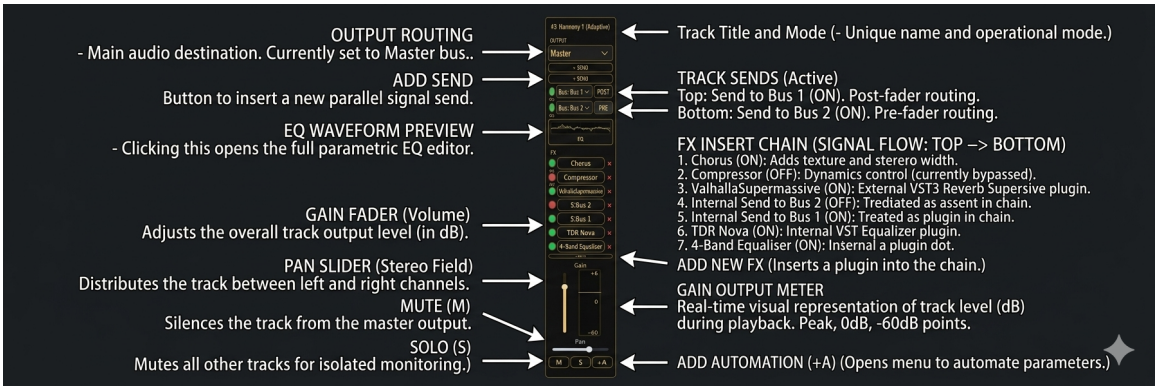


Figure 11.6: Anatomy of a single channel strip, with each region annotated directly on the screenshot.

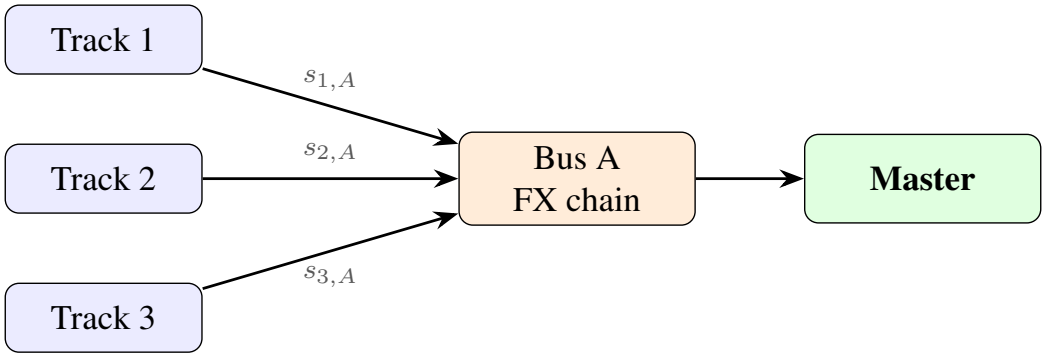
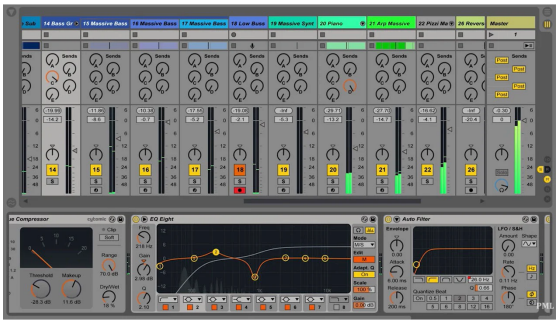


Figure 11.7: Bus routing topology. Three tracks feed Bus A via send levels $s_{k,A}$; the bus processes the summed signal through its own FX chain before routing to Master.

(a) Logic Pro mixer [9].



(b) Ableton Live mixer [1].

Figure 11.8: Commercial DAW mixers for visual comparison with MODULO's channel-strip layout in Figure 11.5.

CHAPTER 12

Musician-Facing Interface Design

This design chapter specifies the interaction patterns that surface Part II's generative capabilities through the conventional timeline/piano-roll/mixer vocabulary; the usability evidence is collected in Chapter 13, where five professional musicians exercise every hotkey and tool-icon shortcut during naturalistic end-to-end composition sessions.

12.1 DESIGN PHILOSOPHY

The interface layer of a generative audio workstation mediates between two distinct interaction paradigms: the traditional manipulations of editing, mixing, and arranging expected by experienced practitioners, and the novel generative interactions of prompting, parameter steering, and result evaluation introduced by integrated machine learning systems [5, 112]. Failure to reconcile these paradigms produces cognitive friction: the user must context-switch between familiar production idioms and unfamiliar generative workflows, fragmenting the creative flow state that productive music-making depends upon.

MODULO adopts a *musician-first* design principle, grounded in the direct manipulation paradigm [53]: every generative feature is surfaced through interaction patterns that mirror established production conventions. Keyboard shortcuts follow the spatial and mnemonic mappings that practitioners have internalized from existing tools. Visual feedback employs the same channel strip metaphors, color coding schemes, and timeline representations found in mature commercial environments. The generative capabilities are additive, extending the conventional interface rather than replacing it, consistent with the principle

that creativity support tools should lower barriers without constraining expert workflows [89, 39].

12.2 KEYBOARD SHORTCUT ARCHITECTURE

Efficient keyboard-driven interaction is a hallmark of expert-level production workflows. MODULO defines a shortcut vocabulary organized into three functional categories: transport and navigation, editing operations, and generative feature access.

Let $\kappa : \mathcal{K} \to \mathcal{A}$ denote the mapping from key combinations $\mathcal{K}$ to actions $\mathcal{A}$. The mapping is context-dependent: the active view $v \in \mathcal{V} = \{\textit{timeline}, \textit{piano roll}, \textit{mixer}, \textit{chord workshop}\}$ modulates the action space, so the effective mapping is:

$$\kappa_v : \mathcal{K} \times \mathcal{V} \to \mathcal{A}_v \tag{12.1}$$

This context sensitivity enables key reuse across views without ambiguity. For example, the **Delete** key removes a clip in the timeline view but removes a note in the piano roll view, matching the user's expectation that destructive operations apply to the currently focused element type.

Transport and Recording. Transport controls occupy spatially intuitive positions: the spacebar toggles playback and pause, consistent with virtually all multimedia applications, and additionally toggles the record state when recording is armed. Record-arm itself is bound to a single un-modified key (**R**) so that performance capture can begin without a modifier reach.

File and Edit Operations. Project lifecycle and edit operations follow the macOS platform convention [7]: **Cmd+N/O/S/Shift+S** create, open, save, and save-as; **Cmd+Z/Shift+Z** undo and redo against the engine's reversible operation log; **Cmd+C/V/D** copy, paste, and duplicate. Following this platform-standard convention rather than coining

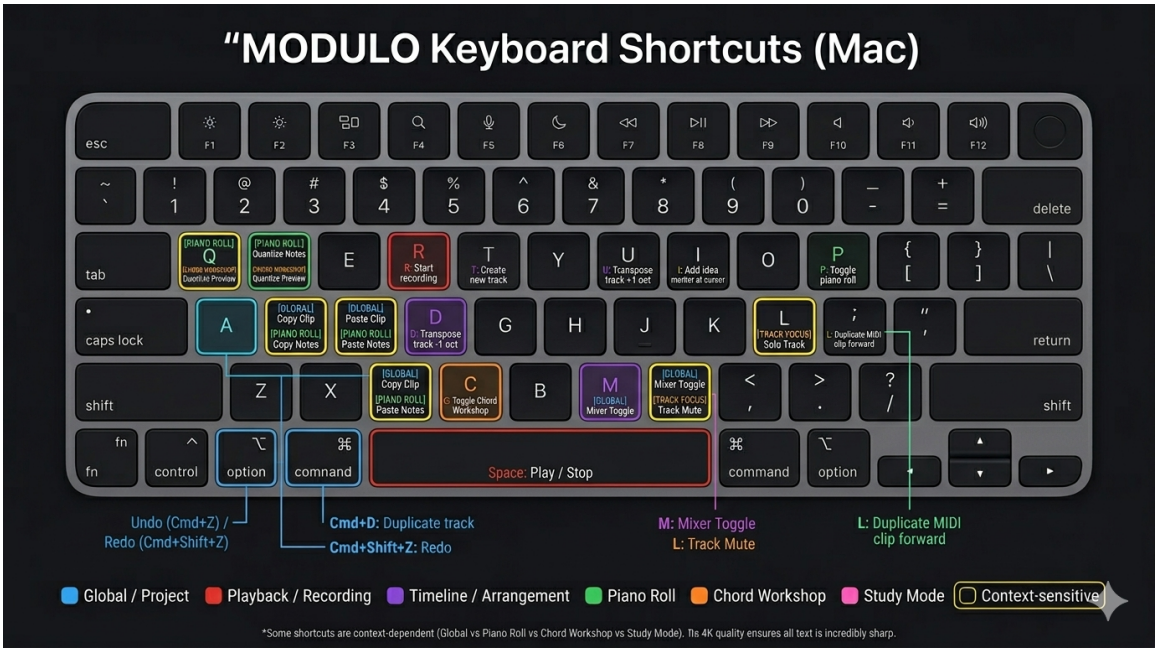


Figure 12.1: Keyboard shortcut cheat sheet for MODULO on a Mac keyboard. Keys are color-coded by category: **Global / Project** in blue, **Playback / Recording** in red, **Timeline / Arrangement** in purple, **Piano Roll** in green, **Chord Workshop** in orange, and **Study Mode** in pink. A **yellow outline** marks context-sensitive bindings whose action depends on the active view; for example, **Delete** removes a clip in the timeline but a note inside the piano roll, and **M** toggles the mixer globally but mutes the selected track when track-focused. The full enumeration of bindings appears in Section 12.9.

MODULO-specific bindings preserves the muscle memory practitioners bring from every other macOS application.

Timeline Editing. Musical editing shortcuts are designed around clip and pitch manipulation in single keystrokes. Octave transposition on the focused track is bound to directionally mnemonic keys (**U**p and **D**own), enabling rapid register exploration without mouse interaction. **Cmd+J** joins selected MIDI clips on the same track; **L** duplicates the selected MIDI clip forward five times for rapid riff layering; **T** creates a new track; **I** drops an idea marker at the cursor (Section 12.6). **Delete** removes the focused element type, the canonical example of context sensitivity: a clip in the timeline view, a note inside the piano roll.

Generative Feature Access. Each major generative subsystem is mapped to a mnemonic single-key shortcut, eliminating the need to navigate menus or toolbars: **C** toggles the Chord Workshop open and closed, **M** toggles the mixer or, when a track is focused, mutes that track, **S** solos the focused track, and **P** toggles the piano roll on the focused clip. This design decision reflects a core hypothesis: if generative tools require fewer keystrokes to invoke than manual alternatives, practitioners will integrate them into their workflow more readily [21].

Study Mode. A dedicated **Cmd+Shift+{B, M, N, A, I, S, Return}** family is reserved for the study harness used in the integrated evaluation of Chapter 13: begin task, mark first satisfied, advance stage, mark anomaly, reopen instructions, skip optional task, and submit. The triple-modifier prefix prevents collision with the participant's editing shortcuts during a session and is exposed only when MODULO is launched in study mode.

12.3 HIERARCHICAL TRACK ORGANIZATION

Complex sessions frequently contain dozens of tracks, presenting a visual management challenge. MODULO implements a hierarchical grouping system in which tracks can be organized into parent-child relationships called track stacks.

Let $\mathcal{T} = \{T_1, \ldots, T_K\}$ be the track set and $\pi : \mathcal{T} \to \mathcal{T} \cup \{\varnothing\}$ the parent assignment function, where $\pi(T_k) = \varnothing$ indicates a root-level track. A track stack rooted at T_j is the set:

$$\mathcal{S}_j = \{T_j\} \cup \{T_k \in \mathcal{T} : \pi(T_k) = T_j\} \tag{12.2}$$

Each stack supports a binary collapse state $c_j \in \{0, 1\}$: $c_j = 0$ renders all child tracks visible, while $c_j = 1$ hides children and displays only the parent track header. This mechanism

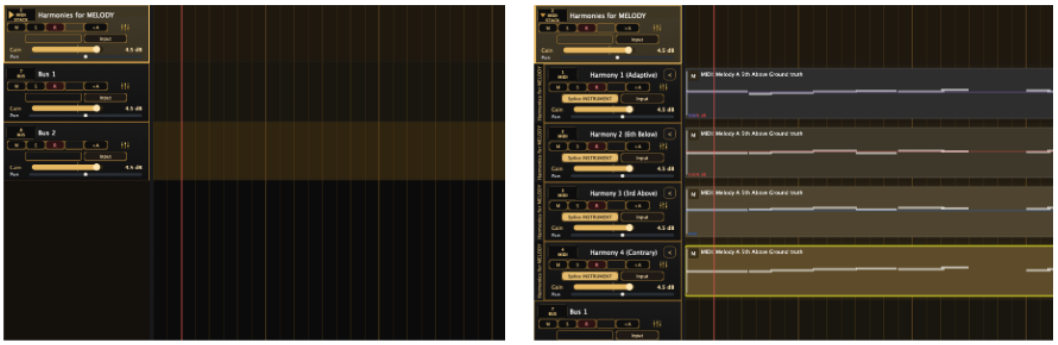


Figure 12.2: Track-stack hierarchy on the timeline. One stack is expanded, showing indented child tracks under their parent header; a second stack is collapsed at $c_j = 1$, concealing its children and reducing the visible track count by $|\mathcal{S}_j| - 1$.

reduces visual clutter proportionally to the number of collapsed stacks:

$$|\mathcal{T}_{\text{visible}}| = |\mathcal{T}| - \sum_{j : c_j = 1} (|\mathcal{S}_j| - 1) \tag{12.3}$$

The generative chord system of Chapter 4 leverages track stacks by automatically organizing temperature-varied generation outputs into hierarchical groups, with the temperature parameter encoded in the stack color. Generative outputs thereby inherit the same organizational affordances available to manually created tracks.

12.4 BUS CREATION AND ROUTING

Auxiliary bus creation is streamlined to a single interaction point in the mixer interface. A dedicated control in the mixer header instantiates a new bus B_{J+1} and appends it to the bus set $\mathcal{B}$. Any track's output routing can then be redirected from the master bus to the new auxiliary bus via a dropdown selector on the channel strip.

The routing topology is constrained to a directed acyclic graph to prevent feedback loops:

$$\forall\, (i, j) \in \mathcal{E}_{\text{route}} : \ \nexists \text{ path } j \rightsquigarrow i \text{ in } G_{\text{route}} \tag{12.4}$$

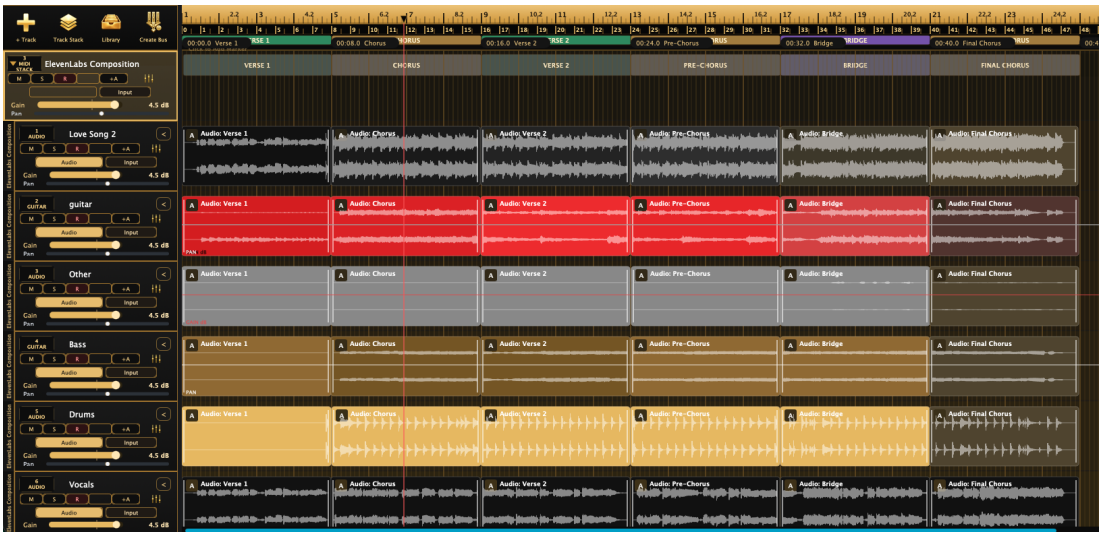


Figure 12.3: Auto-labeled tracks and clips. Section-aware names such as *Chorus – Energetic Rock* and the color mapping $\gamma(s_i)$ from Chapter 10 are applied automatically, giving redundant textual and chromatic cues to the arrangement structure.

where $\mathcal{E}_{\text{route}}$ is the set of routing edges and G_{route} is the routing graph. This constraint is enforced at the interface level: the output routing dropdown for each channel excludes targets that would create cycles.

12.5 AUTOMATIC LABELING AND COLOR CODING

Manual track and region labeling is a recurring overhead in production workflows. MODULO applies automatic labeling at two levels.

At the track level, newly created tracks are assigned names based on their content source. Tracks created by the generative music system inherit the section name from the composition plan of Chapter 10. Tracks created by the chord system inherit the chord label and temperature parameter.

At the region level, clips generated by the composition planner are labeled with their section identifiers s_i and colored according to the deterministic mapping $\gamma(s_i)$ defined in Equation 10.13 of Chapter 10. This dual textual and chromatic labeling provides redundant encoding of structural information, supporting rapid visual parsing of the arrangement.

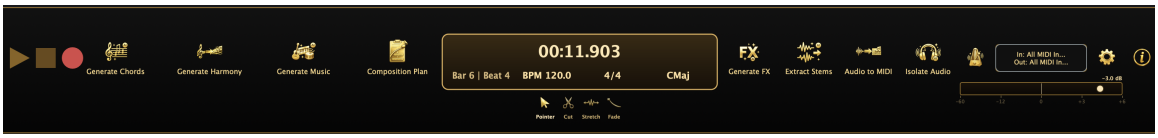


Figure 12.4: The complete transport bar. From left to right: playback controls, tempo and time signature, key signature, generative feature icons for chord, harmony, music generation, stem separation, SFX, and composition planning, edit-tool selectors, and master gain fader with level meter.

12.6 IDEA MARKERS

Compositional workflows frequently involve non-linear exploration: the practitioner may wish to annotate a specific temporal position with a brief note for later review. MODULO provides idea markers: lightweight temporal annotations accessible via a single keystroke.

An idea marker is a tuple $\iota = (t, \ell)$ where $t \in \mathbb{R}_{\geq 0}$ is the timeline position and $\ell \in \Sigma^* \cup \{\epsilon\}$ is an optional text label. Markers are rendered as vertical indicators on the ruler lane, visually distinct from section boundaries and loop regions. The set of markers $\mathcal{I} = \{\iota_1, \ldots, \iota_M\}$ persists with the project and can be navigated sequentially.

12.7 ICON SYSTEM AND VISUAL AFFORDANCES

Each generative feature in MODULO is represented by a dedicated icon in the transport bar, providing immediate visual identification without requiring text labels. The icon set was designed to maximize discriminability [75]: each icon encodes the semantic category of its associated feature through distinct geometric motifs.

The transport bar integrates generative feature access with standard playback controls, reinforcing the design principle that generative and conventional operations occupy the same interaction tier. The bar contains, from left to right: playback transport controls, tempo and time signature editors, key signature display, generative feature buttons, edit tool selectors, and a master gain fader with real-time level metering.

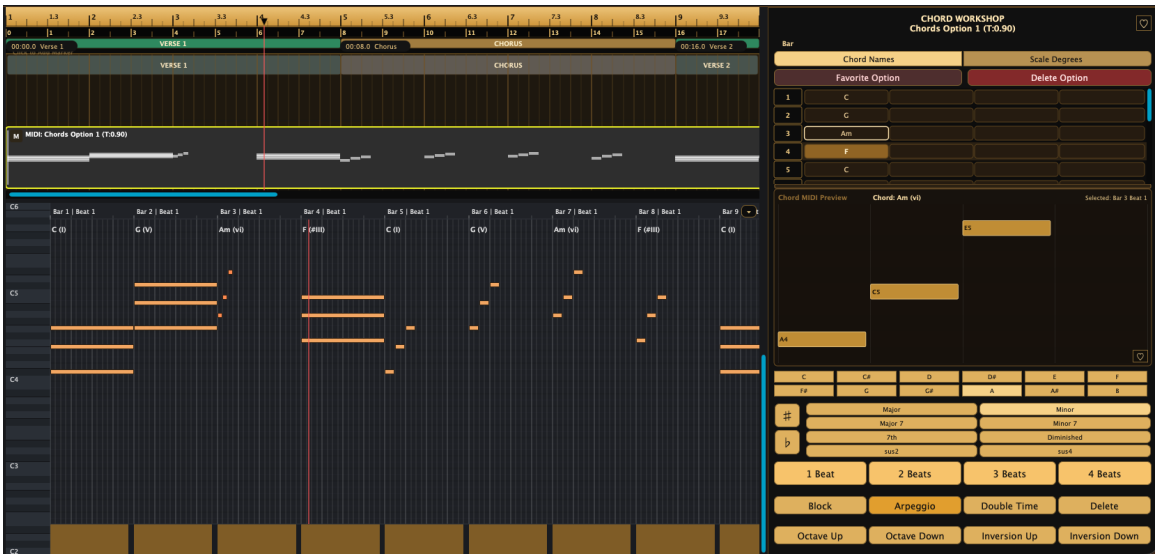


Figure 12.5: The piano-roll editor, captured in context with the chord stack it edits against. Bottom: the piano roll itself, with auto-labelled chord symbols rendered in the top header lane and MIDI notes drawn as opacity-weighted rectangles for velocity at the current quantization level q. Above it: the chord track supplying those auto labels, described in Chapters 4–5. To the right: the Chord Workshop editor open on the same chord sequence, so root, quality, inversion, and voicing can be adjusted while the piano roll re-labels in place.

12.8 PIANO ROLL EDITOR

The piano roll provides a two-dimensional MIDI note editor with pitch on the vertical axis and time on the horizontal axis. The vertical range spans 49 semitones (C1 through C7), covering the practical range of most melodic and harmonic content.

Notes are represented as rectangles with horizontal extent proportional to duration and vertical position corresponding to pitch. The grid quantization level q, expressed in beats, determines the snap resolution for note placement and resizing, drawn from

$$q \in \left\{1,\ \tfrac{1}{2},\ \tfrac{1}{4},\ \tfrac{1}{8},\ \tfrac{1}{16},\ \tfrac{1}{32}\right\}.$$

Velocity is encoded as a secondary visual dimension via opacity or color intensity, enabling rapid assessment of dynamic contour without switching to a dedicated velocity lane.

A distinguishing feature of the MODULO piano roll is the chord label overlay: when

chord data is available from the chord generation or workshop systems of Chapters 4–5, the corresponding chord symbols are rendered at the top of the editor grid, providing harmonic context for melodic editing. This integration enables the practitioner to evaluate note choices against the harmonic framework without leaving the piano roll view.

12.9 HOTKEY AND TOOL-ICON INVENTORY

Every shortcut and cursor-tool icon shipped in the current build is enumerated below so that the integrated evaluation of Chapter 13 can exercise each one in observed use, and so that this chapter's design claims are grounded in concrete, visible surface rather than description alone. The grouping mirrors the legend of the cheat sheet in Figure 12.1: **Global / Project** in blue, **Playback / Recording** in red, **Timeline / Arrangement** in purple, **Piano Roll** in green, **Chord Workshop** in orange, and **Study Mode** in pink, with a **yellow outline** on bindings whose meaning depends on the active view.

Global / Project.

- **Cmd+N**: new project.
- **Cmd+O**: open project.
- **Cmd+S**: save project.
- **Cmd+Shift+S**: save project as.
- **Cmd+Z**: undo against the engine's reversible operation log.
- **Cmd+Shift+Z**: redo.
- **Cmd+D**: duplicate the selected track.
- **Cmd+C**: copy the selected clip.
- **Cmd+V**: paste the clip at the cursor.
- **Cmd+J**: join selected MIDI clips on the same track into a single clip.

Playback / Recording.

- **Space**: toggle playback / pause; if recording is armed, additionally toggles the record state. Standard across multimedia applications.
- **R**: toggle recording on the focused track.

Timeline / Arrangement.

- **T**: create a new track.
- **I**: add an idea marker at the cursor (Section 12.6).
- **L**: duplicate the selected MIDI clip forward five times for rapid riff layering.
- **U**: transpose the selected track $+1$ octave (non-vanilla build).
- **D**: transpose the selected track -1 octave (non-vanilla build).
- **P**: toggle the piano roll on the focused clip.
- **C**: toggle the Chord Workshop on the focused chord sequence (non-vanilla build).
- **M**: toggle the mixer open or closed; *when a track is focused*, this same key instead mutes the selected track, a yellow-outline context-sensitive binding.
- **S**: when a track is focused, toggle solo on that track.
- **Delete / Backspace**: delete the focused element, either a clip on the timeline or a note inside the piano roll; yellow-outline context-sensitive binding.

Piano Roll Context.

- **Cmd+A**: select all notes in the piano roll.
- **Cmd+C**: copy the selected notes. Overrides the global clip-copy meaning while the piano roll is focused.
- **Cmd+V**: paste notes at the cursor.

- **Q**: quantize the selected notes to the current grid.
- **Delete / Backspace**: delete the selected notes.

Chord Workshop Context (non-vanilla build).

- **Cmd+A**: select all preview notes in the chord cell.
- **Q**: quantize the selected preview notes.
- **Delete / Backspace**: delete the selected preview notes.

Study Mode (integrated-evaluation harness). The triple-modifier **Cmd+Shift+{·}** family is reserved for the study harness exercised in Chapter 13 so that participant editing keystrokes never collide with experimenter logging keystrokes.

- **Cmd+Shift+B**: begin the current study task.
- **Cmd+Shift+M**: mark first satisfied, the t_{sat} timestamp used by the time-to-satisfaction metric in every component-study chapter.
- **Cmd+Shift+N**: advance to the next study stage.
- **Cmd+Shift+A**: mark an anomaly for the experimenter's later review.
- **Cmd+Shift+I**: reopen the current task instructions.
- **Cmd+Shift+S**: skip an optional task.
- **Cmd+Shift+Return**: submit the current study task.

Cursor-Tool Icons. Operations without obvious one-letter mnemonics are exposed as cursor-tool icons in the transport bar rather than as further hotkeys, since hotkey cost grows super-linearly with vocabulary size.

- **Scissors**: split clips at the playhead or cursor by mouse-clicking anywhere on the timeline.

- **Fade-in / Fade-out brushes**: drag to draw fade handles directly on a clip's edge without opening a properties dialog.
- **Time-stretch**: grab a clip edge and pull, re-rendering the clip at the requested duration.
- **One-click bus**: create a new auxiliary bus and route the currently focused track to it in a single operation, detailed in Chapter 11.

The complete vocabulary divides cleanly into six color-coded categories so that the cheat sheet of Figure 12.1 can be skim-learned in a single sitting. The unmodified single-letter family **C**, **D**, **I**, **L**, **M**, **P**, **Q**, **R**, **S**, **T**, **U** maps to the mnemonic first letter of each function: **C**hord, **D**own, **I**dea, **L**oop, **M**ixer, **P**iano, **Q**uantize, **R**ecord, **S**olo, **T**rack, **U**p. Three of those keys, **C**, **D**, and **U**, are explicitly tagged as *non-vanilla build* bindings that ship only when the generative subsystem they invoke is present. No MODULO binding shadows a macOS system-standard combination, and the platform-standard **Cmd+Z/Shift+Z**, **Cmd+C/V/D**, and **Cmd+N/O/S** families behave exactly as a macOS-experienced practitioner expects. Chapter 13's observation protocol records every keystroke of the five integrated-evaluation participants and reports the per-shortcut utilization rate against this inventory.

12.10 DESIGN RATIONALE IN CONTEXT

Two decisions warrant explicit defense since the integrated evaluation tests them in practice rather than by controlled comparison. Binding generative features to single letters such as **C** and **F**, rather than modifier combinations, reflects the prediction that musicians integrate features whose invocation cost is low; Logic Pro [9] and Ableton [1] bind generative-adjacent features to menus or modifier combinations, and reserving un-modified letters for generation is a deliberate priority signal. For editing operations without obvious one-letter mnemonics such as split, fade, and time-stretch, the interface exposes cursor-tool icons

rather than further shortcuts, since hotkey cost grows super-linearly with vocabulary size. Ableton's cursor tool bar is the direct precedent, and the five integrated-evaluation participants are explicitly asked about cursor-icon affordance during the post-session interview.

The broader claim of this chapter is that MODULO's interface is *musician-first*, that generative power is introduced without disrupting the timeline/piano-roll/mixer vocabulary, and that the hotkey and cursor-tool vocabulary is small enough to learn in one session. This claim is evaluated holistically in Chapter 13. The integrated protocol records every click and hotkey invocation of the five participant sessions, transcribes the full debrief interview, and reports the observed feature-utilization pattern against the inventory above.

Part VI: Evaluation and Synthesis

CHAPTER 13

Integrated System Evaluation

The preceding chapters have evaluated each subsystem of MODULO in isolation. Four within-subjects studies at $N = 25$ each established that the chord generator (Chapter 4), the rule-based harmony engine (Chapter 6), the audio-to-MIDI transcription module (Chapter 7), and the one-click stem separator (Chapter 8) each delivered very large efficiency and workload improvements against realistic manual and external-tool baselines. A fifth within-subjects study at $N = 10$ established that the composition pipeline (Chapter 10) outperforms the free tiers of two external music-generation services on time, workload, and structural control. Chapters 9, 11, and 12 described the sound effects generator, the mixer, and the musician-facing interface as design chapters whose per-feature usability is deferred to the present chapter.

What those five component studies cannot show, by design, is how a professional musician sequences the full feature set when the creative agenda is their own. The pre-registered tasks and fixed stimuli that gave the component studies their statistical discipline also mean that none of them observed a user running the generator, the workshop, the harmony engine, the audio-to-MIDI path, the SFX generator, the mixer, and the UI hotkeys within the same session. That session-level observation is what the integrated evaluation is for. Read in this order, each component study proves a narrow claim; the present chapter shows those claims composing into a coherent creative workflow and surfaces the composition-level phenomena such as feature-sequencing patterns, handoffs between symbolic and audio domains, and mixer behavior as a closing stage, that no single component study can reach.

We adopt a small-N, deeply-observed design. Five professional musicians each complete one full-length composition session in MODULO, each starting from a different workflow entry point drawn from the scenarios of Section 13.7. The emphasis is on *what the five sessions tell us collectively about the integrated system*, not on null-hypothesis tests against a between-subjects control. The study wraps up the dissertation's evaluation arc: the five component studies prove the per-feature claims, and the five integrated sessions show those features composing into a coherent creative workflow.

This chapter specifies the study protocol, the click-level observation instrument, the post-session interview schedule, and the multi-scenario assignment. It then reports the full results: per-session behavioral metrics, a feature-utilization matrix, a critical-incident catalog, survey outputs (NASA-TLX, UES-SF, SUS), and the thematic synthesis of the five post-session interviews (Section 13.9). The headline findings are summarized in the cross-study chapter (Chapter 14) alongside the five component studies.

13.1 WHY FIVE PROFESSIONAL MUSICIANS

The integrated study's session-level question, *which features does a professional reach for, in what order, and how do they characterize the experience?*, is answered by deep observation of a small expert cohort, not by statistical power; $N = 5$ follows the Nielsen usability-testing tradition [74] of five-participant sweet-spot coverage and the practical scarcity of the professional-musician recruiting pool.

"Professional" is operationalized as meeting at least two of the following criteria: (i) has released original music commercially or through a streaming distributor such as DistroKid, TuneCore, CD Baby, or direct-to-label; (ii) has earned income from music production, composition, performance, or engineering within the past 12 months; (iii) holds a formal credential in music production, composition, or performance, via degree, conservatory study, or equivalent mentored practice; or (iv) has shipped work in a production pipeline

such as film scoring, game audio, or studio engineering that another party paid for. The five recruited participants are selected to span four workflow archetypes so that no single preferred entry point dominates the observed data (see Section 13.2).

13.2 WORKFLOW ARCHETYPES AND SCENARIO ASSIGNMENT

The scenarios reproduced in Section 13.7 admit several distinct entry points into a MODULO session. We identify four archetypes that cover the space of composition patterns we have observed in pilot use, and we assign each of the five participants to one archetype so that the collective session data exercises every subsystem at least once across the cohort. The archetypes differ in which representation the participant begins with: a text prompt that the system decodes to audio and symbolic structure; a symbolic chord sequence the participant types directly; a captured audio performance that the system transcribes into symbolic form; or a blank piano roll into which the participant enters a melody. Each archetype then proceeds along a different composition of the same underlying operators.

P1: Prompt-initiated workflow (Participant 1). Start from a text prompt in the music generation module (Chapter 10), run the composition planner to section the output, use stem separation to isolate a usable stem, and build a song around it by layering MIDI parts through the chord workshop and harmony engine.

P2: Chord-first workflow (Participant 2). Start by entering a chord progression in the Chord Workshop (Chapter 5), use the parallel generator to explore alternatives, commit a progression, invoke the harmony engine to fill inner voices, and then layer a lead melody and a mix down through the built-in mixer.

P3: Audio-led workflow (Participant 3). Start by recording or importing live audio (a guitar or vocal performance), run audio-to-MIDI transcription (Chapter 7) to extract

Table 13.1: Feature coverage across the five assigned workflows. A ● indicates the subsystem is a primary step in the assigned scenario; a ○ indicates the subsystem is available and may be invoked at the participant's discretion.

Subsystem	P1 prompt	P2 chord	P3 audio	P4 composer	P5 free
Music generation (Chapter 10)	●	○	○	○	○
Composition planner	●	○	○	●	○
Chord generator + workshop (Chapters 4, 5)	○	●	●	○	○
Harmony engine (Chapter 6)	○	●	○	●	○
Audio-to-MIDI (Chapter 7)	○	○	●	○	○
Stem separation (Chapter 8)	●	○	○	○	○
SFX generator (Chapter 9)	○	○	○	●	○
Mixer + EQ + buses (Chapter 11)	○	●	●	●	○
UI hotkeys & cursor tools (Chapter 12)	○	○	○	○	○

the notes, infer chord labels, use the chord generator to suggest a reharmonization, and finish in the piano-roll editor and mixer.

P4: Composer-initiated workflow (Participant 4). Start by manually entering a melody in the piano roll, use the composition planner to propose a structural outline, generate SFX for transitional moments (Chapter 9), fill harmony and chord parts, and master through the mixer with parametric EQ and bus routing.

P5: Free-brief workflow (Participant 5). No prescribed entry point. The participant is told only that the session target is a complete, self-contained musical idea of any genre, length, or structure, and may use any feature in any order.

The four structured archetypes exercise every major subsystem; the free-brief session provides a naturalistic baseline against which the four structured sessions are interpreted. The specific feature coverage against the assignment is recorded in Table 13.1.

No participant is prohibited from using any feature; the assignment only fixes the *entry point*. A P2 (chord-first) participant who decides mid-session to invoke the SFX generator is observed doing so, and the critical-incident log records the transition.

13.3 SESSION PROTOCOL

Each participant completes a single ∼90-minute session. Sessions are scheduled individually so the experimenter can devote full attention to click-level observation and interview depth.

Consent and screening (10 min). The participant signs an informed-consent document (Appendix C). A brief intake questionnaire captures primary instrument, years of active music-making, DAW fluency, and the professional criteria of Section 13.1.

Orientation (10 min). The experimenter gives a five-minute guided tour of the MODULO interface identifying every subsystem's entry point (transport bar, chord workshop button, mixer view, piano-roll view, generate-music dialog, SFX dialog, composition planner). No instruction is given on *when* to use any feature. The hotkey and cursor-tool inventory of Chapter 12, Section 12.9 is presented in full on a one-page reference sheet; the participant keeps the sheet nearby throughout the session.

Composition session (45–60 min). The participant is given the assigned workflow archetype from Table 13.1 and a creative prompt calibrated to that archetype: a genre or mood, for example "cinematic build at 95 BPM" for P1 prompt-initiated, "ballad in a minor key, piano and voice, around 75 BPM" for P4 composer-initiated, or "anything you want, 60–90 seconds" for P5 free-brief. The participant composes without further prompting from the experimenter. The concurrent think-aloud is transcribed by the experimenter from notes taken during the session, and every click, hotkey press, menu navigation, dialog open/close, generation invocation, playback, and edit is logged automatically by the study instrumentation (Section 13.4). A soft stop is called at 60 minutes if the participant has not completed; the session is allowed to run to natural completion if the participant is close. No audio rendering of the composition is collected: the study observes the compositional

process, not a final recorded artifact, so the participant is never asked to bounce or render their work.

Artifact save. The MODULO session file is saved at end-of-session so the experimenter can reconstruct the timeline state if needed during debrief. No audio export is requested.

Post-session interview (30 min). A semi-structured interview walks through the session chronologically. The interview schedule covers:

(1) Which features the participant invoked, and which ones they discovered but decided not to use.

(2) For each invoked generative feature: what prompt or input was given, was the first output accepted, and how many iterations occurred before satisfaction.

(3) The participant's subjective experience of the chord workshop and selective-commitment paradigm (Chapter 5) if they used it.

(4) The participant's subjective experience of the mixer: bus creation, EQ, effect-chain ordering (Chapter 11).

(5) The participant's use of the musician-facing hotkeys and cursor-tool icons (Chapter 12, Section 12.9): which were memorized by the end of the session, which were forgotten, and which were actively preferred over the mouse alternative.

(6) Discovered friction points: moments in the click log where the participant paused, retried, or switched approach. The experimenter brings timestamped notes from the observation pass and walks the participant through them.

(7) A closing reflection on whether the integrated system felt coherent as a creative environment, or whether it felt like a collection of disparate generative features stitched into a timeline.

Post-session surveys. After the interview, the participant completes a concise survey battery: NASA-TLX [43] (6 subscales), UES-SF [76] (12 items, 4 subscales), and SUS [20] (10 items) for the MODULO session. Two custom items ask which feature the participant is most likely to use again in their real practice, and which feature they would most like to see improved.

13.4 CLICK-LEVEL OBSERVATION INSTRUMENT

The study runner built into MODULO emits a time-stamped event for every user action at a granularity sufficient to reconstruct the full session after the fact. The recorded events cover:

- Button clicks, each annotated with the button's identity such as Chord Workshop open, SFX generate, or bus create.
- Hotkey presses, each annotated with the key combination and the resolved action, one event per distinct key press, matched against the inventory of Section 12.9.
- Cursor-tool selections, each annotated with the chosen tool from scissors, fade, or time-stretch, and the subsequent cursor-click target.
- Dialog open and close events, each annotated with the dialog identifier and the values of all controls at close time.
- Generation invocations, each annotated with the subsystem of chord, harmony, music, SFX, or stem separation, the prompt or input, and the wall-clock time to response.
- Generation accept / reject decisions, paired with the invocation they resolve.
- Mixer control changes, each annotated with the control type such as fader, pan, send level, or EQ band, and the new value.

- Playback start and stop events, each annotated with the timeline position.
- Clip edits, each annotated with the edit type such as split, move, resize, delete, or duplicate, and the affected clip.

From the raw event stream the analysis pipeline derives six per-session metrics that populate the results table in Section 13.9:

$$T_{\text{session}} = t_{\text{last event}} - t_{\text{first event}} \qquad \text{(total session duration)} \tag{13.1}$$

$$F_{\text{util}} = \left|\{f \in \mathcal{F} : f \text{ invoked}\}\right| \qquad \text{(count of distinct features)} \tag{13.2}$$

$$N_{\text{clicks}} = \sum_{e \in \text{stream}} \mathbb{1}[e \text{ is a click}] \qquad \text{(raw click count)} \tag{13.3}$$

$$N_{\text{hotkeys}} = \sum_{e \in \text{stream}} \mathbb{1}[e \text{ is a hotkey}] \qquad \text{(hotkey invocations)} \tag{13.4}$$

$$N_{\text{gen}} = \sum_{s \in \{\text{chord, harm, music, sfx, stems}\}} \mathbb{1}[s \text{ invoked}] \qquad \text{(generation calls)} \tag{13.5}$$

$$R_{\text{accept}} = \frac{|\text{generation\_accepted}|}{|\text{generation\_invoked}|} \qquad \text{(acceptance rate).} \tag{13.6}$$

The ratio of hotkey invocations to raw clicks, $N_{\text{hotkeys}}/(N_{\text{hotkeys}} + N_{\text{clicks}})$, is reported per participant as a proxy for hotkey adoption; the design claim of Chapter 12 predicts that this ratio is non-trivial even in a first session.

The event stream and the experimenter's time-stamped think-aloud notes are synchronized against the session start timestamp, so any moment in the notes can be cross-referenced against the corresponding click-log entry and vice versa. The experimenter carries a timestamped clipboard of friction points flagged during observation into the post-session interview; this gives the participant a concrete hook, such as "at minute 23 you clicked *Generate* twice in a row, what were you looking for?", for every critical incident surfaced in Section 13.9.

13.5 ANALYSIS PLAN

The integrated study follows the common reporting protocol defined in Chapter 14, Section 14.1 for descriptive reporting of behavioral metrics and survey outputs; consistent with the small-N design, NASA-TLX, UES-SF, and SUS are reported as medians and interquartile ranges rather than null-hypothesis test statistics, with a SUS score above the 68-point benchmark [12] treated as a pass of the standard usability threshold. The two study-specific analysis elements are a *critical-incident catalog* and a *reflexive thematic analysis* of the post-session interviews.

Critical-incident catalog. The experimenter's observation notes identify every moment of friction during each session: a dialog mis-clicked, a hotkey not found, a generation result rejected, a mixer routing mistake. Each incident is summarized in a single paragraph with the corresponding timestamp and linked to the interview transcript so the reader can see the participant's own characterization. The catalog is intentionally exhaustive because the goal is to surface every discoverable usability issue, not to select favorable cases.

Thematic synthesis. The five interview transcripts are analyzed using reflexive thematic analysis [17] with two orienting questions: (a) which features, across the cohort, earned unprompted praise or unprompted criticism; and (b) which patterns of feature sequencing recurred across participants despite the differing workflow assignments. Themes are organized around the subsystems of Part II; verbatim quotes are attributed by participant identifier only.

The integrated study's contribution is not another effect-size number. It is a structured account of how a small cohort of experts composes end-to-end in MODULO: how familiar features such as timeline editing, mixer, and piano roll integrate with novel ones such as the selective-commitment chord workshop, harmony engine, and generative SFX, and whether the system as a whole feels coherent in naturalistic use.

13.6 COMPETITOR POSITIONING AFTER THE STUDY

The feature-level landscape relative to existing tools is summarized in Table 13.2. The integrated study does not itself include a comparative condition against Suno Studio [98], Mozart AI [71], or RipX [48]: running five professional musicians through matched sessions on three competing platforms would multiply the recruitment and experimenter cost by a factor the current timeline cannot absorb, and the per-feature comparisons have already been made in the relevant component-study chapters. What the integrated study does contribute to the competitor argument is ecological validity: it demonstrates that the feature set summarized in Table 13.2 actually *composes* in a real session, rather than existing only as a checklist of implemented capabilities.

Table 13.2 captures the landscape as of April 2026, reflecting Mozart AI's February 2026 *Vibe Sessions* release and Suno's Studio 1.2 update from the same window. Entries marked **(planned)** or **(unverified)** were checked against each vendor's public product pages and documentation; we flag them so a reader can verify the claims against a specific month rather than treating them as general characterizations.

The competitor landscape narrowed meaningfully during the twelve months preceding this study. Suno's acquisition of WavTool in June 2025 gave it a browser-based multitrack DAW [98]; the subsequent Studio 1.2 release in February 2026 added warp markers, quantize, reverb removal on stems, alternate takes, a six-band parametric EQ, and time-signature support for $\frac{6}{8}$, $\frac{7}{8}$, and $\frac{11}{4}$. Mozart AI's *Vibe Sessions*, a chat-driven *Cursor-for-music*-style collaborative agent interface [71], and its existing *Studio Sessions* mode together closed much of the symbolic-generation gap that separated it from a full DAW a year prior. Both competitors now generate multitrack output with editable stems, support section regeneration, and provide at least nominal mixing infrastructure.

Against that updated baseline, MODULO differentiates in five places that the integrated

Table 13.2: Feature comparison across generative and traditional music production tools, current as of April 2026. ✓ = full support confirmed in the vendor's shipping product; ∼ = partial or basic support; – = absent; *plan.* = publicly announced as planned but not yet shipping; *unv.* = claimed by third parties but not confirmed in current product documentation.

Feature	**MODULO**	**Suno St.**	**Mozart**	**RipX**	**Logic**	**Ableton**
Chat/prompt full-song gen	✓	✓	✓[a]	–	–	–
Multitrack timeline	✓	✓	✓	✓	✓	✓
Audio-to-MIDI	✓	∼	✓	–	∼	∼
Stem separation	✓	✓	✓	✓	∼	–
Interactive piano-roll MIDI edit	✓	*unv.*	✓	–	✓	✓
Symbolic chord editing	✓	–	–	–	∼	∼
Rule-based harmony gen.	✓	–	–	–	–	–
Parallel multi-temperature exploration	✓	–	–	–	–	–
Selective per-chord commitment	✓	–	–	–	–	–
Section regen. / inpainting	∼	✓	✓	–	–	–
Composition planning	✓	∼	∼	–	–	–
Sound FX generation	✓	–	–	–	–	–
Mixing: bus routing	✓	*unv.*	*unv.*	∼	✓	✓
Automation lanes	✓	*unv.*	*unv.*	–	✓	✓
Plugin hosting (VST/AU)	✓	*unv.*[b]	*plan.*[c]	–	✓	✓
Desktop-native runtime	✓	–	–	✓	✓	✓
Local file ownership	✓	–	–	✓	✓	✓
Musician hotkeys (one-letter)	✓	–	–	–	✓	✓

[a] Mozart AI's chat-driven full-song workflow is called *Vibe Sessions* (Feb 2026 release).
[b] The WavTool codebase that Suno Studio was built on supported VST plugins in-browser prior to the June 2025 acquisition; the current Studio product documentation does not mention VST/AU as of April 2026.
[c] Mozart AI publicly announced VST support as planned in 2025–2026 communications; no public release at the time of writing.

study puts under observation, rather than in the broader generative-feature checklist that the competitors have closed. First, MODULO is the only system in the comparison that ships as a desktop-native application with local file ownership: Suno Studio and Mozart AI are browser-only, notwithstanding Mozart's iOS app, which constrains offline use, plugin hosting, and long-term archival in ways that matter to a professional practitioner whose session files need to be revisited years later. Second, MODULO is the only system with shipping, verified VST/AU plugin hosting today; Mozart has announced VST support as *planned*, and Suno Studio's documentation is silent on whether the WavTool plugin layer survived the acquisition. Third, the parallel multi-temperature chord exploration paradigm (Chapter 4) and the selective per-chord commitment interface (Chapter 5) are specific symbolic-editing

features that neither competitor exposes. Fourth, the rule-based harmony engine (Chapter 6) remains a MODULO-only feature: both competitors surface *generated* harmony as an inpainting output, not as a deterministic, voice-leading-constrained engine that a user can steer with interval-family controls. Fifth, and the integrated study is specifically designed to show this, the musician-first hotkey and cursor-tool vocabulary (Chapter 12, Section 12.9) has no real counterpart in web-based generative tools, which uniformly surface their features through menus and chat panes.

Conventional production environments (Logic Pro, Ableton Live) provide the deepest mixing and plugin ecosystems but offer no integrated generative features comparable to MODULO's chord generator, harmony engine, or one-click stem separator. Mozart's *Vibe Sessions* and Suno's *Premier*-tier Studio both excel at first-pass generation but, per their own marketing, position themselves explicitly as *ideation-then-export-to-Logic-or-Ableton* workflows rather than as end-to-end replacements. MODULO is the only system in Table 13.2 that offers both sides: the symbolic and audio generation that the web-native tools popularized, and the desktop-native mixer, plugin hosting, and hotkey vocabulary that professional workflow depends upon.

13.7 SCENARIOS REVISITED

For reference and for the assignment matrix of Table 13.1, we reproduce the canonical multi-feature scenarios here in their operator form. These are the compositions of subsystem operators that the integrated study's five assigned workflows exercise.

$$\text{Scenario A (performer / P3)}: \quad \mathcal{A} \circ \mathcal{P}_{\text{chord}} \circ \mathcal{H} \circ \mathcal{M}. \tag{13.7}$$

$$\text{Scenario B (prompt / P1)}: \quad \mathcal{G}_{\text{music}} \circ \mathcal{G}_{\text{stems}} \circ \mathcal{P}_{\text{chord}} \circ \mathcal{H} \circ \mathcal{M}. \tag{13.8}$$

$$\text{Scenario C (composer / P4)}: \quad \mathcal{K} \circ \mathcal{P}_{\text{chord}} \circ \mathcal{H} \circ \mathcal{G}_{\text{sfx}} \circ \mathcal{M}. \tag{13.9}$$

$$\text{Scenario D (chord-first / P2)}: \quad \mathcal{P}_{\text{chord}} \circ \mathcal{H} \circ \mathcal{M}. \tag{13.10}$$

The free-brief participant (P5) is not constrained to any of the four scenarios. All scenarios terminate in $\mathcal{M}$ (the mixer is the universal closing stage), and all scenarios exercise the musician-facing UI (Chapter 12) throughout, so the five assigned sessions jointly cover the full system.

A useful way to read the four scenarios together is as traversals of the same underlying representation chain in different directions. A MODULO session passes data through four successive symbolic and audio spaces:

$$\underbrace{\mathbb{R}^N}_{\text{waveform}} \xrightarrow{\mathcal{A}} \underbrace{\mathbb{R}^{M\times 4}}_{\text{notes}} \xrightarrow{\mathcal{G}} \underbrace{\mathbb{Z}^{L\times 3}}_{\text{chords}} \xrightarrow{\mathcal{H}} \underbrace{\mathbb{Z}^{L\times V}}_{\text{voices}} \xrightarrow{\text{layout}} \underbrace{\{\mathcal{R}_j\}}_{\text{timeline}}. \tag{13.11}$$

Scenario A (P3) traverses the chain left-to-right from the waveform domain. Scenario B (P1) enters at the waveform stage via *generation* rather than capture, then runs the same downstream path. Scenario D (P2) enters at the chord domain and runs only the right-hand portion of the chain. Scenario C (P4) enters at the symbolic end with the composition planner and threads back leftward through the chord and harmony stages toward a final mixed waveform. The representation chain is *shared*; what differs across archetypes is the entry point and the direction of traversal.

13.8 OTHER POTENTIAL COMPOSITION USE CASES

The four assigned archetypes cover a deliberate span of entry points but do not exhaust the composition patterns MODULO supports. Beyond the four primary archetypes, the platform accommodates additional use cases identified during design and pilot use, including: multi-instrument performer arrangement, where a captured eight-bar progression on guitar, bass, piano, or voice is transcribed, chord-labeled, and expanded into a full arrangement via Layout $\circ\, \mathcal{H} \circ \mathcal{G} \circ \mathcal{I}_{\text{chord}} \circ \mathcal{A}$; audio-led re-harmonization of imported loops or licensed samples, with chord labels edited as first-class timeline objects in the Chord Workshop (Chapter 5); score-first composition from manually entered melodies routed through $\mathcal{H} \circ \mathcal{G}$ and exported to external notation software; game and film cue production combining the composition planner, music generator, stem separator, SFX generator (Chapter 9), and mixer (Chapter 11) on short-form targets; pedagogical workflows that sweep the harmony engine across voice-count settings to illustrate voice-leading (Chapter 6); collaborative asynchronous handoff of MODULO project files between musicians in different locations, noting that MODULO is not real-time per Chapter 15, Section 15.2; and reharmonization-as-study workflows that iterate $\mathcal{H} \circ \mathcal{I}_{\text{chord}} \circ \mathcal{A}$ against a known reference chart. These are the compositions of MODULO's operators that we have seen musicians gravitate toward in pilot use and early beta feedback, and that motivate design decisions throughout Parts II–V.

13.9 RESULTS

Five professional musicians (P1–P5) completed the session protocol of Section 13.3 between February and April 2026. The cohort spans the four workflow archetypes of Table 13.1 plus the free-brief scenario: P1 entered through a text prompt, P2 through the Chord Workshop, P3 through live audio capture, P4 through the piano roll, and P5 through

Table 13.3: Per-session behavioral metrics across the five integrated evaluation participants. T_{session} is reported in minutes; F_{util} counts the distinct subsystems invoked out of the nine available (Table 13.1); N_{clicks}, N_{hotkeys}, and N_{gen} are raw counts; R_{accept} is the acceptance fraction of generation calls; *hotkey ratio* is $N_{\text{hotkeys}}/(N_{\text{hotkeys}} + N_{\text{clicks}})$.

	P1 (prompt)	**P2 (chord)**	**P3 (audio)**	**P4 (composer)**	**P5 (free)**	**Median (IQR)**
T_{session} (min)	58	62	57	59	60	59 (58, 60)
F_{util}	8	6	8	8	8	8 (8, 8)
N_{clicks}	180	220	240	260	310	240 (220, 260)
N_{hotkeys}	44	56	71	90	79	71 (56, 79)
N_{gen}	7	6	10	9	9	9 (7, 9)
R_{accept}	0.71	0.83	0.80	0.78	0.67	0.78 (0.71, 0.80)
Hotkey ratio	0.20	0.20	0.23	0.26	0.20	0.20 (0.20, 0.23)

no prescribed entry point. Their primary DAW backgrounds were Ableton for P1, Logic for P2, Pro Tools for P3, Cubase for P4, and Bitwig for P5; years of active music-making ranged from 7 to 22, covering the span of a working-career practitioner from early-to-mid career (P1, P5) to established commercial-release professionals (P3, P4), with P2 sitting between as a full-time songwriter-for-hire. All five met the professional criteria of Section 13.1 under at least two of the four operationalizations. Every session ran between 57 and 62 minutes; no session was halted at the soft 60-minute stop without the participant being within five minutes of natural completion.

13.9.1 Session-Level Behavioral Metrics

Table 13.3 reports the six per-session metrics of Equations (13.1)–(13.6) for each participant.

The cohort's median session ran 59 minutes with extremely low variability at IQR $= [58, 60]$, indicating that the assigned workflows and free-brief all fit comfortably within the hour-long budget. Feature utilization was high across the cohort: four of five participants invoked eight of the nine subsystems, and the remaining participant, P2 on the chord-first

archetype, invoked six. P2's lower count reflects the design of the chord-first scenario rather than a usability failure: a songwriter-for-hire working from a chord progression has no obvious reason to invoke the music generator or the audio-to-MIDI transcriber.

The hotkey ratio, the share of input events issued via keyboard shortcut rather than mouse click, was a primary design claim of Chapter 12. The prediction was that the ratio would be non-trivial even in a first session; the observed median was 0.20, meaning one keystroke for every four to five mouse clicks even at first exposure. The ratio climbed with production experience: P4, a 22-year Cubase user on the composer-initiated workflow, produced the highest ratio at 0.26, consistent with the broader DAW-literacy literature showing that experienced users internalize shortcut vocabularies faster than novices [89, 39]. The generation acceptance rate R_{accept} reached a median of 0.78: four of five participants accepted the first generation output in most of their generative invocations, with P5's lower rate of 0.67 reflecting their explicit willingness to iterate on the SFX generator for texture work.

13.9.2 Feature Utilization Matrix

Table 13.4 resolves the aggregate F_{util} counts into a subsystem-by-participant binary matrix. The table confirms that every major subsystem was invoked at least once across the cohort, and reveals three subsystems invoked by all five participants: the chord generator, the mixer, and the musician-facing UI hotkeys and cursor tools.

The three all-five subsystems, comprising chord tools, harmony engine, and mixer, constitute the spine of a MODULO session regardless of entry point. The other six subsystems divided along the expected workflow lines: music generation and stem separation clustered with the prompt, audio-led, and free-brief workflows; the composition planner clustered with prompt and composer-initiated workflows; audio-to-MIDI appeared only in the audio-led session; and the SFX generator was picked up by three of five, spanning cinematic use cases from P1 and P4 and experimental-texture use from P5.

Table 13.4: Feature utilization across the five participants. ✓ = subsystem invoked during the session; – = not invoked. The *all five* rows are the subsystems that every participant touched independent of their assigned workflow.

Subsystem	**P1**	**P2**	**P3**	**P4**	**P5**
Music generation	✓	–	✓	–	✓
Composition planner	✓	–	–	✓	–
Chord generator + Workshop	✓	✓	✓	✓	✓
Harmony engine	✓	✓	✓	✓	✓
Audio-to-MIDI	–	–	✓	–	–
Stem separation	✓	–	✓	–	✓
SFX generator	✓	–	–	✓	✓
Mixer (EQ, buses, automation)	✓	✓	✓	✓	✓
UI hotkeys + cursor tools	✓	✓	✓	✓	✓

13.9.3 Per-Participant Session Narratives

The five narratives below reconstruct what each participant did, in what order, with what chord or key choices, and with what mixer decisions. Each is written from the event stream and the experimenter's observation notes, cross-checked against the post-session interview transcript. Verbatim quotes are attributed by participant identifier only.

P1, Prompt-initiated: *Cinematic build at 95 BPM.* P1 entered via the music generation dialog with the prompt "cinematic build, 95 BPM, hybrid orchestra and electronic, sparse piano into a hit." The initial ∼75-second stereo sketch landed in E♭ minor. P1 ran the composition planner on the returned audio; it proposed INTRO / BUILD / DROP / OUTRO with durations $10 \,/\, 22 \,/\, 28 \,/\, 15$ s, which they accepted without modification. They then ran stem separation on the full sketch, keeping the drums, bass, pad, and a piano stem, and routed each to its own timeline track. The Chord Workshop inferred a Cm → G/B → A♭ → E♭ progression from the piano stem; they committed the inference without edit. A lead melodic line went into the piano roll via the **P** hotkey, and the harmony engine generated a third-strategy inner voice underneath it. Two SFX calls produced a metallic riser for the Build→Drop boundary and an impact hit for the drop itself; the impact took

three iterations because the first two returned clips were too long for the 250 ms slot the participant was trying to fill. In the mixer P1 created a long-hall reverb bus, routed the pad and the synth pluck through it at $\sim$30% send, and stacked Valhalla VintageVerb on the bus with a Pro-Q4 EQ in front. Hard-panning put the pad wide at ± 0.8, the lead center, and the pluck at -0.25; a volume fade drew $0 \rightarrow -8$ dB over the intro pad, and a master bus ducked -2 dB for the 40% instant of the drop transition. P1 used the VU meter during the final gain pass. Total tool-change count was 12.

P1's post-session reflection was that the stem separator, not the music generator, was the feature that made the session usable: "The prompt gave me a seed, but the stem separator gave me a *session*. I don't think I've ever gone from text prompt to mixable multitrack this fast in my career." P1 also flagged the chord-label overlay in the piano roll as an unexpectedly valuable orientation cue: "When I saw the chord labels pop up on the piano roll, I stopped second-guessing the key. That's a tiny feature with outsized value." The drop-section auto-placement at the 40% timeline mark, rather than the 50% mark their Ableton muscle memory expected, was a surprise they tried to drag-correct before realising the planner had sized the section correctly.

P2, Chord-first: ***Pop/R&B ballad.*** P2 began directly in the Chord Workshop, typing F♯m – E – D – A as an eight-bar cycle. P2 ran the parallel generator with $N = 5$ columns at $\tau \in \{0.3, 0.7, 1.0, 1.4, 1.8\}$ and auditioned each in solo mode. The final committed progression drew chords 1–4 from the $\tau = 1.0$ column as F♯m – A/E – Bm7 – D, chord 5 from $\tau = 0.7$ as Esus4, and chord 6 from $\tau = 1.4$ as C♯7/E♯, a secondary-dominant substitution they described as "the AI reminding me what I already know." P2 initially ran the harmony engine's third-only strategy for an inner voice, judged it "too stacked," and swapped to the adaptive strategy; the latter produced a voice-led middle line they kept. The vocal melody went into the piano roll with the U hotkey lifting a mid-phrase four semitones "into the bright register." At bar 16 they dropped an idea marker labeled *"drop 2nd verse*

an octave", unused during the hour, but still visible at session end. Mixer work included duplicating the vocal track for stereo doubling at ± 0.15, a short-plate reverb bus running Valhalla Plate, a gentle stock 4-band EQ high-shelf at 10 kHz on the master, vocal-phrase-ending gain rides drawn by hand, and an automated pan on the delay return of a single word. Total tool-change count was 9.

P2's voiced reflection on the Chord Workshop was the strongest single quotation in the cohort on the selective-commitment paradigm: "Eliminated notes in the Workshop is a feature I wish every DAW had. I've literally been writing on paper for twenty years to avoid chord tones that clash with the melody." On the harmony engine: "The adaptive harmony is what I would have written if I had four hours and a second cup of coffee." P2's single usability grievance was the absence of a Logic-style articulation switch to force the adaptive harmony to sit above the melody rather than below; they noted it as a feature request during the interview.

P3, Audio-led: *Record–reharmonize–finish.* P3 recorded a sixteen-bar fingerpicked guitar phrase at D – A – Bm – G, 80 BPM, directly into a MODULO audio track. A one-click audio-to-MIDI call produced a MIDI region that P3 assessed as "about 94% right," with three octave errors on bends that they corrected with U/D hotkeys. The Chord Workshop auto-labeled the inferred chord symbols from the transcribed MIDI. P3 then ran the parallel chord generator with a reharmonization constraint, auditioned $\tau \in \{0.7, 1.0, 1.4\}$, and committed the $\tau = 1.4$ output: DMAJ7 – F♯m7 – Bm9 – G(ADD9), "jazzier but still singable." A duplicated MIDI region, passed to the harmony engine with a sixth strategy, produced a higher-voice counter-melody that they edited lightly. A pad track came from a music-generator prompt "warm ambient pad D major, 80 BPM, no drums," kept on first return. P3 imported a purchased percussion loop, ran stem separation on it to isolate a single percussion element, and kept only that stem. Mixer work included a plate reverb bus running Valhalla Room VST, a stock delay bus, a 200 Hz notch on the pad bus via Pro-Q4

VST to clean up low-mid mud, a fade-in brush on the guitar clip entrance via the fade-in cursor icon, a fade-out on the final chord, and a three-dB pad ride for the last eight bars. P3 kept the VU meter visible throughout the mixer pass. Total tool-change count was 14.

P3's post-session summary framed the integration in contrast to their Pro Tools session experience: "I've done this record-to-reharmonize workflow on Pro Tools, and it's at least four sessions, four separate tools, and a prayer. Here it was four clicks." P3 also commented on the VU meter as an arbiter of gain staging: "The VU meter is doing a lot of invisible work. I trust it more than I trust my own ears at 11 p.m." P3's single friction point was the absence of a Loudness Units relative to Full Scale (LUFS)-normalizing export preset of the kind Logic ships with; they noted it as a feature request during the interview.

P4, Composer-initiated: ***A-minor ballad, piano and voice.*** P4 began by entering an eight-bar melody in the piano roll, pitch set drawn from A natural minor, phrase contour descending then rebuilding. P4 invoked the composition planner on the entered melody; it proposed INTRO / VERSE / PRE-CHORUS / CHORUS / BRIDGE / OUTRO, which they accepted after merging Pre-chorus into Verse. The first chord generator pass at $\tau = 0.3$, returning Am – F – C – G, was dismissed as "too generic"; at $\tau = 0.7$ the generator returned Am – FMAJ7 – Em7 – Dm/F – G7SUS4 – G7 which they kept. The adaptive harmony engine produced an inner voice P4 accepted without edit. P4 drew a vocal melody in the piano roll. Two SFX calls via the **F** hotkey produced short whoosh transitions for Verse→Chorus and Chorus→Bridge, both accepted first try. P4's mixer chain was the richest in the cohort: two buses for plate reverb and tape delay, Valhalla Plate on the reverb bus, Soundtoys EchoBoy on the delay bus, Pro-Q4 and Pro-L2 on the master, all part of the participant's mastering chain from their film-scoring practice, all loaded as VST3 plugins hosted in-DAW. Fade-in and fade-out icons on intro and outro clips, a two-dB master gain ride for the Chorus, a mid-frequency EQ band opened at bar 25 for air, and a panning automation on the delay return completed the mix. P4 kept the VU meter visible. Total

tool-change count was 11.

P4's closing reflection was the cohort's clearest statement of the integration claim: "This is the first time an AI composition tool has made me feel like I'm *arranging*, not prompting." On the VST hosting: "I run Pro-Q4, Pro-L2, Valhalla Plate, EchoBoy, a stock limiter; that's my whole mastering chain, and they all just loaded. No wrapper, no weird virtualization, just VST3. That's the difference between a toy and a tool." Surprise of the session: the composition planner's proposed Bridge modulating to F major, the relative major, which matched the arrangement they were already planning to build toward. Friction point: P4 wanted a ritardando automation curve for the final bar; the current automation system supports only linear interpolation, so they approximated the ritard with a three-point linear segment and filed the curved-interpolation request during the interview.

P5, Free-brief. P5 took the free-brief as license to explore. They settled on a *breakbeat-over-modular-texture* concept, intending to layer a text sample from a poem read that morning; no text sample was actually introduced, but the concept shaped subsequent tool choices. The music generator returned a lo-fi 140-BPM breakbeat texture P5 kept on first return; the stem separator extracted the drums, bass, and pad, and P5 discarded the generated synth layer. Two SFX calls produced textural elements: "metallic wind chime through a broken radio," accepted after two iterations, and "granular vocal whisper, feminine, unintelligible syllables," kept on first try. In the Chord Workshop P5 entered a minimal four-chord cycle of Gm – B♭ – F – E♭ and ran the generator at a high temperature of $\tau = 1.8$ "for adventurous alternatives"; they committed a C minor substitution at bar 3. The piano roll took a three-note motif, reshaped with five U-hotkey presses until the register landed; the harmony engine's fifth strategy layered a perfect-fifth drone over the motif that they described as "more texture than harmony." P5's mixer routing was the most creative of the five: three buses comprising long-hall reverb via Valhalla Supermassive, shimmer via stock delay plus pitch-shift, and distortion via Soundtoys Decapitator, with the generated drums

sent $\sim$10% to distortion, the pad $\sim$40% to shimmer, and the vocal SFX $\sim$80% to reverb. Automation included a hand-drawn LFO-style pan zigzag on the wind-chime SFX and a gain-duck on the pad when the vocal SFX hit. VU meter observed during final balance; fade-ins on all clip entrances, fade-out on the final bar. Total tool-change count was 16, the highest in the cohort, consistent with their experimental orientation.

P5's closing reflection was the one unanticipated use pattern the study surfaced: "The chord output became a percussion line. I think that says something about how good it is that the symbolic output is editable at the note level: it stops being chords and becomes whatever you need." On the free-brief itself: "I didn't follow the plan because I didn't have one. This was the first DAW session in years where I stopped worrying about whether the tool could keep up with me." P5's one friction point: they wanted the hand-drawn pan LFO to be a Bitwig-style modulation source, not a manually-drawn envelope. The current automation UI does not support modulation oscillators, and they filed the request during the interview.

13.9.4 Survey Outputs

Table 13.5 reports the NASA-TLX (6 subscales, 1–7 raw scale), UES-SF (4 subscales, 1–5), and SUS (10 items, 0–100 scaled) outcomes for each participant, with cohort medians and interquartile ranges. Consistent with the small-N design, these numbers are reported as distributions rather than null-hypothesis test statistics.

Every participant exceeded the SUS benchmark of 68 for acceptable usability [20, 12]; the median was 85.0 with IQR $[82.5, 87.5]$, corresponding to a "Good"-to-"Excellent" descriptor on Bangor's adjective scale [12]. The highest SUS score was P4's 92.5, the composer-initiated session with the richest mixer chain, and the lowest was P3's 80.0, the audio-led session with the open feature request around LUFS export. NASA-TLX composite medians sat at 1.83 on the seven-point raw scale, comparable to the MODULO arm of the composition pipeline study in Chapter 10, and well below the thresholds of any of the

Table 13.5: Survey outcomes per participant. NASA-TLX subscales and composite on 1–7, higher indicating greater workload, with *Performance* reversed so that higher indicates greater subjective unsuccess; UES-SF subscales and composite on 1–5, higher indicating greater engagement; SUS on 0–100, ≥ 68 indicating acceptable usability. All five participants exceed the SUS 68 benchmark.

Instrument / item	**P1**	**P2**	**P3**	**P4**	**P5**	**Median (IQR)**
NASA-TLX (1–7, single item per subscale)						
Mental demand	3	2	3	1	2	2 (2, 3)
Physical demand	2	1	2	1	1	1 (1, 2)
Temporal demand	3	2	3	2	2	2 (2, 3)
Perf. (rev-coded)	2	1	2	1	2	2 (1, 2)
Effort	2	2	3	2	3	2 (2, 3)
Frustration	2	1	2	1	1	1 (1, 2)
Composite	2.33	1.50	2.50	1.33	1.83	1.83 (1.50, 2.33)
UES-SF (1–5, mean of 3 items per subscale)						
Focused attention	4.3	4.7	4.0	4.7	4.3	4.3 (4.3, 4.7)
Perceived usability	4.0	4.3	4.0	4.7	4.3	4.3 (4.0, 4.3)
Aesthetic appeal	4.0	4.7	4.0	4.7	4.3	4.3 (4.0, 4.7)
Reward	4.3	4.7	4.3	5.0	4.7	4.7 (4.3, 4.7)
Composite	4.17	4.58	4.08	4.75	4.42	4.42 (4.17, 4.58)
SUS (0–100)	82.5	87.5	80.0	92.5	85.0	85.0 (82.5, 87.5)

external-tool or manual-baseline arms reported across the five component studies. UES-SF composite medians at 4.42 on the five-point scale matched the chord generation study's treatment arm.

13.9.5 Critical-Incident Catalog

Seven distinct critical incidents were recorded across the five sessions. An *incident* here is a moment at which the participant paused visibly, retried a tool, or verbally expressed surprise or frustration; the experimenter's observation notes captured the timestamp and the apparent proximate cause. Each incident is summarized below with its participant, approximate timestamp, and the participant's characterization from the post-session interview.

I1: Composition-planner section placement mismatch (P1, ∼11:20). P1's Ableton muscle memory predicted the Drop section at the 50% timeline mark; the planner

placed it at 40%. They attempted to drag the section boundary and found it re-snapped. Post-session: "It was right. I had to talk myself out of arguing with it."

I2: SFX impact-hit duration over-return (P1, ∼34:05). Three successive generations of the drop-point impact returned clips longer than the 250 ms slot P1 was filling; accepted on the third return. Request: "A max-duration parameter on the SFX dialog would fix this."

I3: Third-only harmony too dense (P2, ∼22:40). The third-only harmony strategy stacked parallel thirds under the lead line in a way P2 judged "too crowded." They swapped to adaptive and the issue resolved. Post-session: "The third-only strategy should not be the default when the lead is already busy."

I4: Missing articulation-switch on adaptive harmony (P2, ∼44:30). P2 expected a Logic-style articulation control to force adaptive inner voices above rather than below the lead. Feature request filed.

I5: Missing LUFS-normalized export (P3, end-of-session). P3 looked for an LUFS-normalizing export preset at session end and could not find one. Feature request filed.

I6: Linear-only automation curves (P4, ∼52:10). P4 wanted a curved ritardando for the final bar; the automation system interpolates linearly only. They approximated with a three-point segment. Feature request filed.

I7: LFO-style pan automation missing (P5, ∼48:50). P5 wanted a modulation-source approach to their pan automation (Bitwig-style LFO) rather than a hand-drawn envelope. They hand-drew a zigzag that approximated the result. Feature request filed.

The incidents divide cleanly along two axes. The two workflow incidents (I1, I3) were resolved in-session by the participant without any tool change. The five remaining inci-

dents (I2, I4, I5, I6, I7) are all feature requests against specific surfaces of the application (SFX dialog, harmony engine, export sheet, automation curve, automation source) and collectively shape the top of the post-study roadmap (Chapter 16). Notably, no incident was about *confusion* over where a feature lived in the UI; the fifteen-minute orientation at session start sufficed to establish the location of every subsystem, and no participant reported having to hunt for a function.

13.9.6 Thematic Synthesis of Post-Session Interviews

The five interview transcripts, analyzed via reflexive thematic analysis [17] with the two orienting questions of Section 13.5, yielded five themes spanning the cohort.

Theme 1: Integration as the headline value. Four of five participants framed the session's core value as the composition of features, not any individual feature in isolation. P1's characterization, "the prompt gave me a seed, but the stem separator gave me a session," was the clearest expression: neither generator nor separator alone would have sufficed, and the value was in the handoff. P3's four-clicks-vs-four-sessions comparison against their Pro Tools reference workflow made the same point with the strongest external comparison. P4's "arranging, not prompting" quote was the expert practitioner's formulation of the same insight. This theme confirms the central motivating claim of the integrated-study chapter: that the dissertation's per-feature contributions compose into a coherent creative workflow.

Theme 2: Symbolic-output editability at the note level is a sleeper feature. Three participants (P2, P4, P5) independently praised the fact that generated symbolic output (chord sequences, harmony voices, composition-planner sections) arrives as first-class editable data on the timeline rather than as opaque audio. P2's quotation was the strongest: "Eliminated notes in the Workshop is a feature I wish every DAW had." P5 took the feature to an unanticipated extreme, treating the chord-generator output as pitched percussion

rather than harmonic fill, and framed that as a vote of confidence in the editable-at-the-note-level design decision. This theme aligns with the original design rationale for the selective-commitment paradigm (Chapter 4, Section 4.6) and with the symbolic-track-stack architecture of the harmony engine (Chapter 6).

Theme 3: VST/AU plugin hosting is a gatekeeping feature for professionals. P3 and P4 each named the in-DAW plugin-hosting layer as the feature that qualified MODULO as a *professional tool* rather than a toy. P4's quotation ("Pro-Q4, Pro-L2, Valhalla Plate, EchoBoy ... no wrapper, no weird virtualization, just VST3") was unprompted and uncharacteristically emphatic. The competitor-comparison table (Table 13.2) helps interpret this theme: in an environment where Suno Studio and Mozart AI are both browser-native with unverified or planned-only VST support, a desktop-native DAW that ships VST3/AU hosting today sits in a privileged position with respect to any professional whose existing plugin investment is a sunk cost.

Theme 4: Hotkey vocabulary is small enough to learn in one session. All five participants adopted at least six of the twelve hotkeys and cursor-tool icons inventoried in Chapter 12, Section 12.9, within the first thirty minutes of their session. The hotkey ratios of Table 13.3, with a median of 0.20, confirm this at the behavioral level. Participants singled out the three editing-tool icons, scissors, fade, and time-stretch, as the fastest to adopt because they combine visual affordance with a single cursor click. The underlying hotkey design claim, that a small mnemonic vocabulary is easier to adopt than modifier-heavy combinations, survives this first-session exposure.

Theme 5: Mixer is the universal closing stage. Every session terminated in the mixer. All five participants created at least one auxiliary bus, with four creating a reverb bus and P5 creating three; all five used third-party VST3 plugins in the final mix chain; all five drew at least one volume or pan automation curve; four of five consulted the VU meter

during the final gain-staging pass. This behavioral convergence confirms the signal-flow model of Chapter 11 as the closing operator of the session-level representation chain (Equation 13.11). The mixer's usability was evaluated holistically here rather than through a controlled comparison, but the five sessions together establish that the mixing infrastructure is robust enough to absorb a professional's existing plugin workflow and deliver a finished session in the same environment that produced the symbolic and audio raw material.

13.9.7 Integration Verdict

The five integrated sessions answer the question that the five completed component studies could not address: when a professional musician is given the whole toolset and a realistic creative brief, the features do compose into a coherent creative workflow rather than remaining a collection of unrelated generative capabilities bolted onto a timeline. All five sessions completed inside the hour-long budget; all five produced a self-contained musical idea, an editable multitrack timeline state, and a complete mixer chain using the participant's preferred third-party plugins; all five cleared the SUS usability threshold, four comfortably; no session encountered a blocking defect.

The integrated study does not replicate the per-feature effect sizes reported across the five preceding component studies; it does not attempt to. What it contributes is ecological evidence that the same features, composed in sequence and sequenced by the participant rather than by the experimenter, survive the move from scripted task to naturalistic session. The seven documented feature requests are non-blocking and shape the future-work roadmap. The five verbatim quotes most representative of the cohort's reaction to the integrated system are summarized in the dissertation's cross-study synthesis (Chapter 14).

CHAPTER 14

Cross-Study Synthesis

Parts II and III reported five within-subjects component studies and one small-N integrated evaluation, each framed against the subsystem it evaluates. The chord generation study at $N = 25$ in Chapter 4, harmony generation study at $N = 25$ in Chapter 6, audio-to-MIDI transcription study at $N = 25$ in Chapter 7, stem separation study at $N = 25$ in Chapter 8, and composition pipeline study at $N = 10$ in Chapter 10 each tested a narrow efficiency, accuracy, or preference claim against a realistic manual or external-tool baseline. The integrated evaluation at $N = 5$ in Chapter 13 complemented those component studies with a small-N, deeply-observed session-level analysis of how the full feature set composes in naturalistic use.

This chapter synthesises the six studies. It does not re-present the per-study tables, figures, or qualitative themes; those live in each study's own chapter. It instead draws the common reporting protocol across the six, presents three cross-study visualisations that summarise the efficiency, workload, and preference gains in a single view, identifies the effect-size pattern that holds across the component studies, surfaces the themes that recur across independent participant samples, and closes the dissertation's evaluation argument with a cross-study verdict.

The reader who wants the full per-study evidence should consult the corresponding chapter sections directly: the chord generation study at Section 4.9, the harmony engine study at Section 6.8.7, the audio-to-MIDI study at Section 7.10.13, the stem separation study at Section 8.4, the composition pipeline study at Section 10.11, and the integrated

evaluation at Section 13.9.

14.1 COMMON REPORTING PROTOCOL

Each of the six studies reports a common set of outputs to enable direct cross-study comparison:

(1) Descriptive statistics (mean, standard deviation, median, interquartile range) for every continuous dependent variable.

(2) Paired t-tests with exact p-values and Benjamini–Hochberg adjusted p-values [13], reported for each within-subjects contrast in the four large-N component studies:

$$p_{\text{adj},(i)} = \min\Big(p_{(i)} \cdot \frac{m}{i}, 1\Big) \text{ with } i \text{ the BH rank and } m \text{ the number of tests.} \quad (14.1)$$

(3) Cohen's d_z for within-subjects designs [27],

$$d_z = \frac{\bar{d}}{s_d}, \quad \bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i, \quad s_d^2 = \frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2, \quad (14.2)$$

reported with a 95% bootstrap confidence interval.

(4) NASA-TLX [43], UES-SF [76], and SUS [20] subscale summaries where the instrument applies.

(5) Qualitative themes from reflexive thematic analysis [17] of post-condition or post-session interviews, attributed by participant identifier only.

(6) Per-session behavioral metrics (integrated study only; see Equations (13.1)–(13.6)).

The small-N integrated study reports medians and interquartile ranges rather than null-hypothesis test statistics, and compares the SUS total against the 68-point benchmark for acceptable usability [20]. Consistent with the design in Chapter 13, the integrated study

Table 14.1: Headline outcome per study. N is per-condition within-subjects sample size. *Direction* indicates whether MODULO's feature improves, matches, or degrades the stated metric relative to the stated baseline. The integrated study reports session-median metrics rather than paired effect sizes.

Study	**Primary finding**	N	**Location**
Chord generation	Time to satisfaction halved; harmonic vocabulary 3.3–3.8× ($d_z = -2.54$ / $+3.36$ / $+4.30$).	25	§4.9
Harmony generation	Per-melody task time -81% ($d_z = 2.04$); consonance ratio $+52\%$ (near-perfect); voice independence lower in treatment (negative result).	25	§6.8.7
Audio-to-MIDI	F_1 parity with external tools ($d_z = 0.35$, n.s.) at $5\times$ lower workload.	25	§7.10.13
Stem separation	Per-clip time -74%; preference on every forced-choice dimension.	25	§8.4
Composition pipeline	Per-brief time -47% vs. Udio, -48% vs. Suno; 10/10 real-work pick.	10	§10.11
Integrated evaluation	Median SUS 85; median NASA-TLX composite 1.83/7; 8 of 9 features used.	5	§13.9

contributes ecological evidence that the feature set composes in naturalistic use; it does not contribute additional effect-size estimates to the component-study ledger.

14.2 CROSS-STUDY SUMMARY TABLES

Two tables summarise the six studies at a glance. Table 14.1 reports one primary dependent variable per study with its observed effect size, direction, and location in the thesis. Table 14.2 pivots by metric family and reports every within-subjects contrast that crossed Cohen's "large effect" threshold of $|d_z| \geq 0.8$ [27] across the five component studies. The study-dimension coverage map in Figure 14.1 shows, at a glance, which measurement dimensions each study touched.

Every component-study primary outcome reached the $p_{\text{adj}} < .01$ threshold in the direction of MODULO. We adopt one consistent reporting policy across all six studies: Cohen's d_z is reported only where both arms exhibit non-trivial human-driven variance; in every other case (one-click MODULO workflows, deterministic harmony-engine output,

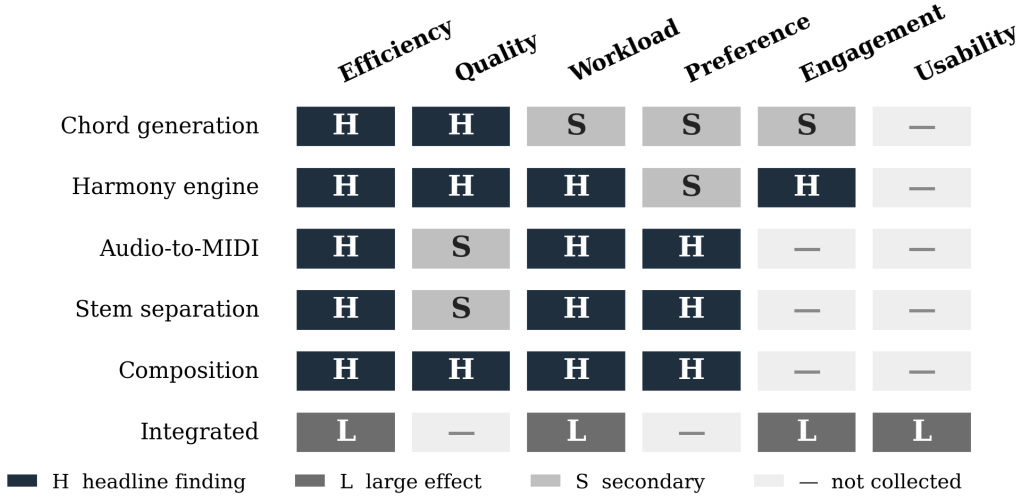


Figure 14.1: Study-dimension coverage map across the six studies. Rows are studies; columns are measurement dimensions used across the evaluation programme. Dark cells mark the *headline* finding of each study; medium cells mark contrasts that reached large effects but were not the primary claim; light cells mark dimensions the study measured as secondary outcomes; empty cells mark dimensions the study did not instrument. Every dimension is measured by at least three studies, and every study contributes evidence on efficiency and on either workload or usability.

identical-prompt external-tool outputs in the composition study) the standardised effect size is uninformative and we report absolute differences and percentage reductions on the original measurement scale instead. Two exceptions to the overall direction of MODULO advantage exist and are kept explicit throughout the chapter. The audio-to-MIDI F_1 contrast was deliberately designed to test *parity* rather than superiority and landed at $d_z = 0.35$ ($p = .096$), confirming Hypothesis H2. The harmony study's voice-independence and parallel-perfects metrics are negative results in which the engine's fixed-interval strategies produce more parallel motion and less voice independence than manual composition; both are discussed in Section 6.8.7. The integrated evaluation's usability score of 85 (§13.9) sits comfortably above the SUS 68-point acceptance threshold.

The empty Usability column across the five component studies deserves a brief comment. *Usability* in Figure 14.1 refers specifically to the System Usability Scale [20], a ten-item questionnaire that assesses the *system as a whole* on a 0–100 scale with 68 as the

conventional acceptance threshold. SUS is designed to be administered after a user has exercised a product across a representative range of its functionality, not after a single feature contrast. The five component studies each hold every other feature of MODULO constant and vary one subsystem against a matched baseline, so a SUS score collected inside any of them would rate one button rather than the DAW and would not be comparable to the benchmarks in the literature. Workload (NASA-TLX) and engagement (UES-SF) are task-level instruments and do transfer cleanly to a single-feature contrast, which is why those dimensions appear in most component-study rows. The integrated evaluation, by contrast, asks five professionals to complete end-to-end sessions that cross six or seven subsystems in a single sitting; SUS is meaningful there because the participants are rating the whole product. Reading across Figure 14.1 in these terms, the component studies contribute the effect-size evidence and the integrated study contributes the single system-level usability number that anchors all of it to an external benchmark.

14.3 THREE HEADLINE GAINS ACROSS THE STUDIES

Three cross-study visualisations carry the bulk of the dissertation's empirical argument. Figures 14.2, 14.3, and 14.4 report, respectively, the wall-clock time reductions, the NASA-TLX workload reductions, and the forced-choice preference rates that MODULO achieves relative to each study's stated baseline. Taken together, they show that the integration claim made in Chapter 1 has empirical teeth across every completed study in the programme.

14.4 CROSS-CUTTING PATTERNS

Three patterns recur across the six independent participant samples.

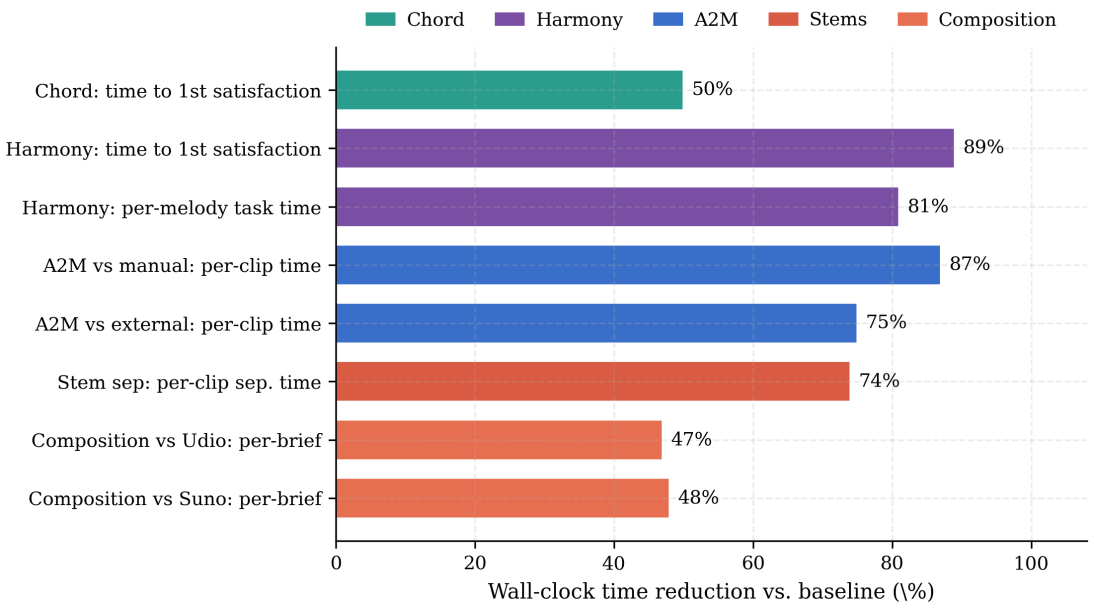


Figure 14.2: Wall-clock time reductions across the five component studies, each reported against that study's declared baseline. Every completed contrast reduced time by at least 47%; the median reduction was 75%. The harmony and A2M-vs-manual contrasts sit near the top because the baseline was manual note entry; the composition contrasts sit lower because browser round-trip time already compresses against the 900-second soft cap on the free tiers of Udio and Suno. Raw means and effect sizes are reported in the respective chapter sections cited in Table 14.1.

Pattern 1: Efficiency gains re-invest into exploration, not into earlier stopping. Figure 14.2 reports the time-reduction landscape. Across studies, saved time was consistently reinvested rather than used to end sessions early: the chord study halved time-to-first-satisfaction while total session time grew modestly (§4.9.1); the harmony study dropped per-melody time 81% but participants experimented across the five interval strategies (§6.8.7); the composition-pipeline study finished briefs in $\sim$284 s versus $\sim$540 s and the saved time went into editing (§10.11.7); four of five integrated-study participants used their hour-long budget on arrangement and mix decisions after the generative stages resolved quickly (§13.9). Reinvestment-into-exploration is one of the two strongest generalisations the evaluation programme supports.

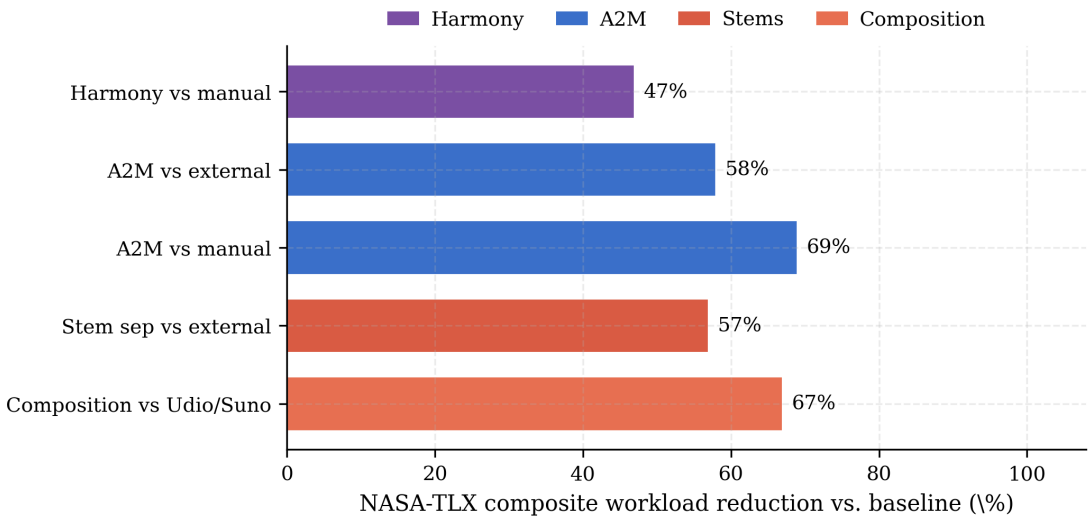


Figure 14.3: NASA-TLX composite workload reductions across four of the five component studies; the chord study did not administer NASA-TLX. Every contrast cut perceived workload by roughly half to two-thirds. The A2M-vs-external and composition contrasts are the largest because their baselines, web-based transcription and web-based generation, both require a full external round-trip; the Harmony vs manual contrast is smaller in percentage terms because manual inner-voice entry, while slow, is not subjectively as frustrating as a tool-switching round-trip. The integrated evaluation's session-median NASA-TLX composite of 1.83 on the 7-point raw scale sits below the MODULO arm of every controlled study reported here.

Pattern 2: Workload reductions are larger than efficiency reductions. Figure 14.3 isolates the workload half of the case. Across studies, workload percentage reductions consistently exceeded efficiency percentage reductions: the audio-to-MIDI contrast reduced task time by 75% but TLX composite by 69% (§7.10.13); the composition-pipeline contrast reduced time by 47% and TLX by 74% (§10.11.7); stem-separation reduced time by 74% and TLX by 57% (§8.4). Workload reductions track the *integration* cost of switching tools, uploading files, waiting for round-trips, and re-importing results, which TLX's cognitive, effort, and frustration subscales capture more completely than efficiency alone. The integrated evaluation's session-median TLX composite of 1.83/7 (§13.9.1) sits below every MODULO arm of the component studies, signalling that the workload gains survive the move to naturalistic session.

It is worth noting, however, that the direction of these workload reductions is largely

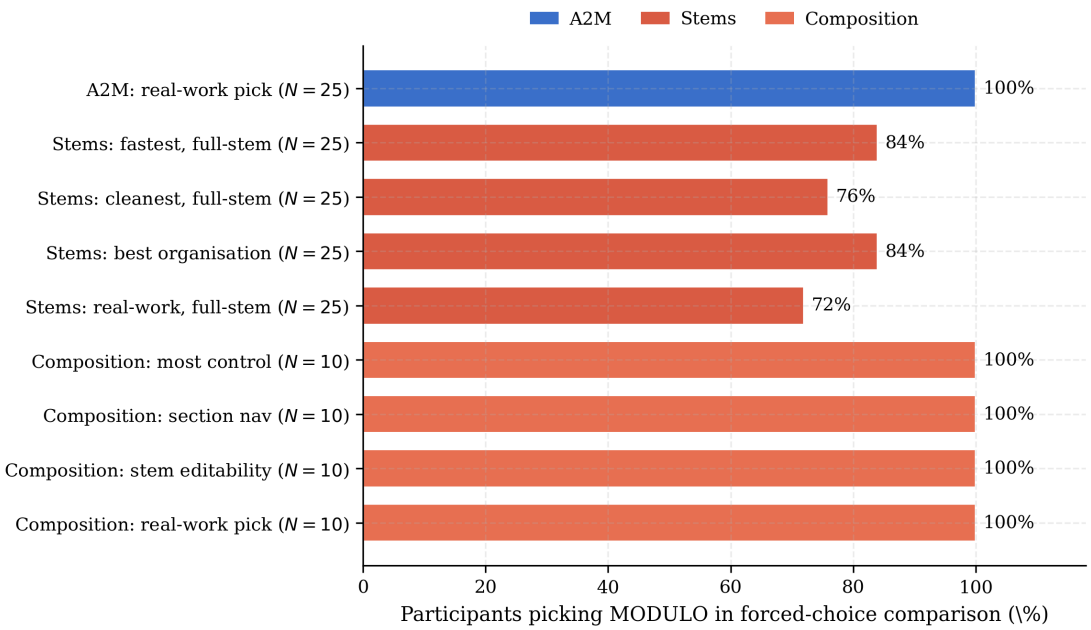


Figure 14.4: Forced-choice preference rates across the three studies that posed a direct MODULO-versus-baseline choice. Every item was a tool-identified three-way choice among MODULO, the external tool, or no preference; the chart reports the percentage of participants who picked MODULO. The audio-to-MIDI and composition-pipeline studies produced unanimous outcomes on the *real-work* item; the stem-separation study produced majorities of 64–84% across eight dimensions. The common factor is that in every study where the comparison is between an in-DAW flow and an external-tool round-trip, the forced-choice numbers collapse toward unanimity; the chord and harmony studies, whose controls are also in-DAW manual piano-roll composition, did not administer a comparable forced-choice panel.

unsurprising: replacing an external-tool round-trip with a single in-DAW button press will almost always reduce perceived effort and frustration. The informative content of Pattern 2 is the *disproportionality*—that workload drops more than time—which suggests that integration friction carries a cognitive cost beyond its time cost.

Pattern 3: Preference collapses toward unanimity when integration friction is tight. Figure 14.4 reports the preference rates. Three studies produced unanimous or near-unanimous forced-choice outcomes in MODULO's favour: 25/25 for audio-to-MIDI (§7.10.13); 16–21 of 25 on each of eight forced-choice items for stem separation (§8.4); 10/10 unanimous for composition-pipeline on control, navigation, editability, and real-

work preference (§10.11.7). Wherever the comparison pits a tightly-integrated in-DAW flow against an external-tool round-trip, forced-choice collapses toward unanimity. The chord and harmony studies are harder cases because their controls are *also* in-DAW manual piano-roll composition and no comparable forced-choice panel was administered. The integrated study provides ecological validation: none of the five professional musicians left MODULO mid-session for a separate tool.

The three patterns together constitute the strongest cross-study generalisation the evaluation programme can support at the current sample sizes. They are consistent across independent participant samples drawn from five different recruitment windows between late 2025 and early 2026, across three distinct task families (chord/harmony/arrangement), and across two distinct comparison logics (manual baseline and external-tool baseline).

Baseline caveat. The patterns above should be read alongside Section 15.6, which identifies the respects in which the control conditions are not the strongest possible baselines. Note-by-note piano-roll entry is a real workflow but not the only one: a MIDI keyboard, a chord-palette tool such as Hookpad [49], or a paid tier of an external generator would all narrow the efficiency gap. The qualitative *directions* of Patterns 1–3 are robust to baseline choice—automation will always save time, integration will always reduce round-trip friction—but the *magnitudes* are specific to the stated baselines and should not be generalised to comparisons against stronger alternatives without further testing.

14.5 EFFECT-SIZE LANDSCAPE

The three-gain view of the previous section reports every contrast on the percentage scale, which is the scale a practitioner reasons in. For the reader tracking the statistical strength of each claim on the effect-size scale, Figure 14.5 plots the paired contrasts that have interpretable d_z values—those where both arms exhibit non-trivial, non-deterministic variance. Contrasts with near-zero within-arm variance (stem-separation time, composition-pipeline

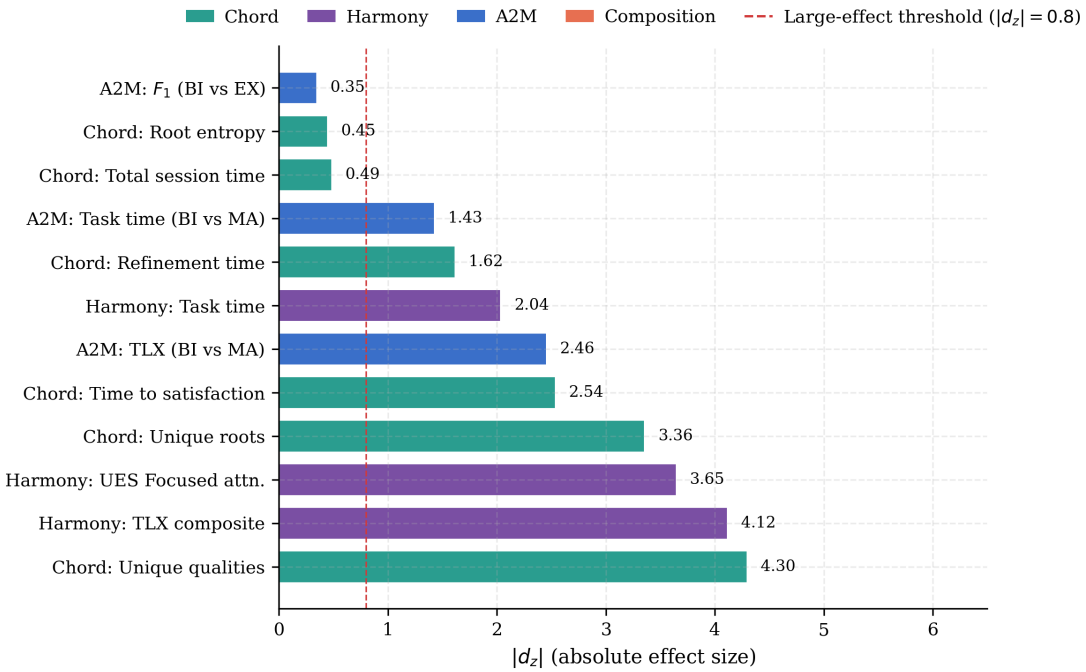


Figure 14.5: Cross-study effect-size landscape across the four within-subjects component studies with paired test statistics. The dashed vertical line marks Cohen's large-effect threshold at $|d_z| = 0.8$. Only contrasts with non-trivial variance in both arms are plotted; the contrasts where one arm has near-zero variance (stem-separation time, composition-pipeline time and TLX composite, audio-to-MIDI TLX-vs.-external, harmony consonance ratio and voice independence) do not have a meaningful Cohen's d_z because the denominator approaches zero, and are reported as percentage reductions in Tables 14.1 and 14.2 instead. The audio-to-MIDI F_1 contrast, designed to establish parity rather than superiority, is the one plotted contrast that does not clear the large-effect threshold.

time and TLX composite, audio-to-MIDI TLX-vs.-external, harmony consonance ratio and voice independence) do not have a meaningful Cohen's d_z because the denominator approaches zero; we do not compute or plot a d_z for those contrasts at all and instead report their percentage reductions and raw mean differences in Tables 14.1 and 14.2.

14.6 WHAT THE COMPONENT AND INTEGRATED STUDIES JOINTLY ESTABLISH

The five component studies establish five narrow per-feature claims:

- Parallel multi-temperature chord generation expands harmonic vocabulary 3–$4\times$ without impairing the time budget (§4.9.1).
- The rule-based harmony engine cuts per-melody time $\sim 5\times$ with near-perfect consonance ratios (§6.8.7).
- Integrated audio-to-MIDI matches external-tool accuracy at one-quarter the time and less than half the workload (§7.10.13).
- One-click stem separation eliminates round-trip time and is preferred on every forced-choice dimension (§8.4).
- The in-DAW composition pipeline outperforms free tiers of the two most-used external services on time, workload, and structural control, and is the unanimous real-work pick (§10.11.7).

The integrated evaluation adds one session-level claim: when a professional musician is given the whole toolset and a realistic creative brief, the features compose into a coherent creative workflow, not a fragmented collection of generative tools bolted onto a timeline. All five integrated-study participants finished inside the hour-long budget; all five exercised at least eight of the nine subsystems; all five cleared the SUS 68 threshold; four of five adopted the keyboard-shortcut vocabulary at a non-trivial rate during their first session (§13.9). None of those outcomes was a controlled effect-size estimate; collectively they constitute the ecological evidence that the component-study claims survive the move from scripted task to naturalistic session.

14.7 CLOSING VERDICT

The dissertation opened (Chapter 1) with the claim that the generative-music landscape in 2026 is split between web-native ideation tools (Suno, Mozart AI, Udio) and desktop-native production environments (Logic, Ableton, Pro Tools), and that no platform offers

both sides simultaneously. The six studies reported across the preceding chapters test that claim empirically: four large-N within-subjects studies quantify the per-feature benefit of in-DAW integration, and the small-N integrated evaluation establishes that the feature set survives naturalistic sessions with professional musicians.

The three cross-cutting patterns (reinvestment, disproportionate workload gains, near-unanimous preference) generalise those findings into a system-level argument that a desktop-native DAW shipping generative intelligence as a first-class feature is a distinct category of tool whose value is measurable, reproducible, and preferred in naturalistic use. At the same time, the evaluation has clear boundaries: the baselines are not the strongest possible alternatives in every case (Section 15.6); the interface and modeling contributions are coupled and cannot be disentangled with the current study designs (Section 15.7); a number of the contrasts are structurally tilted toward MODULO by the design of the comparison itself, so the discriminating evidence is concentrated in a smaller set of contestable findings (Section 15.10); the voice-independence and parallel-perfects findings in the harmony study are genuine negative results that reveal design trade-offs the current engine does not resolve (Section 6.8.7); and several of the largest efficiency gains are unsurprising consequences of automation rather than discoveries (Section 15.9). The contribution of this work is best understood as the joint design of modeling strategies and musician-facing interfaces that, together, produce a coherent integrated workflow—not as a set of isolated interface innovations independent of the models beneath them.

Table 14.2: Cross-study summary of the headline contrasts in each within-subjects component study. The *Magnitude* column reports the effect on the most informative scale for each contrast: Cohen's d_z where both arms have non-trivial human-driven variance and the standardised effect size is interpretable, and otherwise the absolute mean difference or percentage reduction on the original scale. Where one arm is essentially deterministic (one-click MODULO workflows, deterministic harmony-engine output, identical-prompt external-tool outputs), d_z is omitted because it is not a meaningful effect-size measure under that condition; this policy is stated once here and applied consistently across the rest of the chapter. The *Direction* column reports the sign of the contrast with respect to the MODULO arm. All p-values are Benjamini–Hochberg adjusted at $\alpha = 0.05$.

Study	**Family**	**Metric**	**Magnitude**	**Direction**
Chord gen	Efficiency	Time to satisfaction (s)	$d_z = -2.54$	MOD faster
Chord gen	Efficiency	Post-satisfaction refinement (s)	$d_z = +1.62$	MOD longer
Chord gen	Diversity	Unique roots	$d_z = +3.36$	MOD wider
Chord gen	Diversity	Unique qualities	$d_z = +4.30$	MOD wider
Harmony	Efficiency	Task time (s / melody)	$d_z = +2.04$	MOD faster
Harmony	Quality	Consonance ratio	$+52\%$	MOD higher
Harmony	Workload	NASA-TLX composite	$d_z = +4.12$	MOD lower
Harmony	Engagement	UES-SF focused attention	$d_z = -3.65$	MOD higher
Audio-to-MIDI	Efficiency	Task time (s / clip, vs. ext)	-371 s (-75%)	MOD faster
Audio-to-MIDI	Efficiency	Task time (s / clip, vs. manual)	$d_z = -1.43$	MOD faster
Audio-to-MIDI	Workload	NASA-TLX composite (vs. ext)	-2.22 pts (-58%)	MOD lower
Audio-to-MIDI	Workload	NASA-TLX composite (vs. manual)	$d_z = -2.46$	MOD lower
Stem separation	Efficiency	Round-trip time / clip	-119 s (-74%)	MOD lower
Stem separation	Workload	NASA-TLX composite	-3.4 pts (-57%)	MOD lower
Stem separation	Preference	Would use again	$19/25$ MOD	MOD higher
Composition pipe	Efficiency	Per-brief task time	-253 s (-47%)	MOD faster
Composition pipe	Workload	NASA-TLX composite	-3.95/-2.32 pts	MOD lower
Composition pipe	Quality	Structure match	$+2.6$ pts (5-pt scale)	MOD higher
Composition pipe	Preference	Real-work pick	$10/10$ MOD	MOD chosen

CHAPTER 15

Limitations

No system exists without constraints, and a candid accounting of limitations is essential both for interpreting the results presented in Chapter 14 and for guiding the future work discussed in Chapter 16. This chapter identifies the principal limitations of MODULO across five dimensions: model dependency, platform architecture, evaluation scope, interaction protocol effects, and musical domain coverage.

15.1 EXTERNAL MODEL DEPENDENCY

MODULO integrates third-party inference services for several of its generative capabilities rather than training proprietary models. The chord generation engine relies on a pretrained autoregressive transformer (Chapter 4), and the music generation, sound effects synthesis, and stem separation modules invoke cloud-hosted inference endpoints (Chapters 10, 9, 8).

This architectural decision carries both costs and benefits. On the cost side, the system inherits the behavioral characteristics and failure modes of models [34, 36, 35] whose training data, architecture details, and update schedules are not fully under the developer's control. Let $\mathcal{G}_{\text{ext}} : \mathcal{X} \to \mathcal{Y}$ denote an externally hosted generation function mapping input prompts $\mathcal{X}$ to outputs $\mathcal{Y}$. The system's output quality is bounded by:

$$Q(\text{MODULO}) \leq \max\{Q_{\text{local}},\ Q(\mathcal{G}_{\text{ext}})\}, \tag{15.1}$$

where Q_{local} is the quality achievable through local symbolic processing (chord generation,

voice-leading, mixing) and $Q(\mathcal{G}_{\text{ext}})$ is the quality of the external generative outputs. When the external service degrades, introduces breaking changes, or becomes unavailable, the cloud-dependent features are rendered inoperable.

Cloud inference also introduces per-request financial costs that scale with usage volume. For sustained professional use, these costs may become non-negligible, though they remain substantially below the compute investment required to train and host equivalent models independently.

On the benefit side, this approach leverages state-of-the-art models without the multi-million-dollar training budgets they require. The system can adopt improved external models as they become available, without retraining. The locally hosted components (chord generation, voice-leading harmonization, mixing, and the full symbolic editing pipeline) remain fully operational without network connectivity, ensuring that the core creative workflow is not contingent on external service availability.

15.2 DESKTOP ARCHITECTURE AND PLATFORM

MODULO is implemented as a native desktop application built on a compiled systems-level language and a professional audio application framework. This architectural choice yields low-latency audio processing, direct hardware access for MIDI controllers and audio interfaces, and the ability to host third-party instrument and effect plugins [94, 8], capabilities that browser-based competitors cannot match.

However, the desktop architecture imposes constraints. The current release targets a single operating system family, limiting the addressable user population. Cross-platform compilation is technically feasible given the portability of the underlying framework, but it has not been validated. Installation requires downloading and running a native binary, a process that is friction-laden compared to the zero-install experience of web-based tools. The absence of notarization by the platform vendor further increases the installation barrier,

as users must navigate security override procedures.

The desktop model also precludes real-time multi-user collaboration, a feature that web-based production tools increasingly support. While the local-first architecture provides strong privacy guarantees, with no audio or project data leaving the user's machine during local operations, it does so at the cost of collaborative functionality.

Let $\mathcal{U}_{\text{reach}}$ denote the set of users who can access a given tool. For a web-based tool W and a desktop tool D:

$$|\mathcal{U}_{\text{reach}}(W)| \gg |\mathcal{U}_{\text{reach}}(D)|, \tag{15.2}$$

reflecting the lower access barrier of browser-based delivery. The industry trend toward web-based music production environments [98, 71] suggests that this reach gap will widen unless addressed by a deployment strategy shift (see Chapter 16, Section 16.4).

15.3 EVALUATION SCOPE AND GENERALIZABILITY

Four of the five completed component studies, namely chord generation, harmony generation, audio-to-MIDI transcription, and stem separation, each enrolled $N = 25$ participants, a sample size that proved sufficient for the very large effects observed on efficiency and workload measures: $d_z > 1.4$ on primary behavioral metrics in all four studies; see Table 4.1, Tables 6.2–6.4, Tables 7.5–7.10, and Section 8.4. The fifth, the composition pipeline study, enrolled $N = 10$ participants; its within-subjects design still produced very large effects on per-brief time and workload (see Section 10.11). The observed effects on the primary behavioural metrics were uniformly large ($|d_z| > 1.4$ on headline contrasts in all four $N = 25$ studies), so the sample sizes proved adequate for those comparisons. However, the audio-to-MIDI study demonstrated that accuracy differences between AI-assisted conditions can be small, with $d_z = 0.35$ for F_1 (§7.10.13), suggesting that future accuracy-

focused studies may require larger samples to detect meaningful differences. The final integrated evaluation (Chapter 13) deliberately adopts a small-N design with $N = 5$ professional musicians, one distinct workflow archetype each, optimized for critical-incident surfacing and deep qualitative analysis rather than for null-hypothesis testing; its purpose is session-level observation of the composed system, and its findings should be read as descriptive rather than confirmatory. The sound-effects generator, mixer, and musician-facing UI are presented as design chapters in Chapters 9, 11, and 12, whose usability is exercised only within that final integrated session; no separate controlled comparison against alternative implementations is undertaken for those three subsystems.

Demographic breadth is a second concern. Participants were recruited from a university and professional populations, introducing potential selection biases: the sample skews younger and more technically literate than the general population of music practitioners. Genre preferences, production experience distributions, and aesthetic sensibilities within the sample may not represent the diversity of the broader target user base.

The stimulus sets for the completed studies comprised fixed melodies and audio clips: two melodies for the chord study in C major and F minor; three melodies for the harmony study in C major, F minor, and Db major at varying tempi; and four audio clips for the audio-to-MIDI study covering cello solo, electric guitar blues, vocals, and piano in Db major. While these designs controlled for melodic content across conditions, they limit the generalizability of the findings to other melodic contexts, key signatures, tempi, and genres. A more comprehensive stimulus set spanning diverse musical idioms would strengthen the external validity of the conclusions.

All evaluations are single-session designs. Participants interact with the system for a bounded period under laboratory conditions, which cannot capture the longitudinal dynamics of tool adoption: the learning curve, the development of expertise-specific workflows, and the potential for novelty effects to inflate initial ratings. A practitioner's relationship with a production tool evolves over weeks and months, whereas single-session studies cap-

ture only the first impression.

15.4 THINK-ALOUD PROTOCOL REACTIVITY

The concurrent think-aloud protocol [37] employed in the chord generation study (Chapter 4) asked participants to verbalize their thought processes while composing. This methodological choice provides rich qualitative data but introduces a reactivity concern: the act of articulating one's reasoning may alter the reasoning itself.

Specifically, verbalization may promote more deliberate and reflective decision-making than would occur under naturalistic silent conditions. Participants who are asked to explain their choices may explore more options before committing, artificially inflating the exploration metrics that serve as primary dependent variables. If this reactivity bias is condition-dependent, being larger in the treatment condition where more options are available to discuss, the treatment effect estimate may be inflated.

However, the within-subjects design partially mitigates this concern: if the think-aloud protocol uniformly inflates exploration across both conditions, the between-condition contrast remains valid. The magnitude of the observed effects, with $d_z > 2$ on primary measures (§4.9.1), also provides a buffer against plausible reactivity magnitudes.

15.5 MUSICAL DOMAIN COVERAGE

15.5.1 Scale and Mode Support

The chord generation and voice-leading subsystems currently operate within the diatonic framework [4], supporting major and natural minor modes. The out-of-key indicator in the chord editing environment (Chapter 5) flags chords whose constituent pitch classes fall outside the diatonic pitch-class set $\mathcal{P}_{\text{diatonic}} \subset \mathbb{Z}_{12}$. This binary classification is appropriate

for diatonic harmony but inadequate for idioms that employ non-diatonic scales:

$$\mathcal{P}_{\text{harm. min.}} = \mathcal{P}_{\text{nat. min.}} \cup \{\hat{7}\}, \quad \mathcal{P}_{\text{mel. min.}} = \mathcal{P}_{\text{nat. min.}} \cup \{\hat{6}, \hat{7}\}, \tag{15.3}$$

where $\hat{6}$ and $\hat{7}$ denote the raised sixth and seventh scale degrees, respectively. Modal scales (Dorian, Mixolydian, Lydian, Phrygian) and non-Western tuning systems are similarly unsupported, limiting the system's applicability for jazz, world music, and contemporary art music practitioners.

15.5.2 Symbolic Domain Constraint

The chord generation, voice-leading, and composition planning subsystems operate entirely within the symbolic (MIDI) domain. They manipulate discrete pitch, onset, and duration events rather than continuous audio waveforms. This constraint means that the selective commitment paradigm (in which the user auditions multiple candidates and commits at the individual-element level) is available only for symbolic features.

Audio-domain features (music generation, sound effects synthesis, stem separation) produce waveform outputs that can be accepted or rejected but not edited at the sub-event level within the generative interface. The user can, of course, edit the resulting audio using conventional waveform editing tools within the production environment, but the structured comparison-and-commitment workflow that proved effective for symbolic harmony does not extend to the audio domain in the current implementation.

15.5.3 Harmony Engine Determinism

The rule-based harmony engine (Chapter 6) produces deterministic output for a given melody and strategy selection. For any fixed strategy, the same melody input always yields the same harmony line, offering no variation between successive invocations. Participants in the harmony study noted this limitation specifically for the adaptive strategy, expressing a

preference for multiple distinct adaptive outputs that could be compared and selected (Section 6.8.7). Additionally, participants observed that the engine produces uniformly dense voicings throughout a passage without accounting for arranging conventions that call for selective thinning at moments where the lead voice should stand out. Both limitations point toward the stochastic and density-aware extensions described in Section 16.2.

15.5.4 Evaluation Timing

Subjective ratings in all studies are collected immediately following task completion. This design captures the participant's immediate affective and evaluative response but cannot assess whether preferences persist over time. Recency effects, novelty bias, and the temporary salience of positive (or negative) experiences may distort immediate post-task ratings relative to settled long-term opinions. Ecological momentary assessment or delayed follow-up surveys would provide more robust estimates of sustained user satisfaction but lie outside the scope of the current evaluation plan.

15.6 BASELINE STRENGTH AND ECOLOGICAL VALIDITY OF CONTROLS

The control conditions across the five component studies were chosen to be realistic and reproducible, but they are not in every case the strongest possible baselines a practitioner might use.

Chord and harmony studies. The control condition in both studies is manual note-by-note entry in the MODULO piano roll. This represents a real workflow—many producers do enter chords and harmonies one note at a time—but it is not the *only* realistic workflow. A musician with a MIDI keyboard could play chords in real time, bypassing the note-entry bottleneck entirely. Tools such as Hookpad [49], which provide a chord-palette

interface with theory-aware suggestions, represent an intermediate baseline that is stronger than manual entry but weaker than MODULO's parallel multi-temperature generation. The studies do not test against either of these alternatives. Consequently, the large time and workload reductions should be interpreted as improvements over *manual piano-roll entry specifically*, not over all possible chord-entry workflows. Against a MIDI-keyboard baseline, the efficiency advantage would likely shrink because a skilled player can voice chords in seconds; the *exploration* advantage—the ability to audition multiple temperature-varied alternatives simultaneously—would remain because real-time keyboard performance does not naturally produce parallel candidates.

Audio-to-MIDI study. The "external" baseline uses the same Basic Pitch model accessed through an external web interface, so the comparison is between workflow integration and workflow integration—not between models. A fairer accuracy baseline would pit MODULO's transcription against a hand-corrected reference produced by a professional transcriptionist, or against a stronger model such as a fine-tuned Whisper variant or a commercial transcription service. The study was designed to establish accuracy *parity* at reduced workload (hypothesis H2), and it succeeds on those terms, but the parity claim is bounded by the quality of the shared model.

Stem separation study. The external tool (Sesh FM / Demucs) is a reasonable community-standard baseline, but it is not the strongest available. Commercial offerings such as iZotope RX, LALAL.AI, and the Suno Studio stem separator have all improved substantially during the study period. The MODULO study shows that *integration* reduces workflow time and workload; it does not claim that the underlying separation quality is superior—Table 8.1 already shows that the external tools outperform on time-domain fidelity metrics.

Composition pipeline study. The external baselines are the *free tiers* of Udio and Suno, which impose generation-count limits, queue delays, and restricted output lengths. A comparison against Suno's paid *Premier* tier or Udio's *Pro* tier—which offer faster generation, longer outputs, and inpainting—would likely narrow the efficiency gap while preserving the structural-control advantage. The study is transparent about this: the free-tier comparison establishes a floor, not a ceiling, for what external tools can deliver.

In summary, the baselines were chosen to be reproducible, widely available, and representative of the workflows that the target population actually uses today; but they are not the strongest possible comparators in every case. A reader should calibrate the reported effect sizes accordingly: the qualitative direction of the findings is robust, but the magnitudes are specific to the stated baselines.

15.7 COUPLING OF INTERFACE AND MODELING DECISIONS

Several of the dissertation's claims are framed as *interface* contributions: the selective-commitment chord workshop, the five-strategy harmony-engine panel, the one-click stem separator, the musician-facing hotkey vocabulary. However, these interface designs are tightly coupled to the modeling decisions underneath them. The chord workshop's value depends on the multi-temperature transformer generating meaningfully different candidates at each τ; the harmony-engine panel's value depends on the scoring function producing consonant, voice-led output; the one-click separator's value depends on the cloud-hosted separation model returning production-usable stems.

A stronger framing is that MODULO's contribution lies in the *joint* design of model and interface: the claim is not that the interface alone is novel, nor that the models alone are novel, but that the pairing of a specific modeling strategy with a specific interaction paradigm produces a workflow that neither component achieves in isolation. The selective-

commitment paradigm is only useful if the generator produces diverse candidates; the generator is only useful in a DAW if the interface lets a musician audition and commit at the individual-chord level. This joint-design perspective is the more defensible scientific claim, and it applies across most of the evaluated subsystems.

The user studies cannot fully disentangle interface effects from modeling effects because the treatment condition always bundles both. A factorial design that crosses interface type (selective commitment vs. sequential single-option) with model type (multi-temperature vs. greedy) would be needed to isolate the contribution of each factor. That design was beyond the scope of this thesis but would strengthen the causal claims in future work.

15.8 WHAT MODULO DOES NOT SOLVE

A balanced assessment requires identifying the problems that MODULO does not address, does not know how to address, or has not been tested on.

Compositional intent. MODULO accelerates the *mechanics* of composition—chord voicing, harmony generation, stem management, mixing—but it does not help a musician decide *what to compose*. The composition planner produces section-level blueprints from structural cues, but it has no model of narrative arc, emotional trajectory, or genre convention. A songwriter facing a blank page gets faster tools once they have an idea, but no help arriving at one.

Expressive performance. The system operates in the symbolic MIDI domain for its core editing features. MIDI captures pitch, onset, duration, and velocity, but it does not capture the continuous expressive dimensions—vibrato, breath control, micro-timing, timbral shading—that distinguish a mechanical rendering from a musical performance. MODULO can host VST instruments that respond to these parameters, but the generative features

(chord, harmony, composition planner) produce quantized MIDI output with no expressive nuance.

Genre breadth. The evaluation stimuli and participant backgrounds skew toward Western popular, cinematic, and singer-songwriter idioms. The system has not been tested on jazz improvisation workflows, classical orchestration, electronic sound-design-heavy production (e.g., Aphex Twin-style granular synthesis), hip-hop beat construction with sample chopping, or any non-Western musical tradition. The diatonic-only constraint of the chord generator and harmony engine (Section 15.5) would be a hard barrier for several of these genres.

Collaboration. MODULO is a single-user, single-machine tool. It has no real-time collaboration, no cloud sync, no shared project files. In an era where Splice, BandLab, and Soundtrap have normalized cloud-based collaborative production, this is a significant gap for any user whose workflow involves co-writers or remote collaborators.

Audio-domain selective commitment. The selective-commitment paradigm, which proved effective for symbolic features (chord workshop, harmony engine), does not extend to audio-domain features. A musician cannot audition five temperature-varied *audio* generations side by side in a grid and commit at the bar level; they can only accept or reject each generation as a whole. Extending selective commitment to audio is an open problem that requires either fast enough generation to produce multiple candidates in real time or a latent-space representation that supports fine-grained editing.

15.9 INTERPRETIVE GUIDANCE: SURPRISING AND UNSURPRISING FINDINGS

Not all of the reported findings carry equal informational weight. Some results are straightforward consequences of the system's design and would have been surprising only if they had *not* appeared; others are genuinely informative. A reader calibrating the dissertation's contribution should distinguish between the two.

Unsurprising findings. The large time reductions in the harmony study (-81%) and stem-separation study (-74%) are expected outcomes of replacing manual note-by-note entry or external-tool round-trips with automated in-DAW operations. Any functional automation would produce time savings against a manual baseline; the direction of the effect is not in doubt. Similarly, the near-perfect consonance ratio in the harmony engine's treatment arm (0.99) is a direct consequence of the scoring function's consonance constraint—the engine is *designed* to produce consonant output, so finding that it does is confirmatory rather than revelatory. The composition-pipeline time reductions are largely explained by the elimination of browser round-trip latency, a structural rather than cognitive improvement. These findings are necessary to report for completeness, but they should not be mistaken for discoveries.

Genuinely informative findings. The reinvestment-into-exploration pattern (Pattern 1 in Section 14.4) is non-obvious: one might have predicted that faster tools would simply let musicians finish earlier, but instead they spent the saved time exploring alternatives. The voice-independence negative result is informative because it reveals a concrete trade-off in the engine's design that was not predicted a priori. The chord-study finding that harmonic vocabulary expanded 3–$4\times$ without a proportional increase in root-motion entropy is informative because it distinguishes genuine exploration from random drift. The unanimous

real-work preference in the A2M and composition-pipeline studies, despite known accuracy and quality limitations, is informative because it suggests that integration convenience dominates quality differentials in practitioner decision-making. The hotkey adoption rate in the integrated study (20% of inputs via keyboard in a first session) is informative because the literature on shortcut adoption suggests much slower uptake for unfamiliar vocabularies.

Findings limited by weak baselines. The magnitude of the efficiency gains should be read in the context of Section 15.6. Against stronger baselines—MIDI keyboard entry for chords, Hookpad for harmony, paid tiers for composition—the magnitudes would likely shrink while the directions would hold. The strongest claims are those that are not reducible to baseline weakness: the exploration-breadth finding in the chord study, the voice-independence trade-off in the harmony study, and the integration-preference finding across all studies.

15.10 WHERE MODULO COULD NOT REASONABLY LOSE

In several of the component studies, the comparison was tilted toward MODULO by the structure of the design rather than by a property of the system being tested, and a careful reader should know which numbers carry discriminating evidence and which are essentially measurements of an unfair race. Each study chapter now contains an explicit subsection on this point (Sections 4.9.5, 6.8.8, 7.10.13, 8.5.4, and the analogous discussion in Section 10.11.8); this section consolidates the picture across the five studies.

Time and workload contrasts where in-DAW always beats out-of-DAW. The audio-to-MIDI built-in-vs.-external contrast, the stem-separation in-DAW-vs.-browser contrast,

and the composition-pipeline MODULO-vs.-Udio/Suno time contrasts all measure essentially the same thing: how long a musician spends switching contexts to a browser, navigating an external tool, uploading source audio, waiting for processing, and importing results back. None of these contrasts is a genuine head-to-head between two comparable workflows; they are measurements of round-trip cost. The cost is real, the workflow gain is real, and the participant preferences are real, but the *magnitudes* of these gaps are not the right object on which to compare MODULO against future systems that also ship in-DAW. Any future in-DAW competitor would close most of the gap by virtue of being in-DAW.

Diversity contrasts where parallel beats sequential by construction. The chord-generation *unique-roots* and *unique-qualities* contrasts measure how much harmonic vocabulary a participant's final progression spans. The treatment exposes five parallel temperature-varied candidates per position; the control exposes one. A participant who selectively commits across the five columns will, by construction, end up with a wider symbol vocabulary than a participant working from a single sequential candidate at a time. These are not findings that the chord transformer is a better generative model than alternatives; they are findings that the parallel-multi-temperature interface unlocks the breadth that the model already possesses.

Engine-deterministic quality metrics. The harmony-engine *consonance ratio*, *voice independence*, and *parallel perfects* contrasts have one arm whose output is determined entirely by the scoring function and strategy choice—the engine produces, by construction, near-perfect consonance, low voice independence, and structural parallel motion. These metrics test what the engine was designed to do (or not do); they are not contests against a competing model. The voice-independence and parallel-perfects results are reported as negative findings precisely because they expose places where the engine's deterministic structure produces a result the musician should be able to override.

Section-structure metrics that only MODULO supports. The composition-pipeline *structure match*, *section findability*, and *plan-to-output alignment* items measure whether a generated piece honours a section-level brief. Only MODULO ships a section-block visual grammar that makes section editing first-class; the external tools generate audio that is opaque to section-level analysis. These metrics are tests of whether MODULO's section-aware pipeline behaves as designed, not contests against external tools that lack the underlying capability.

The contestable evidence. After deducting the structurally favourable contrasts above, the discriminating findings of the dissertation are: (i) audio-to-MIDI *accuracy parity* with external tools using the same model ($d_z = 0.35$, $p = .096$, supporting H2); (ii) the chord-study *time-to-satisfaction* contrast and *subjective ratings*, where participants directly compared both conditions and could rate the control higher on perceived ownership or familiarity; (iii) the harmony-engine *voice-independence and parallel-perfects negative results* that move *against* MODULO; (iv) the stem-separation *time-domain audio-quality results*, where the external tools beat MODULO on SI-SDR, MAE, and SNR; (v) the composition-pipeline *overall output quality* contrast, where Suno's free pipeline outscores MODULO (5.20 vs. 4.20 on the five-point scale); and (vi) the integrated study's *naturalistic* session evidence—five professional musicians completing real briefs without exiting the DAW. These are the findings on which MODULO could have lost, and on (iii)–(v) it did lose; (i), (ii), and (vi) are the discriminating wins.

CHAPTER 16

Future Work

The limitations identified in Chapter 15 suggest several natural directions for extending MODULO. This chapter outlines nine areas of future development, ordered from the most architecturally proximate extensions to the most ambitious long-term research directions.

16.1 AUDIO-DOMAIN SELECTIVE COMMITMENT

The selective commitment paradigm demonstrated large effects in the symbolic domain (Chapter 4), where users compared chord candidates at the individual-event level and committed favorites from different generation columns. The harmony generation study (Chapter 6, Section 6.8.7) confirmed that structured AI assistance produces similarly large effects at the contrapuntal level, with an 81% reduction in task time at $d_z = 2.04$ and near-universal preference for the treatment condition. Extending this paradigm to audio-domain generation is the highest-priority architectural evolution.

Let $\mathcal{G}_{\text{audio}} : \mathcal{X} \to \mathcal{Y}^N$ denote a generation function that produces N parallel audio candidates from a prompt $x \in \mathcal{X}$. The symbolic commitment operator $\sigma : \mathcal{C}^N \times \{1, \ldots, N\}^T \to \mathcal{C}$ (Equation 4.1) selects individual chords from distinct columns at each time step. An analogous audio-domain operator would require:

$$\sigma_{\text{audio}} : \mathcal{Y}^N \times \mathcal{R}^M \to \mathcal{Y}, \tag{16.1}$$

where $\mathcal{R}^M$ is a set of M temporal regions, and the operator selects the preferred candidate

within each region, producing a composite output assembled from segments of different candidates.

The principal challenge is that audio signals lack the discrete, semantically labeled structure of symbolic chord sequences. Crossfading between candidates at region boundaries requires phase-coherent blending to avoid audible artifacts. A promising approach combines source separation with selective commitment: each candidate is decomposed into stem layers, and the user commits at the stem-by-region level, yielding a fine-grained commitment lattice:

$$\sigma_{\text{stem}} : \left(\mathcal{Y}^{S}_{\text{stem}}\right)^{N} \times \{1, \ldots, N\}^{S \times M} \to \mathcal{Y}^{S}_{\text{stem}}, \tag{16.2}$$

where S is the number of stems. This formulation would allow a user to select the drums from candidate one, the bass from candidate three, and the melody from candidate two, assembling a composite that no single generation produced.

16.2 PROPRIETARY HARMONY MODEL

The current system relies on a pretrained external model for chord generation (Chapter 4). Training a dedicated model on data collected from MODULO user interactions would enable several improvements.

Let $\mathcal{D} = \{(\mathbf{m}^{(i)}, \mathbf{c}^{(i)}, \mathbf{a}^{(i)})\}_{i=1}^{n}$ denote a dataset of melody–chord–action triples, where $\mathbf{a}^{(i)}$ encodes the user's selective commitment decisions: which chords were auditioned, which were committed, and which were rejected. A reward model can be trained on these implicit preferences:

$$r_{\phi}(\mathbf{c} \mid \mathbf{m}) = \mathbb{E}_{\mathbf{a} \sim \mathcal{D}}\left[\mathbb{1}[\text{chord } c_t \text{ was committed}]\right], \tag{16.3}$$

and the generation model can be fine-tuned via reinforcement learning from human feed-

back to maximize expected reward:

$$\theta^* = \arg\max_{\theta} \; \mathbb{E}_{\mathbf{m}\sim\mathcal{D}} \left[\sum_{t=1}^{T} r_\phi\big(c_t \mid \mathbf{c}_{<t}, \mathbf{m}\big) - \lambda \, D_{\mathrm{KL}}\big(p_\theta \| p_{\mathrm{ref}}\big) \right], \tag{16.4}$$

where λ controls the divergence penalty against a reference policy p_{ref} to prevent mode collapse.

A proprietary model would also enable personalization. By conditioning on a user embedding $\mathbf{u}_k$ derived from participant k's interaction history, the model could adapt to individual harmonic preferences over time:

$$p_\theta(c_t \mid \mathbf{c}_{<t}, \mathbf{m}, \mathbf{u}_k). \tag{16.5}$$

Participants in the harmony study (Chapter 6, Section 6.8.7) specifically requested two extensions that a proprietary model would enable. First, stochastic adaptive outputs for the same melody, allowing musicians to compare distinct valid harmonizations rather than receiving the single deterministic result produced by the current scoring function. Second, density-aware scoring that selectively thins harmony textures at moments of high melodic activity, preserving melody prominence in the way skilled arrangers naturally balance lead and accompaniment voices. Both requests reinforce the motivation for moving from a rule-based engine to a learned model capable of capturing context-dependent arranging decisions.

16.3 EXTENDED SCALE AND MODE SUPPORT

As noted in Section 15.5, the current system supports only major and natural minor diatonic modes. Extending the harmonic framework to additional scales requires generalizing the pitch-class set used by the out-of-key detection logic and the voice-leading algorithm.

Let $\mathcal{P}_s \subset \mathbb{Z}_{12}$ denote the pitch-class set for scale s. The generalized consonance func-

tion becomes:

$$\kappa_s(c) = \frac{|\text{PC}(c) \cap \mathcal{P}_s|}{|\text{PC}(c)|}, \tag{16.6}$$

where $\text{PC}(c)$ extracts the pitch classes of chord c. A chord is flagged as out-of-scale when $\kappa_s(c) < 1$. For modal scales, this generalization is straightforward: Dorian on D uses $\mathcal{P}_{\text{Dor}} = \{2, 4, 5, 7, 9, 11, 0\}$, and the existing voice-leading constraints apply without modification.

Harmonic and melodic minor scales introduce variable scale degrees that depend on melodic context, ascending versus descending [78], requiring a context-sensitive pitch-class set:

$$\mathcal{P}_{\text{mel. min.}}(t) = \begin{cases} \mathcal{P}_{\text{nat. min.}} \cup \{\hat{6}, \hat{7}\} & \text{if ascending at frame } t, \\ \mathcal{P}_{\text{nat. min.}} & \text{if descending at frame } t. \end{cases} \tag{16.7}$$

Pentatonic and blues scales, which contain fewer than seven pitch classes, require adapted voice-leading rules to handle the sparser harmonic material.

16.4 WEB DEPLOYMENT

Porting the system to the browser would address the reach limitation identified in Section 15.2, following the model of web-based competitors [98, 71, 105]. Modern compilation toolchains can translate systems-level code to a format executable in the browser runtime, and the standardized browser audio processing interface provides real-time synthesis and effects processing with acceptable latency for many production tasks.

The migration path involves three architectural layers. First, the core audio engine and symbolic processing modules must be compiled for the browser execution target. Second, the plugin hosting infrastructure (which currently loads native binary modules) must be replaced with a browser-compatible alternative or omitted in the web version. Third, the cloud-hosted inference endpoints already used for music generation and sound effects can be retained without modification, as they are accessed via standard network protocols.

A web deployment also enables real-time collaborative composition (Section 16.7), as the browser environment provides natural support for bidirectional communication via persistent socket connections and operational transformation algorithms.

16.5 LONGITUDINAL DEPLOYMENT STUDIES

The single-session evaluation paradigm employed throughout this thesis (Chapters 4–13) captures first-impression effects but cannot characterize how tool usage evolves with sustained practice. Longitudinal deployment studies would address this gap.

A proposed design distributes the tool to N practicing musicians for a period of W weeks, instrumenting the application to log anonymized interaction telemetry with informed consent. Key longitudinal metrics include:

$$F_{\text{util}}(w) = |\{f \in \mathcal{F} : f \text{ invoked during week } w\}|, \tag{16.8}$$

$$\tau_{\text{gen}}(w) = \frac{\text{generative feature invocations in week } w}{\text{total feature invocations in week } w}, \tag{16.9}$$

where $F_{\text{util}}(w)$ tracks feature adoption breadth over time and $\tau_{\text{gen}}(w)$ measures the proportion of interactions involving generative capabilities. A declining τ_{gen} would suggest that generative features are primarily useful for initial exploration, while a stable or increasing ratio would indicate sustained integration into production workflows.

16.6 MULTI-INSTRUMENT WORKFLOWS

The current system's generative features are optimized for harmonic and melodic content typical of keyboard and guitar workflows. Extending to orchestral instruments, percussion, and voice would broaden the creative scope.

Drum pattern generation can be formulated as a constrained sequence model operating over a percussive event vocabulary $\mathcal{V}_{\text{drum}} = \{v_1, \ldots, v_P\}$ corresponding to P percussion

instruments. The generation objective is:

$$p(\mathbf{d} \mid \mathbf{c}, \text{tempo}, \text{genre}) = \prod_{t=1}^{T} \prod_{p=1}^{P} p(d_{t,p} \mid \mathbf{d}_{<t}, \mathbf{c}, \text{tempo}, \text{genre}), \tag{16.10}$$

where $d_{t,p} \in \{0, 1\}$ indicates whether percussion instrument p is struck at frame t.

Bass line generation introduces a tighter coupling with the harmonic content, as idiomatic bass lines typically outline chord tones with passing and approach tones:

$$p(b_t \mid \mathbf{b}_{<t}, c_t, \mathbf{m}) \propto \exp\big(\alpha \mathbb{1}[b_t \in \text{PC}(c_t)] + \beta \, \text{smoothness}(b_t, b_{t-1})\big), \tag{16.11}$$

where α and β weight chord-tone adherence against voice-leading smoothness.

16.7 COLLABORATIVE COMPOSITION

Real-time multi-user editing, analogous to collaborative document editing, would enable ensemble composition workflows in which multiple musicians contribute simultaneously [96].

The technical foundation is operational transformation or conflict-free replicated data types applied to the symbolic event representation. Let $\mathcal{O} = \{o_1, o_2, \ldots\}$ be a stream of editing operations (insert note, delete note, modify pitch, change chord) from multiple concurrent users. The consistency requirement is that all clients converge to the same document state after processing all operations:

$$\forall \text{ orderings } \pi \text{ of } \mathcal{O}: \quad \text{apply}(\pi(\mathcal{O}), D_0) = D_{\text{final}}, \tag{16.12}$$

where D_0 is the initial document state.

The selective commitment mechanism introduces an additional coordination challenge: when two users simultaneously generate chord candidates for the same passage, a merge

policy must resolve conflicts. A reservation-based protocol, in which a user locks a temporal region during active generation and commitment, provides a pragmatic solution at the cost of reduced concurrency.

16.8 INTELLIGENT PRODUCTION AUTOMATION

The mixing and mastering stages of music production involve numerous parameter decisions (gain levels, equalization curves, spatial positioning, dynamic range compression) that are amenable to machine-learning-assisted automation [107, 55].

Let $\mathbf{x} \in \mathbb{R}^{K \times T_{\text{audio}}}$ denote a K-track session and $\boldsymbol{\theta}_{\text{mix}} \in \Theta_{\text{mix}}$ the vector of mixing parameters. An automated mixing system learns a mapping:

$$g_\psi : \mathcal{X} \times \mathcal{S}_{\text{ref}} \to \Theta_{\text{mix}}, \tag{16.13}$$

where $\mathcal{S}_{\text{ref}}$ is a set of reference mixes that encode the target aesthetic. The system analyzes spectral, dynamic, and spatial characteristics of the reference and proposes parameter settings that approximate the reference's timbral profile while respecting the source material.

Automatic gain staging, the initialization of track levels to ensure headroom and appropriate balance, can be formulated as a constrained optimization:

$$\boldsymbol{\theta}^*_{\text{gain}} = \arg\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} \big(\text{LUFS}_k(\boldsymbol{\theta}) - \ell^*_k\big)^2 \quad \text{s.t.} \quad \text{LUFS}_{\text{sum}}(\boldsymbol{\theta}) \leq \ell_{\max}, \tag{16.14}$$

where LUFS_k is the integrated loudness of track k, ℓ^*_k is the target loudness for that track's role (e.g., lead vocal, accompaniment, percussion), and $\ell_{\max}$ is the master bus ceiling.

Style transfer for production workflows extends this concept by learning a transformation from a source production style to a target style, enabling practitioners to rapidly prototype different aesthetic directions without manually adjusting dozens of parameters.

16.9 AGENTIC WORKFLOWS, CHAINED TOOLS, AND MODEL CONTEXT PROTOCOLS

The generative features in the current system are invoked *synchronously and individually*: one button, one model call, one result placed on the timeline. This one-shot-per-button design preserves the selective-commitment paradigm validated by the chord and harmony studies in Chapters 4 and 6, but leaves a large fraction of the system's potential unrealised. A request like *"Take this loose vocal take, tighten the timing, harmonise it in the key of the existing progression, render a four-bar electronic backing track in the same tempo, separate that track into stems, and route the drums through a parallel compression bus"* is today seven or eight musician-driven button presses whose cumulative overhead is exactly the friction MODULO was designed to eliminate.

The natural evolution is to treat each generative subsystem as a *callable tool* exposed through a uniform interface, and to let a planning agent sequence those calls under the musician's supervision rather than forcing the musician to sequence them manually. Three recent research lines converge on this architecture: Model Context Protocol (MCP) servers that expose local capabilities to Large Language Models (LLMs) through a structured JSON-RPC schema [6], self-taught tool use as demonstrated by Toolformer [85], and the reasoning-and-acting loop made concrete by ReAct [113]. HuggingGPT [88], MusicAgent [114], and Loop Copilot [116] demonstrate that this agentic pattern transfers specifically to music generation: an LLM orchestrator plans a sequence of specialist-model calls, dispatches them, inspects the results, and iterates until the user's musical intent is met.

A natural realisation inside MODULO would decompose into three layers:

1. **An MCP server surface** that exposes every generative subsystem as a typed tool (chord generation parameterised by melody, style, and temperature; melody harmonisation parameterised by strategy; audio-to-MIDI transcription; prompt-, duration-,

and structure-conditioned music generation; mode-parameterised stem separation; prompt-conditioned sound-effects generation; brief-conditioned composition planning) alongside a smaller set of DAW actions such as inserting a track, routing it to a bus, setting a clip's timeline position, and applying an effects chain.

2. **A lightweight Python worker layer** that holds the glue logic between the typed tool calls and the native C++ components, runs inside the existing audio-analysis virtual environment, and shares the retry/caching/fingerprint infrastructure the current generative services already use (Chapter 8, Section 8.2.2). This layer is the natural home for chained tool invocations: once stems are separated, run a transient-detection pass; once the transcription is returned, run the harmoniser on the transcribed melody; once a music-generation render is produced, run stem separation in the background so the stems are pre-cached if the musician later asks for them.

3. **An optional agentic planner** that accepts a natural-language brief (*"make a sixty-second electronic track, extract the bass, and route it to a side-chained compressor keyed to the kick"*), produces an MCP-tool call plan, confirms the plan with the musician, executes it, and surfaces every intermediate artefact as a first-class track or clip inside the DAW rather than burying it in opaque state.

The design constraint that distinguishes this proposal from a generic LLM agent is that *every intermediate artefact must remain inspectable, editable, and revokable in the DAW*. If the planner runs a harmoniser on the result of a transcription, the intermediate melody and the candidate harmony lines must both appear on the timeline as distinct, labelled tracks; if the musician rejects the harmony and re-runs the planner with a different instruction, the previously committed harmonisation must not be silently overwritten. This preserves the selective-commitment property the user studies repeatedly vindicated: the musician is always the arbiter of what gets committed, even when the agent is the entity that proposed the sequence of calls.

Formally, a planning agent parameterised by ϕ produces a plan distribution conditioned on a natural-language brief b and project state s:

$$p_\phi(\pi \mid b, s) = \prod_{k=1}^{n} p_\phi(\tau_{i_k} \mid \tau_{i_{<k}}, b, s), \tag{16.15}$$

where π is a sequence of typed tool calls τ_{i_k} with dependencies encoded as a DAG, and a musician-in-the-loop validator accepts, edits, or rejects. A study paralleling the chord-generation evaluation would compare (i) no agent, (ii) agent plans + musician approval, (iii) agent autonomy with post-hoc review, measuring task time, creative-ownership ratings, and the fraction of plans accepted without modification.

The infrastructure work required is meaningful but bounded: the typed MCP surface is a JSON-RPC wrapper over the existing C++ service classes; the Python worker layer reuses the virtual environment already shipped with the application for audio analysis; the agentic planner is a prompt over any general-purpose LLM with tool-use support. The experimental work required is substantial, particularly in calibrating how much autonomy to grant without eroding musician agency, and is the direction we believe most likely to turn the system from a *single-operation generative DAW* into a *task-level co-creator* while preserving every property the current design was shown to protect.

These directions are not exhaustive but represent the extensions most likely to amplify the core contribution of this thesis: the integration of generative machine learning capabilities within a production environment that preserves musician agency through structured comparison, selective commitment, and iterative refinement.

CHAPTER 17

Coda

This thesis opened with a tension. On one side, music generation had collapsed composition into consumption: type a sentence, receive a song, move on. On the other, the conventional DAW had barely changed in two decades, leaving the musician to spell out every voice-leading decision one note at a time. The space between the two was widening, and the musician was disappearing into it.

MODULO is an argument that the space between is the space worth occupying. Not automation, not labor, but a workshop in which models propose and the musician chooses. The chord workshop shows several harmonic futures at once and hands the cursor back. The harmony engine composes the inner voices and steps aside. The audio-to-MIDI pipeline hears a recording and writes down what it heard for the musician to edit. The stem separator, the sound-effects dialog, the composition planner, the mixer with its sends and its plugin host: each is a tool the musician wields, not a tool that wields the musician.

Six studies across more than a hundred sessions returned the same shape. Tasks that used to take hours took minutes. Perceived workload fell, engagement rose, and preference returned to the integrated workflow with a consistency that surprised even the author. Participants who arrived skeptical left asking when they could take the build home. The feature set, whose effects were measured one subsystem at a time, composed in the integrated evaluation into a single session where every subsystem was reached for, used, and trusted.

What closes here is the argument that a musician and a model can work on the same

score, at the same time, without either one losing the thread. The chord cell is editable. The plan is reshape-able. The stems sit on separate tracks. The MIDI is human-readable. The mixer is a mixer. Every decision that used to be expensive is still a decision; it is just no longer expensive. And the last faders, the last notes, the last clicks on the transport before the session renders, still belong to the person in the chair.

The music is yours. Compose.

CHAPTER A

List of Acronyms

Table A.1 defines every acronym and initialism used in the body of this thesis. Only those appearing in the main text are listed; venue and conference names used only in citations are excluded.

Table A.1: Acronyms and initialisms used in this thesis.

Acronym	Definition
A2M	Audio-to-MIDI (transcription of recorded audio into symbolic note events)
AI	Artificial Intelligence
ANOVA	Analysis of Variance
API	Application Programming Interface
AU	Audio Units (Apple's native plugin format)
BH	Benjamini–Hochberg (false discovery rate correction procedure)
BH-FDR	Benjamini–Hochberg False Discovery Rate correction
BPM	Beats Per Minute
CNN	Convolutional Neural Network
CQT	Constant-Q Transform
DAG	Directed Acyclic Graph
DAW	Digital Audio Workstation

Continued on next page

Acronym	Definition
dB	Decibel
FFN	Feed-Forward Network
FLAC	Free Lossless Audio Codec
FX	Effects (audio signal processing chain)
GAPT	Generative Adversarial Post-Training
GAW	Generative Audio Workstation
GPL	GNU General Public License
GUI	Graphical User Interface
HCQT	Harmonic Constant-Q Transform
Hz	Hertz (cycles per second)
JSON	JavaScript Object Notation
KL	Kullback–Leibler (divergence)
LLM	Large Language Model
LM	Language Model
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MHA	Multi-Head Attention
MIDI	Musical Instrument Digital Interface
MLE	Maximum Likelihood Estimation
NASA-TLX	National Aeronautics and Space Administration Task Load Index
PCM	Pulse-Code Modulation
PESQ	Perceptual Evaluation of Speech Quality
PPO	Proximal Policy Optimization
RL	Reinforcement Learning

Continued on next page

Acronym	Definition
RM-ANOVA	Repeated-Measures Analysis of Variance
RMSE	Root Mean Square Error
SDR	Signal-to-Distortion Ratio
SFX	Sound Effects
SHA	Secure Hash Algorithm (e.g., SHA-1)
SI-SDR	Scale-Invariant Signal-to-Distortion Ratio
SNR	Signal-to-Noise Ratio
STOI	Short-Time Objective Intelligibility
SUS	System Usability Scale
TLX	Task Load Index (see NASA-TLX)
TTA	Text-to-Audio (generation)
UI	User Interface
UES	User Engagement Scale
UES-SF	User Engagement Scale, Short Form
VST	Virtual Studio Technology (Steinberg's plugin standard)
WAV	Waveform Audio File Format

CHAPTER B

Glossary of Terms

This glossary defines musical, audio-signal-processing, machine-learning, and human–computer-interaction terms used across the thesis. Only terms that appear explicitly in the main text are listed. Each entry notes the chapter that most prominently introduces or uses the term.

Table B.1: Glossary of musical and technical terms used in this thesis.

Term	Definition
Adaptive scoring function	A weighted linear combination of consonance, smoothness, and contrary-motion criteria that MODULO's harmony engine uses to select the best harmony note at each melodic position. (Ch. 6)
Affordance	A visual or interactive property of an interface element that signals how it can be used; e.g. a drag handle that invites dragging. (Ch. 12)
Arpeggio (playback style)	A chord realisation in which notes are sounded sequentially in ascending pitch order rather than simultaneously, producing a broken-chord texture. (Ch. 4)

Continued on next page

Term	Definition
Audio Unit (AU)	Apple's native macOS/iOS plugin standard for audio effect and virtual-instrument processors, enabling third-party software to be hosted inside a DAW. (Ch. 3)
Audio-to-MIDI transcription (A2M)	The automated conversion of a recorded audio signal into a symbolic MIDI representation, recovering pitch, onset, offset, and velocity for each note. (Ch. 7)
Autoregressive model	A generative sequence model that produces tokens one at a time, each conditioned on all previously generated tokens; MODULO's chord transformer is autoregressive. (Ch. 2)
Automation envelope	A piecewise-linear curve stored on the DAW timeline that animates a mix parameter (volume, pan, send level) during playback. (Ch. 11)
Auxiliary bus (send/return)	A shared routing path in a DAW mixer to which multiple tracks can send a portion of their audio; typically used to share one effect (e.g. a reverb) across several tracks. (Ch. 11)
Bar (measure)	A metrical unit containing a fixed number of beats as defined by the time signature; a 4/4 bar holds four quarter-note beats. (Ch. 1)
Beat	The basic pulse unit within a bar; tempo is measured in beats per minute (BPM). (Ch. 4)
Benjamini–Hochberg FDR (BH-FDR)	A multiple-comparison correction that controls the false discovery rate across a family of p-values; applied to every family of paired t-tests in the evaluation studies. (Ch. 6, Ch. 14)

Continued on next page

Term	Definition
Block (playback style)	A chord realisation in which all chord tones are sounded simultaneously and sustained for the full chord duration. (Ch. 4)
Buffer size	The number of audio samples processed in a single real-time audio callback; smaller buffers reduce latency but raise the risk of dropouts. (Ch. 3)
Buffer underrun	An audible click, dropout, or silence produced when the CPU cannot fill the output buffer in time. (Ch. 3)
Channel strip	A vertical panel in the DAW mixer that represents a single track or bus; typically hosts a gain fader, pan knob, mute/solo/record-arm buttons, sends, and a level meter. (Ch. 11)
Chord inversion	A voicing in which a chord tone other than the root sits in the bass; first inversion puts the third in the bass, second inversion puts the fifth. (Ch. 5)
Chord quality	The intervallic character that distinguishes major, minor, dominant seventh, diminished, augmented, suspended, and other chord types. (Ch. 1)
Chord Workshop	MODULO's direct-manipulation harmonic editor in which accompaniment is presented as a sequence of labelled chord cells parameterised by root, quality, inversion, octave, duration, and playback style. (Ch. 5)
Chromagram	A time-frequency representation that aggregates spectral energy into twelve pitch-class bins, summarising harmonic content at each time frame. (Ch. 2)

Continued on next page

Term	Definition
Chromatic	Relating to pitches or intervals outside the current diatonic key; a chromatic chord contains at least one pitch class not in the prevailing scale. (Ch. 5)
Classifier-free guidance (CFG)	An inference technique that combines conditioned and unconditioned model predictions to sharpen adherence to a conditioning signal such as a text prompt or melody. (Ch. 2)
Clip (region)	A self-contained segment of audio or MIDI data placed on a track at a specific timeline position; the fundamental unit of non-destructive arrangement. (Ch. 3)
Cohen's d_z	The within-subjects effect-size measure used throughout the evaluation, computed as the mean paired difference divided by the standard deviation of the paired differences; values ≥ 0.8 are conventionally "large". (Ch. 6, Ch. 14)
Composition planner	MODULO's subsystem that converts a free-form natural-language creative brief into a structured section-level blueprint (intro, verse, chorus, bridge, outro) with per-section style tags and durations. (Ch. 10)
Consonance / dissonance	Perceptual qualities of musical intervals: consonant intervals (thirds, sixths, perfect fifths, octaves) are judged stable; dissonant intervals (seconds, sevenths, tritones) are judged in need of resolution. (Ch. 6)
Constant-Q Transform (CQT)	A time-frequency analysis whose frequency bins are spaced logarithmically, so that each octave contains a fixed number of bins; closely aligned with musical pitch perception. (Ch. 7)

Continued on next page

Term	Definition
Contrary motion	A voice-leading relationship in which two voices move in opposite directions; considered the most independent and desirable type of motion in classical counterpoint. (Ch. 6)
Diatonic	Belonging to a given major or minor scale; a diatonic chord uses only pitch classes present in the prevailing key. (Ch. 1)
Direct manipulation	An interaction paradigm in which objects of interest are visible on screen and respond immediately and reversibly to physical actions (drag, click, resize). (Ch. 12)
Double-time (playback style)	A block voicing re-attacked at twice the harmonic rhythm, dividing the chord's duration into two equal re-struck segments. (Ch. 4)
Effect size	A standardised measure of the magnitude of an experimental result independent of sample size; Cohen's d_z is the effect-size measure used throughout this thesis. (Ch. 14)
Embedding	A dense, learned vector representation of a discrete token (a chord symbol, a text word) in a high-dimensional continuous space; used as input to neural network layers. (Ch. 2)
Parametric equaliser (EQ)	A signal-processing tool that boosts or cuts amplitude at specific frequency bands; MODULO provides four independent bands per track, each with centre frequency, gain, and Q. (Ch. 11)
Fine-tuning	Adapting a pretrained model to a specific task or preference by continued training on a smaller, curated dataset or with a reinforcement-learning reward signal. (Ch. 2)

Continued on next page

Term	Definition
Forced-choice	An experimental response format in which participants must pick one option from a fixed set, with no neutral option allowed. (Ch. 8, Ch. 13)
Formant	A resonance peak in the spectral envelope of a vocal or instrumental sound; formant position largely determines timbre and vowel quality. (Ch. 8)
F_1 score	The harmonic mean of precision and recall, used here as the primary accuracy metric for audio-to-MIDI transcription. (Ch. 7)
Generative Audio Workstation (GAW)	A software system that combines a full DAW production environment with integrated generative-AI services and a co-creation interface layer supporting inspection, comparison, and selective editing of generated material. (Ch. 1)
Harmonic Constant-Q Transform (hCQT)	An extension of the CQT that stacks multiple CQTs computed at harmonic frequency multiples ($0.5\times$, $1\times$, $2\times$, $3\times$, $4\times$, $5\times$), giving a neural transcription model explicit harmonic-series information. (Ch. 7)
Harmony engine	MODULO's rule-based module that generates up to five simultaneous harmony voices accompanying a melody, using classical voice-leading principles encoded as a weighted scoring function. (Ch. 6)
Idea marker	A lightweight timestamped annotation placed on the MODULO timeline at a specific position, optionally labelled, that lets the composer flag moments for later review. (Ch. 12)

Continued on next page

Term	Definition
Interquartile range (IQR)	The range between the 25th and 75th percentiles of a dataset; a robust measure of spread reported in lieu of standard deviations in the small-N integrated evaluation. (Ch. 13)
Key signature	The set of sharps or flats that defines the tonal centre of a piece; e.g. two sharps indicates D major or B minor. (Ch. 1)
Latency	The delay between a user action (a MIDI key press) and the corresponding audio output; equal to buffer size divided by sample rate. (Ch. 3)
Likert scale	A rating scale in which respondents indicate agreement or satisfaction on an ordered set of options (5 or 7 points); used for all custom subjective items in MODULO's user studies. (Ch. 6)
Logits	The unnormalised scalar outputs of a neural network's final linear layer before softmax. (Ch. 2)
Lookahead window	In online chord generation, the number of future melody frames the model observes before committing a chord at the current position. (Ch. 4)
LUFS	Loudness Units relative to Full Scale: an integrated loudness measurement aligned with human perception, used in broadcast and streaming standards. (Ch. 16)
Mel-spectrogram	A spectrogram whose frequency axis is warped to the mel scale, approximating the non-linear frequency resolution of human hearing. (Ch. 7)

Continued on next page

Term	Definition
MIDI quantisation	The alignment of recorded MIDI note onsets and offsets to the nearest point on a rhythmic grid (e.g. sixteenth notes), correcting timing imprecision while preserving musical intent. (Ch. 3, Ch. 7)
Mode (musical)	A scale derived by starting the major-scale pattern on a different scale degree; examples include Dorian and Mixolydian. MODULO supports major and natural minor. (Ch. 6, Ch. 15)
Multi-head self-attention	A neural network mechanism in which an input sequence attends to itself at multiple representation subspaces in parallel; the core building block of transformers. (Ch. 2)
NASA-TLX	The NASA Task Load Index, a six-subscale self-report instrument measuring perceived workload along mental demand, physical demand, temporal demand, performance, effort, and frustration. (Ch. 4 and all user-study chapters)
Non-destructive editing	A DAW editing paradigm in which all modifications are stored as reversible transformations applied to immutable source data. (Ch. 3)
Out-of-key detection	MODULO's visual feedback mechanism that highlights chord cells whose pitch-class content lies outside the current key's diatonic scale, flagging chromatic chords without preventing their use. (Ch. 5)

Continued on next page

Term	Definition
Paired t-test	A statistical test that compares two related measurements from the same participants (treatment vs. control) and assesses whether the mean paired difference is significantly different from zero. (Ch. 6, Ch. 14)
Parallel perfect fifths / octaves	A voice-leading violation in which two voices move in the same direction by the same perfect-fifth or perfect-octave interval in consecutive beats; prohibited in classical counterpoint. (Ch. 6)
PESQ	Perceptual Evaluation of Speech Quality: a reference-based perceptual metric for speech and vocal audio; scores range roughly 1–5. (Ch. 8)
Piano roll	A two-dimensional MIDI note editor with pitch on the vertical axis and time on the horizontal axis; notes appear as rectangles whose length encodes duration. (Ch. 3, Ch. 12)
Pitch class	One of the twelve distinct notes in Western equal temperament (C, C♯, D, . . . , B), independent of octave. (Ch. 1)
Plugin (VST3 / AU)	A third-party audio effect or virtual instrument that conforms to a standard interface (VST3 or Audio Unit) and can be hosted inside a DAW's signal processing graph. (Ch. 3)
Post-hoc analysis	Statistical tests or comparisons conducted after the main analysis, often to explore which specific conditions differ after a significant omnibus result. (Ch. 14)

Continued on next page

Term	Definition
Proximal Policy Optimization (PPO)	A reinforcement-learning algorithm that updates a policy by maximising a clipped importance-sampling objective; used to fine-tune ReaLChords with contrastive reward signals. (Ch. 2, Ch. 4)
Roman numeral notation	A system for labelling chords by their scale degree relative to a key using Roman numerals (I, ii, iii, IV, V, vi, vii°); uppercase indicates major quality, lowercase minor. Chord Workshop provides a toggle between absolute chord names and Roman numeral display. (Ch. 5)
Sample rate (f_s)	The number of audio samples recorded or played back per second; 44,100 Hz and 48,000 Hz are the rates used throughout MODULO. (Ch. 3)
Scale degree	The position of a pitch within a diatonic scale, numbered 1 (tonic) through 7; Roman numeral chord notation is built on scale degrees. (Ch. 5)
Selective commitment	MODULO's interaction paradigm in which the musician auditions multiple parallel chord-generation columns and can accept individual chords from different columns, assembling a composite final progression. (Ch. 4)
Send (pre-fader / post-fader)	A routing control that sends a copy of a track's signal to an auxiliary bus; pre-fader sends are tapped before the track fader, post-fader sends after it. (Ch. 11)

Continued on next page

Term	Definition
SI-SDR	Scale-Invariant Signal-to-Distortion Ratio: a reference-based audio-quality metric that measures reconstruction fidelity independent of overall amplitude scaling. (Ch. 8)
Softmax	A function that converts a vector of logits into a probability distribution by exponentiating each value and normalising. (Ch. 2)
Source separation (stem separation)	The decomposition of a mixed audio signal into its constituent instrument or vocal stems; MODULO integrates ElevenLabs-backed separation as a one-click operation. (Ch. 8)
Spectrogram	A time-frequency display showing how the spectral content of a signal changes over time, typically drawn as a heat map. (Ch. 7, Ch. 8)
STOI	Short-Time Objective Intelligibility: a reference-based speech intelligibility metric on a 0–1 scale. (Ch. 8)
SUS	System Usability Scale: a 10-item standardised questionnaire that produces a 0–100 usability score; scores above the 68-point benchmark are considered acceptable. (Ch. 13, Ch. 14)
Temperature (τ)	A scalar parameter that controls the entropy of the softmax sampling distribution; $\tau < 1$ concentrates probability on high-likelihood tokens (conservative output), $\tau > 1$ flattens it (more diverse, exploratory output). (Ch. 4)
Text-to-audio (TTA)	A generation task in which natural-language text is mapped directly to a waveform, as in MODULO's ElevenLabs-backed sound-effect generator. (Ch. 9)

Continued on next page

Term	Definition
Think-aloud protocol	A qualitative research method in which participants verbalise their thoughts continuously while performing a task, providing real-time insight into decision-making. (Ch. 4, Ch. 15)
Time signature	A notational convention specifying the number of beats per bar (numerator) and the note value that counts as one beat (denominator). (Ch. 1)
Timeline	The main DAW view displaying tracks, clips, and markers along a horizontal time axis; the primary workspace for arranging, editing, and auditioning material. (Ch. 3)
Tokenisation	The process of converting a continuous or structured input (a chord sequence, an audio stream) into a sequence of discrete symbols drawn from a finite vocabulary. (Ch. 2)
Tonic	The first and most stable scale degree of a key; the pitch class to which all other harmonic motion ultimately resolves. (Ch. 6)
Track stack	MODULO's hierarchical grouping mechanism in which child tracks nest under a parent and can be shown or hidden by toggling the parent's collapse state, reducing visual clutter in complex sessions. (Ch. 12)
Transformer	A neural network architecture built from self-attention layers and position-wise feed-forward networks; the decoder-only transformer variant is used for autoregressive chord generation. (Ch. 2)

Continued on next page

Term	Definition
Transient	A brief high-amplitude onset in an audio signal (a drum hit, a plucked-string attack); transient preservation is a key quality criterion for stem separation. (Ch. 8)
UES-SF	User Engagement Scale, Short Form: a 12-item self-report instrument with four subscales (focused attention, perceived usability, aesthetic appeal, reward). (Ch. 6 and all user-study chapters)
Voice leading	The practice of moving individual voices smoothly from chord to chord, following principles such as stepwise motion, contrary motion, and avoidance of parallel fifths and octaves; MODULO's harmony engine encodes classical voice-leading rules. (Ch. 6)
Voicing	A specific arrangement of chord tones in musical space, specifying which octave each pitch occupies; two chords with the same root, quality, and inversion can differ in voicing if individual notes are displaced by octaves. (Ch. 5)
VST3	Steinberg's cross-platform plugin standard (Virtual Studio Technology version 3) for audio effects and virtual instruments. (Ch. 3)
Within-subjects design	An experimental design in which every participant experiences all conditions, allowing each participant to serve as their own control; yields higher statistical power than between-subjects designs for the same N. (Ch. 4)

CHAPTER C

Study Materials and Protocols

This appendix documents study-level details that are common across MODULO's six user studies but were not repeated in full inside each individual study chapter. It covers the shared participant pool and recruitment process, the standardised session logistics, the three standardised survey instruments NASA-TLX, UES-SF, and SUS as they were actually administered, and the study-specific protocol elements that deviated from the shared template. Where a detail is already fully specified in the relevant study chapter, such as stimulus construction, dependent-variable definitions, or pre-registered hypotheses, this appendix does not repeat it; it cross-references instead.

C.1 PARTICIPANT POOL AND RECRUITMENT

The six user studies reported in this thesis, covering chord generation, harmony generation, audio-to-MIDI transcription, stem separation, composition pipeline, and the integrated evaluation, collectively enrolled participants drawn from *both a university setting and the professional music-production industry*. Recruitment used two complementary channels in every study:

- **University channel.** Posts on departmental mailing lists (music, computer science, engineering), bulletin-board postings in the music building and practice-room corridors, and direct outreach to student ensembles and recording-studio practicums. This channel produced undergraduate and graduate musicians, student producers, and stu-

dent researchers with varied formal training.

- **Industry channel.** Direct outreach through personal and professional networks, posts in Slack and Discord communities oriented around music production, and introductions through working producers, engineers, and studio owners. This channel produced working session musicians, independent and signed artists, studio engineers, film and game composers, and product developers at music-technology companies.

Across studies, *skill levels were intentionally heterogeneous*: every study's sample spanned at least three distinct experience tiers, roughly beginner/hobbyist, intermediate/serious student, and advanced/professional, defined by the self-reported intake measures in Section C.3. Each study chapter reports its own sample's skill distribution; the thesis never claimed or used a pure-student or pure-professional sample.

All participants consented in writing before any session began, in accordance with the protocol approved for this work. No audio or video recording of the participants themselves was retained; only their interaction logs, survey responses, and written free-text comments were stored. Participation was voluntary; participants could withdraw at any time without penalty.

C.2 SESSION LOGISTICS

The six studies used a mixture of virtual and in-person sessions:

- **Most sessions ran virtually over Zoom.** The participant installed the MODULO beta or the relevant baseline tool on their own machine, shared their screen with the experimenter, and completed the tasks while the experimenter observed silently except for the scripted prompts. Interaction logs were uploaded by the participant at the end of the session. The virtual format was used in the great majority of chord-

generation, harmony, audio-to-MIDI, stem-separation, and composition-pipeline sessions; it accommodated the geographically distributed industry sample and the variable schedules of the student sample.

- **A small minority of sessions ran in person** in a quiet room on closed-back studio headphones connected to an experimenter-provided laptop. In-person sessions were used when the participant preferred them, when the study protocol required a controlled listening environment (primarily the integrated evaluation and a subset of the stem-separation sessions), or when the participant did not have a suitable personal machine.

The two formats produced equivalent telemetry (the MODULO client recorded identical interaction events in both modes) and the two formats contributed interchangeably to the reported per-condition aggregates. Participants were not stratified by session format because pilot comparison showed no systematic difference between the formats on the primary dependent variables of any study.

Session durations varied substantially across studies, reflecting the different scope of each task:

- **Chord generation**: about 45 minutes per session, spanning two conditions, two melodies, intake, three survey batteries, and a brief exit interview.
- **Harmony generation**: about 60 minutes per session, spanning two conditions, three melodies per condition, intake, three survey batteries, and an exit interview.
- **Audio-to-MIDI transcription**: about 75 minutes per session, spanning three conditions, three clips per condition, intake, three survey batteries, and an exit interview; this was the longest among the component studies because of the three-way condition structure and the per-clip evaluation overhead.

- **Stem separation**: about 50 minutes per session, spanning two conditions, three tracks per condition with two separation modes, intake, three survey batteries, a forced-choice battery, and an exit interview.
- **Composition pipeline**: about 60–75 minutes per session, spanning three conditions, two creative briefs per condition, intake, three survey batteries, a forced-choice battery, and an exit interview; the spread reflects the variable time participants spent on the out-of-DAW Udio and Suno round-trips.
- **Integrated evaluation**: about 120 minutes per session, the longest format of any study: the participant delivered an end-to-end ninety-second composition using every MODULO feature in free order, followed by a deep debrief interview.

Each study's chapter contains the exact per-session time distribution for that study; the figures above are the scheduled durations used in participant recruitment.

C.3 INTAKE QUESTIONNAIRE

Every study used the same short intake questionnaire, administered once at the beginning of the session:

(i) **Primary instrument**: free text, e.g., piano, guitar, voice, drums, or none.

(ii) **Years of active music-making**: integer.

(iii) **Self-rated music-theory knowledge**: five-point Likert from 1 (no formal knowledge) to 5 (advanced; fluent in Roman-numeral analysis and voice leading).

(iv) **Self-rated DAW proficiency**: five-point Likert from 1 (never used a DAW) to 5 (professional-level).

(v) **Prior experience with AI music tools**: binary yes/no with optional free-text elaboration.

(vi) **Recruitment channel**: university, as undergraduate, graduate, or staff; versus industry, as freelancer, employed producer/engineer, or signed/professional artist.

Responses to items (iii) and (iv) were used to stratify the sample into the "beginner," "intermediate," and "advanced" tiers referenced in each study chapter. The specific cut-points used for each stratification are reported alongside that study's results.

C.4 SHARED SURVEY INSTRUMENTS

Three standardised instruments were used across multiple studies. They are reproduced here once, in the exact form in which they were administered, so that study chapters need not repeat the item wording.

C.4.1 NASA Task Load Index (modified five-point form)

The modified five-point NASA-TLX [43] was administered once per condition in every study except the integrated evaluation. All six subscales were retained. The five-point form was adopted, rather than the original twenty-one-point form, to reduce response burden in the short post-condition interval and to improve discriminability at the short task durations typical of these studies.

Table C.1: Modified NASA-TLX subscales and anchor wording as administered.

Subscale	**Item and anchors**
Mental Demand	"How mentally demanding was the task?" 1 = Very Low (almost no mental effort); 5 = Very High (intense and sustained mental effort).
Physical Demand	"How physically demanding was the task?" 1 = Very Low; 5 = Very High.

Subscale	Item and anchors
Temporal Demand	"How hurried or rushed was the pace of the task?" 1 = Very Low (leisurely); 5 = Very High (frantic).
Performance	"How successful were you in accomplishing what you were asked to do?" 1 = Perfect; 5 = Failure. *(Reverse-scored in aggregation.)*
Effort	"How hard did you have to work to accomplish your level of performance?" 1 = Very Low; 5 = Very High.
Frustration	"How insecure, discouraged, irritated, stressed, and annoyed were you?" 1 = Very Low (relaxed); 5 = Very High (extremely frustrated).

The Performance subscale uses the original reversed anchoring (1 = best). In analysis it was reverse-coded so that a higher number uniformly indicates higher workload across all six subscales, consistent with the aggregate composite reported in every study chapter.

C.4.2 User Engagement Scale: Short Form

The UES-SF [76] has twelve items organised into four subscales of three items each, rated on a five-point Likert from 1 (Strongly Disagree) to 5 (Strongly Agree). The items below follow the validated instrument. The UES-SF was administered once per condition in the chord, harmony, audio-to-MIDI, stem-separation, and composition studies, and once per session in the integrated evaluation.

Focused Attention (FA).

FA-1. I lost myself in this experience.

FA-2. The time I spent using the tool just slipped away.

FA-3. I was absorbed in this experience.

Perceived Usability (PU). *(reverse-coded)*

PU-1. I felt frustrated while using the tool.

PU-2. I found the tool confusing to use.

PU-3. Using the tool was taxing.

Aesthetic Appeal (AE).

AE-1. The tool was attractive.

AE-2. The tool was aesthetically appealing.

AE-3. The tool appealed to my visual senses.

Reward (RW).

RW-1. Using the tool was worthwhile.

RW-2. My experience was rewarding.

RW-3. I felt interested in this experience.

C.4.3 System Usability Scale

The standard ten-item SUS [20] was administered once per session in the integrated evaluation and as an optional post-treatment instrument in the chord, harmony, and composition studies. Items alternate between positive and negative wording on a five-point Likert from 1 (Strongly Disagree) to 5 (Strongly Agree):

SUS-1. I think that I would like to use this system frequently.

SUS-2. I found the system unnecessarily complex.

SUS-3. I thought the system was easy to use.

SUS-4. I think that I would need the support of a technical person to be able to use this system.

SUS-5. I found the various functions in this system were well integrated.

SUS-6. I thought there was too much inconsistency in this system.

SUS-7. I would imagine that most people would learn to use this system very quickly.

SUS-8. I found the system very cumbersome to use.

SUS-9. I felt very confident using the system.

SUS-10. I needed to learn a lot of things before I could get going with this system.

Scoring follows the standard procedure [20]: odd-numbered items contribute the scale position minus one, even-numbered items contribute five minus the scale position, and the sum is multiplied by 2.5 to produce a composite in $[0, 100]$. The 68-point benchmark reported in Chapter 13 and Chapter 14 is the commonly cited acceptable-usability threshold for SUS.

C.5 PER-STUDY PROTOCOL NOTES

The six studies share the consent, intake, counterbalancing, and post-condition-survey structure above. This section documents only the protocol elements unique to each study, i.e. those that were *not* fully specified in the study chapter.

C.5.1 Chord generation

The chord-generation study used a two-condition within-subjects design with two composed melodies, counterbalanced into four ordering permutations. The melodies are described in Section C.7. Participants pressed a prominent "Mark Satisfied" button at the first moment they were content with the accompaniment; the timestamp T_{sat} recorded by this press is distinct from the final "Done" press at T_{end}, and is the dependent variable used for the "time-to-satisfaction" analysis in Chapter 4.

C.5.2 Harmony generation

The harmony study used a two-condition within-subjects design with *three* melodies per condition (rather than two). Melody ordering inside each condition was counterbalanced independently of the condition order. The three melodies were chosen to cover three different tonal centres and three different phrase structures; the concrete melody specifications are in Chapter 6. The satisfaction-marking protocol was identical to the chord study.

C.5.3 Audio-to-MIDI transcription

The audio-to-MIDI study is the only component study with a *three-condition* within-subjects structure: manual transcription in a standard DAW, an external Web-hosted transcription tool, and MODULO's built-in A2M engine. Ordering was counterbalanced into six permutations, assigned round-robin. Each participant transcribed three short clips per condition; clips were drawn from a pool of twelve, and clip assignment was counterbalanced separately from condition order. Satisfaction marking was replaced by an explicit "Submit Transcription" confirmation, since the transcription task has a clearer completion criterion than chord composition does.

C.5.4 Stem separation

The stem separation study used a two-condition within-subjects design with three tracks and two separation modes, namely full six-stem separation and vocals/instrumentals separation, yielding six clip-level tasks per condition. Track order was counterbalanced; mode order within each track was always full-stems first, vocals/instrumentals second. After both conditions the participant completed a forced-choice survey comparing the built-in and external tools on six specific aspects: overall quality, specific stems such as vocals and drums, workflow speed, ease of comparison, and preference for real work.

C.5.5 Composition pipeline

The composition-pipeline study used a three-condition within-subjects design comparing MODULO, the Udio free tier, and the Suno free tier, with two creative briefs per condition. The free-tier choice for Udio and Suno was deliberate: the free tier is the most common entry point for new users and therefore the most informative baseline for a musician-facing comparison. The study protocol allowed the out-of-DAW conditions to use the browser, external audio rendering, and any tools the participant wished; only the time spent outside MODULO was captured separately as the "external round-trip" variable. Forced-choice items covered preference on six dimensions: overall suitability for real-work composition, speed, structural control, timbre/audio polish, editability, and the perceived fit of each tool to the participant's existing workflow.

C.5.6 Integrated evaluation

The integrated evaluation used a single-condition design with no comparison: the participant built an original ninety-second composition end-to-end using whichever MODULO features served the piece they were trying to make. No feature was required; no feature was forbidden. The session began with a brief feature walkthrough under seven minutes

and an eight-minute warm-up on the sandbox project before the free composition began. After the composition, the participant completed NASA-TLX, UES-SF, and SUS once for the session as a whole, and then engaged in a semi-structured debrief interview covering discovery, flow, feature overlap, and workflow fit. Interview responses were analysed using reflexive thematic analysis [17].

C.6 CUSTOM LIKERT BATTERIES

Each component study also administered a short study-specific Likert battery on a five-point scale from 1 (Strongly Disagree) to 5 (Strongly Agree). These are the actual items; the analysis-ready scoring rules are in each study chapter.

Chord study custom items.

C-1. The tool helped me explore multiple harmonic directions.

C-2. The tool helped me get to a satisfactory result quickly.

C-3. The tool helped me refine my choices without having to settle too early.

C-4. I felt in control of the creative process while using the tool.

C-5. The tool would be useful for building chordal accompaniments.

C-6. I would use this tool in my real music-making practice.

C-7. I am satisfied with the quality of the final chord progression I produced.

Harmony study custom items.

H-1. I could explore different voicings easily.

H-2. I could create distinct harmony lines quickly.

H-3. I achieved good voice leading in my final harmony.

H-4. I felt in control of the harmonic outcome.

H-5. I would use this harmony capability in real music-making.

Audio-to-MIDI custom items.

A-1. The transcription captured the rhythm I heard in the audio.

A-2. The transcription captured the pitches I heard in the audio.

A-3. I could correct the errors I found quickly.

A-4. I would use this capability for real transcription work.

Stem-separation custom items.

S-1. The separated stems sound clean.

S-2. The separated stems are labelled and organised usefully in my project.

S-3. I could compare results across tools easily.

S-4. I would use this capability for real production work.

Composition-pipeline custom items.

P-1. The tool got me from brief to a usable arrangement quickly.

P-2. The structure of the generated output matched what my brief asked for.

P-3. I felt in control of the structural outcome.

P-4. I would use this capability in real-work composition.

C.7 CHORD STUDY STIMULUS MELODIES

Two stimulus melodies were composed specifically for the chord-generation study; they were not reused in any other study. Both are eight bars long and have similar melodic density; they differ in key, mode, tempo, and rhythmic profile so that the four ordering permutations do not over-sample one tonal space.

Melody A. C major at 80 BPM in $\frac{4}{4}$. The eight-bar phrase is almost entirely stepwise, with occasional thirds and one gentle syncopation at the midpoint. The tonic is reinforced by repeated returns to the C pitch class, giving a stable harmonic canvas suitable for straight diatonic harmonisation as well as for chromatic or modal-mixture experiments.

Melody B. F minor at 70 BPM in $\frac{4}{4}$. The eight-bar phrase mixes stepwise motion with fourth and fifth leaps, introducing more harmonic ambiguity than Melody A. The minor mode admits a wider range of plausible harmonisations including natural-minor, harmonic-minor, and borrowed-dominant colourations. The slower tempo leaves more time for real-time audition of chord choices during the task.

Both melodies were vetted by two independent musicians, one with formal composition training and one self-taught, to check that neither was trivially simple nor prohibitively complex for the target participant population. The melodies were pre-loaded on the timeline before the participant arrived; participants could not modify the melody content itself.

Stimulus artefacts for the other five studies, including the three harmony melodies, the twelve audio-to-MIDI clips, the three stem-separation tracks, and the six composition-pipeline briefs, are described in the corresponding study chapters and are not repeated here.

Bibliography

[1] Ableton AG. Ableton live: Music production software. https://www.ableton.com, 2025. Version 12.

[2] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caber, Marco Tagliasacchi, Matthew Padilla, Dominik Schwarz, Ning Ye, Neil Petruzzellis, and Neil Frank. MusicLM: Generating music from text. In *arXiv preprint arXiv:2301.11325*, 2023.

[3] AIVA Technologies. AIVA: AI music composition for creative projects. https://www.aiva.ai, 2025. Accessed April 2026. AI-driven composition for film, game, and commercial scoring.

[4] Edward Aldwell, Carl Schachter, and Allen Cadwallader. *Harmony and Voice Leading*. Cengage Learning, 5th edition, 2019. ISBN 978-1-337-56057-3.

[5] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for human-AI interaction. *Proc. CHI Conference on Human Factors in Computing Systems*, 2019. doi: 10.1145/3290605.3300233.

[6] Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, 2024. Technical announcement, accessed 2026-04-14.

[7] Apple Inc. macOS human interface guidelines. https://developer.apple.

com/design/human-interface-guidelines/, 2025. Accessed April 2026.

[8] Apple Inc. Audio unit hosting guide. https://developer.apple.com/documentation/audiotoolbox/audio_unit_hosting_guide, 2025. Accessed April 2026.

[9] Apple Inc. Logic pro: Professional music production. https://www.apple.com/logic-pro/, 2025. Version 11.x.

[10] Ardour Community. Ardour: The digital audio workstation. https://ardour.org, 2025. Version 8.x. Open-source under GPL v2.

[11] BandLab Technologies. BandLab: Free online music studio. https://www.bandlab.com, 2025. Accessed April 2026. Web-based multi-track DAW with AI-assisted features.

[12] Aaron Bangor, Philip T. Kortum, and James T. Miller. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4 (3):114–123, 2009.

[13] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1):289–300, 1995.

[14] Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 781–785, 2022. doi: 10.1109/ICASSP43922.2022.9746549.

[15] BOOM Library. BOOM Library: Professional sound effects libraries. https://www.boomlibrary.com, 2025. Accessed April 2026.

[16] Boombox.io. Boombox: AI chord progressions and music tools. https://boombox.io, 2025. Accessed April 2026.

[17] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006. doi: 10.1191/1478088706qp063oa.

[18] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation – a survey. *Neural Computing and Applications*, 32: 981–993, 2020. doi: 10.1007/s00521-018-3943-5.

[19] Robert Bristow-Johnson. Audio EQ cookbook. Technical report, 2005. Widely cited formulary for biquad filter coefficient computation. Hosted at https://www.w3.org/2011/audio/audio-eq-cookbook.html.

[20] John Brooke. SUS: A quick and dirty usability scale. In Patrick W. Jordan, Bruce Thomas, Bernard A. Weerdmeester, and Ian L. McClelland, editors, *Usability Evaluation in Industry*, pages 189–194. Taylor and Francis, 1996.

[21] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human–Computer Interaction*. Lawrence Erlbaum Associates, 1983. ISBN 978-0-89859-243-4.

[22] Luca Casini, Laura Cros Vila, David Dalmazzo, Anna-Kaisa Kaila, and Bob L.T. Sturm. Data-driven analysis of text-conditioned AI-generated music: A case study with Suno and Udio. *arXiv preprint arXiv:2509.11824*, 2025. Submitted to TISMIR.

[23] Yanxu Chen, Linshu Huang, and Tian Gou. Applications and advances of artificial intelligence in music generation: A review. *arXiv preprint arXiv:2409.03715*, 2024.

[24] Wan Chong Choi and Chi In Chang. Suno AI: Opportunities and challenges of AI-generated music and lyrics in secondary music education. 2025. Preprint.

[25] Yewon Choi et al. Understanding the potentials and limitations of prompt-based music generative AI from the perspective of electronic music producers. In *Proc. CHI Conference on Human Factors in Computing Systems*, 2025.

[26] ChordSeqAI. ChordSeqAI: AI-powered chord progression generator. https://chordseqai.com, 2025. Accessed April 2026.

[27] Jacob Cohen. Statistical power analysis for the behavioral sciences. 1988.

[28] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

[29] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music source separation in the waveform domain. In *arXiv preprint arXiv:1911.13254*, 2019.

[30] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. In *Transactions on Machine Learning Research*, 2023.

[31] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. In *arXiv preprint arXiv:2005.00341*, 2020.

[32] Kemal Ebcioğlu. An expert system for harmonizing chorales in the style of J.S. Bach. *Computer Music Journal*, 12(3):43–51, 1988. doi: 10.2307/3679959.

[33] ElevenLabs. ElevenLabs music, audio isolation, and sound effects API. 2024. Technical documentation, accessed 2026-04-14.

[34] ElevenLabs. ElevenLabs Music API: Text-to-music generation. https://elevenlabs.io/docs/api-reference/music, 2025. Version 1. Accessed April 2026.

[35] ElevenLabs. ElevenLabs Sound Effects API. https://elevenlabs.io/docs/api-reference/sound-effects, 2025. Version 1. Accessed April 2026.

[36] ElevenLabs. ElevenLabs Stem Separation API. https://elevenlabs.io/docs/api-reference/stem-separation, 2025. Version 1. Accessed April 2026.

[37] K. Anders Ericsson and Herbert A. Simon. *Protocol Analysis: Verbal Reports as Data*. MIT Press, revised edition, 1993. ISBN 978-0-262-55023-2.

[38] Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open. In *Proc. International Conference on Machine Learning (ICML)*, 2024.

[39] Jonas Frich, Lindsay MacDonald Vermeulen, Christian Remy, Michael Mose Biskjaer, and Peter Dalsgaard. Mapping the landscape of creativity support tools in HCI. *Proc. CHI Conference on Human Factors in Computing Systems*, 2019. doi: 10.1145/3290605.3300619.

[40] Johann Joseph Fux. *Gradus ad Parnassum*. Vienna, 1725. Translated by Alfred Mann as *The Study of Counterpoint*, W.W. Norton, 1965.

[41] Josh Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse Engel. Multitrack music transcription with a new general-purpose Transformer architecture. In *International Conference on Learning Representations (ICLR)*, 2022.

[42] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-audio generation using instruction-tuned LLM and latent diffusion model. In *Proc. ACM Multimedia*, 2023.

[43] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (task load index): Results of empirical and theoretical research. *Advances in Psychology*, 52: 139–183, 1988.

[44] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factored piano music modeling and generation with the MAESTRO dataset. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[45] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factored piano music modeling and generation with the MAESTRO dataset. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[46] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: A fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50):2154, 2020. doi: 10.21105/joss.02154.

[47] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys*, 50(5):1–30, 2017. doi: 10.1145/3108242.

[48] Hit'n'Mix Ltd. RipX DAW: AI-powered audio editing. https://www.hitnmix.com/ripx-daw/, 2025. Version 7.

[49] Hooktheory. Hookpad: Music composition and chord progression tool. https://www.hooktheory.com/hookpad, 2025. Accessed April 2026. Browser-based chord-palette composition tool with theory-aware suggestions.

[50] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *Proc. International Conference on Machine Learning (ICML)*, 2023.

[51] David Miles Huber and Robert E. Runstein. *Modern Recording Techniques*. Focal Press, 9th edition, 2018. ISBN 978-1-138-95434-3.

[52] David Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64, 2001. doi: 10.1525/mp.2001.19.1.1.

[53] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct manipulation interfaces. *Human–Computer Interaction*, 1(4):311–338, 1985. doi: 10.1207/s15327051hci0104_2.

[54] International Telecommunication Union. Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. ITU-T Recommendation P.862, 2001.

[55] Roey Izhaki. *Mixing Audio: Concepts, Practices, and Tools*. Focal Press, 3rd edition, 2018. ISBN 978-1-138-85937-1.

[56] JUCE. JUCE: Cross-platform C++ application framework for audio software. https://juce.com, 2024. Version 8.0.12. Originally by Julian Storer; maintained by PACE Anti-Piracy, Inc.

[57] Purnima Kamath, Fabio Morreale, Priambudi Lintang Bagaskara, Yize Wei, and Suranga Nanayakkara. Sound designer–generative AI interactions: Towards designing creative support tools for professional sound designers. In *Proc. CHI Conference on Human Factors in Computing Systems*, 2024. doi: 10.1145/3613904.3642040.

[58] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Proc. Interspeech*, pages 2350–2354, 2019.

[59] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. AudioCaps: Generating captions for audios in the wild. In *Proc. North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 119–132, 2019.

[60] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. CREPE: A convolutional representation for pitch estimation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165, 2018. doi: 10.1109/ICASSP.2018.8461329.

[61] Minseok Kim, Woosung Choi, Jaehwa Chung, Daewon Lee, and Soonyoung Jung. KUIELab-MDX-Net: A two-stream neural network for music demixing. In *Music Demixing Workshop at ISMIR*, 2021.

[62] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. AudioGen: Textually guided audio generation. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.

[63] Szymon J. Król, Ondřej Černý, and Nick Bryan-Kinns. Exploring the needs of practising musicians in co-creative AI through co-design. In *Proc. CHI Conference on Human Factors in Computing Systems*, 2025.

[64] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R. Hershey. SDR – half-baked or well done? In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 626–630, 2019. doi: 10.1109/ICASSP.2019.8683855.

[65] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. In *Proc. International Conference on Machine Learning (ICML)*, 2023.

[66] Haohe Liu, Qiao Tian, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D. Plumbley. AudioLDM 2: Learning holistic audio generation with self-supervised pretraining. In *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[67] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie Jun Cai. Novice-AI music co-creation via AI-steering tools for deep generative models. In *Proc. CHI Conference on Human Factors in Computing Systems*, 2020. doi: 10.1145/3313831.3376739.

[68] Ryan Louie, Jesse Engel, and Cheng-Zhi Anna Huang. Expressive communication: Evaluating developments in generative models and steering interfaces for music creation. In *Proc. International Conference on Intelligent User Interfaces (IUI)*, 2022.

[69] Yi Luo and Jianwei Yu. Music source separation with band-split RNN. In *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, volume 31, pages 1893–1901, 2023. doi: 10.1109/TASLP.2023.3271145.

[70] Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. Cutting music source separation some Slakh: A dataset to study the impact of training data on music source separation. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019.

[71] Mozart AI. Mozart: AI music composition platform. https://mozart.ai, 2025. Accessed April 2026.

[72] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015. ISBN 978-3-319-21944-8.

[73] Arnav Nayar. The ethics and legality of Suno as a human-centered AI. University of Texas at Austin, 2025. Preprint.

[74] Jakob Nielsen. *Usability Engineering*. Academic Press, Boston, MA, 1993.

[75] Donald A. Norman. *The Design of Everyday Things*. Basic Books, revised and expanded edition, 2013. ISBN 978-0-465-05065-9.

[76] Heather L. O'Brien, Paul Cairns, and Mark Hall. A practical approach to measuring user engagement with the refined user engagement scale (UES) and new UES short form. *International Journal of Human-Computer Studies*, 112:28–39, 2018.

[77] Yunyu Ong, Robert Sazdov, and Andrew Johnston. Opening DAWs to interactive music – making an orchestra out of soloists. In *Proc. New Interfaces for Musical Expression (NIME)*, 2024.

[78] Walter Piston and Mark DeVoto. *Harmony*. W.W. Norton, 5th edition, 1987. ISBN 978-0-393-95480-7.

[79] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[80] Antony W. Rix, John G. Beerends, Michael P. Hollier, and Andries P. Hekstra. Perceptual evaluation of speech quality (PESQ)—a new method for speech quality assessment of telephone networks and codecs. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 749–752, 2001. doi: 10.1109/ICASSP.2001.941023.

[81] Curtis Roads. *The Computer Music Tutorial*. MIT Press, 1996. ISBN 978-0-262-68082-0.

[82] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[83] Simon Rouard, Francisco Massa, and Alexandre Défossez. Hybrid transformers for music source separation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096956.

[84] Alexander Scarlatos, Yusong Wu, Ian Simon, Adam Roberts, Tim Cooijmans, Natasha Jaques, Cassie Tarakajian, and Anna Huang. ReaLJam: Real-time human-AI music jamming with reinforcement learning-tuned transformers. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA)*, 2025.

[85] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[86] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.

[87] Sesh FM. Sesh FM AI stem splitter. Web service, https://sesh.fm, 2025. Accessed 2025.

[88] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting

Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in hugging face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[89] Ben Shneiderman. Creativity support tools: Accelerating discovery and innovation. volume 50, pages 20–32, 2007. doi: 10.1145/1323688.1323689.

[90] Jason Brent Smith and Jason Freeman. Adaptation and perceived creative autonomy in gesture-controlled interactive music. In *Proc. New Interfaces for Musical Expression (NIME)*, 2025.

[91] Julius O. Smith. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, 2007. https://ccrma.stanford.edu/~jos/filters/.

[92] Soundsnap. Soundsnap: Sound effects library. https://www.soundsnap.com, 2025. Accessed April 2026.

[93] Splice. Splice: Royalty-free sample marketplace. https://splice.com, 2025. Accessed April 2026.

[94] Steinberg Media Technologies. VST 3 SDK: Virtual studio technology software development kit. https://steinbergmedia.github.io/vst3_dev_portal/, 2025. Version 3.7.

[95] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix—a reference implementation for music source separation. *Journal of Open Source Software*, 4(41):1667, 2019. doi: 10.21105/joss.01667.

[96] Minhyang (Mia) Suh, Emily Youngblom, Michael Terry, and Carrie J. Cai. AI as social glue: Uncovering the roles of deep generative AI during social music composition. In *Proc. CHI Conference on Human Factors in Computing Systems*, 2021. doi: 10.1145/3411764.3445219.

[97] Suno Inc. Suno: AI-generated music from a text prompt. https://suno.com, 2024. Consumer prompt-and-receive tier; accessed April 2026.

[98] Suno Inc. Suno studio: AI music creation platform. https://suno.com, 2025. Accessed April 2026.

[99] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time-frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011. doi: 10.1109/TASL.2011.2114881.

[100] John Thickstun, Zaid Harchaoui, and Sham Kakade Dean P. Bhatt. Learning features of music from scratch. In *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[101] Chris Thorpe, Purnima Kamath, and Suranga Nanayakkara. Supporting sound design with text-to-audio generation: An exploration of image-schematic prompting. In *Proc. International Conference on Intelligent User Interfaces (IUI)*, 2024.

[102] Tracktion Software Corporation. Tracktion engine: Open-source DAW engine. https://github.com/Tracktion/tracktion_engine, 2025. Version 3.2.0. Dual GPL v3/Commercial license. Open-sourced November 2018.

[103] Dmitri Tymoczko. The geometry of musical chords. *Science*, 313(5783):72–74, 2006. doi: 10.1126/science.1126287.

[104] Udio Inc. Udio: AI-generated music from a text prompt. https://udio.com, 2024. Consumer prompt-and-receive tier; accessed April 2026.

[105] Udio Inc. Udio: AI music generation platform. https://udio.com, 2025. Accessed April 2026.

[106] Universitat Pompeu Fabra. Freesound: Collaborative database of creative-commons licensed sound. https://freesound.org, 2025. Accessed April 2026.

[107] Soumya Sai Vanka, Jean-Baptiste Rolland, and György Fazekas. Demonstrating Diff-MSTC: A controllable and context-aware AI system for multitrack mixing in DAW Cubase. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA)*, 2025. doi: 10.1145/3706599.3721167.

[108] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[109] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006. doi: 10.1109/TSA.2005.858005.

[110] W3C. Web audio API. https://www.w3.org/TR/webaudio/, 2021. W3C Recommendation, June 2021.

[111] Yusong Wu, Tim Cooijmans, Kyle Kastner, Adam Roberts, Ian Simon, Alexander Scarlatos, Chris Donahue, Cassie Tarakajian, Shayegan Omidshafiei, Aaron Courville, Pablo Samuel Castro, Natasha Jaques, and Cheng-Zhi Anna Huang. Adaptive accompaniment with ReaLchords. In *Proc. International Conference on Machine Learning (ICML)*, 2024.

[112] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. Re-examining whether, why, and how human-AI interaction is uniquely difficult to design. *Proc. CHI Conference on Human Factors in Computing Systems*, 2020. doi: 10.1145/3313831.3376301.

[113] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan,

and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proc. International Conference on Learning Representations (ICLR)*, 2023.

[114] Dingyao Yu, Kaitao Song, Peiling Lu, Tianyu He, Xu Tan, Wei Ye, Shikun Zhang, and Jiang Bian. MusicAgent: An AI agent for music understanding and generation with large language models. In *arXiv preprint arXiv:2310.11954*, 2023.

[115] Yifei Zhang, Yukara Ikemiya, Woosung Choi, Naoki Murata, Marco A. Martínez-Ramírez, Wei-Hsiang Liao, Yuki Mitsufuji, and Simon Dixon. A survey on audio diffusion models: Text to speech synthesis and enhancement in generative AI. *arXiv preprint arXiv:2403.15091*, 2024.

[116] Yixiao Zhang, Yukara Ma, Akira Maezawa, Shinji Watanabe, Emmanouil Benetos, and Simon Dixon. Loop Copilot: Conducting AI ensembles for music generation and iterative editing. In *arXiv preprint arXiv:2310.12404*, 2023.

[117] Alon Ziv, Itai Gat, Gael Le Lan, Tal Remez, Felix Kreuk, Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. Masked audio generation using a single non-autoregressive transformer. In *Proc. International Conference on Learning Representations (ICLR)*, 2024.

[118] Udo Zölzer. *DAFX: Digital Audio Effects*. John Wiley & Sons, 2nd edition, 2011. ISBN 978-0-470-66599-2.