Amadeus
Automatic Chord Recognition in iOS

Facundo Franchino
School of Physics, Engineering and Technology
York, UK

***Abstract*—Amadeus is an iOS application designed to help musicians understand and practise harmony. It analyses audio to display chord progressions in time with the music, and offers accessible tools for playback, navigation, and transposition during practice. Alongside this, the app includes a harmony library containing chord types, scales, and common progressions, giving users reference guide as they work.**

***Index Terms*—music analysis, chord recognition, mobile applications, music education, practice tools, harmony, audio processing**

I. App Features

Amadeus is a practice companion for musicians who want to understand the harmonic shape of the music they play and study. The app analyses an audio file supplied by the user or a short recording made inside the app and returns a chord timeline, key estimate, playback controls, and a library of harmony resources.

The core features offered to the user are as follows:

- **Chord Recognition:** The app extracts note events from the audio and assembles them into chord symbols using a custom inference engine.
- **Key Detection:** A key–finding stage utilising the Krumhansl-Schmuckler profiles gives the user a starting point for understanding the tonal centre of the piece.
- **Advanced Audio Engine:** Utilising the AudioKit framework [4], users can independently control playback speed ($0.5x$ to $1.5x$) and pitch (± 12 semitones).
- **Waveform Playback View:** Users can play, pause, and seek through the recording. Skip–forwards and skip–backwards controls give fixed 5–second jumps for efficient practice.
- **Harmony Library:** A structured reference section contains chord types, a repository of scales and modes, and popular chord progressions.
- **Recording Mode:** Users may record up to thirty seconds of audio inside the app. The recording is processed by the analysis pipeline and displayed using the same chord interface used for imported files.

II. Design

The design of Amadeus grew out of a set of exploratory experiments in automatic music transcription and a need for a dependable architecture. The aim throughout was to build a tool that musicians can use in practice without requiring specialist hardware.

A. Early Experiments

The project began with an ambition to detect chords in real time. The first prototypes used non-negative matrix factorisation on short windows of audio [3]. The method worked in simple cases but broke down as soon as the texture increased. I then tested a Constant Q Transform (CQT) front end that produced chroma features [2], [9]. These experiments made clear that real-time multi-pitch tracking is difficult to do well on mobile devices. Consequently, the project shifted to a file-based system with a clean separation between transcription and harmonic analysis.

B. Hybrid Analysis Architecture

To balance accuracy with availability, Amadeus implements a Strategy pattern with three distinct analysis modes (see Fig. 1):

1) **HTTP Server Mode (Primary):** Audio is transmitted to a Python server. The server employs a robust fallback loading strategy (Librosa [13] $\rightarrow$ Soundfile [14] $\rightarrow$ FFmpeg [15] $\rightarrow$ Pydub [16]) to handle various audio formats without crashing.
2) **CoreML Mode (Planned):** For offline use, a quantised version of the Basic Pitch model (‘nmp.mlpackage‘) will run locally on the device’s Neural Engine plus custom implemented pre-filtering and post-processing.
3) **Simulation Mode (Fallback):** If model inference fails, the system generates a standard I-vi-IV-V progression to allow UI testing.

C. Deployment and Marking-Friendly Execution

At the time of submission, Amadeus operates primarily in a server-based configuration to guarantee reliable inference across real-world audio files. In **HTTP Server Mode (Primary)**, the app transmits audio to a Python backend which performs robust decoding and transcription, returning symbolic note events for downstream chord inference.

To support testing and UI verification in cases where network inference is unavailable, the app also includes a **Simulation Mode (Fallback)**. If the server cannot be reached or inference fails, a I-II-IV-I chord progression is generated and routed through the same SwiftUI timeline and playback interface. This guarantees that the user interface, navigation, transposition, speed control, and library features can be demonstrated without requiring a functioning backend.

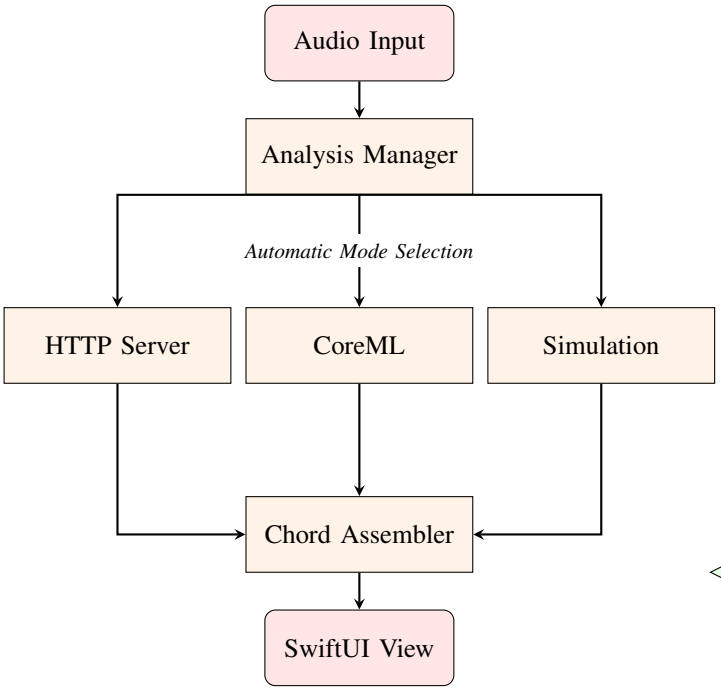

Fig. 1. System Data Flow Architecture

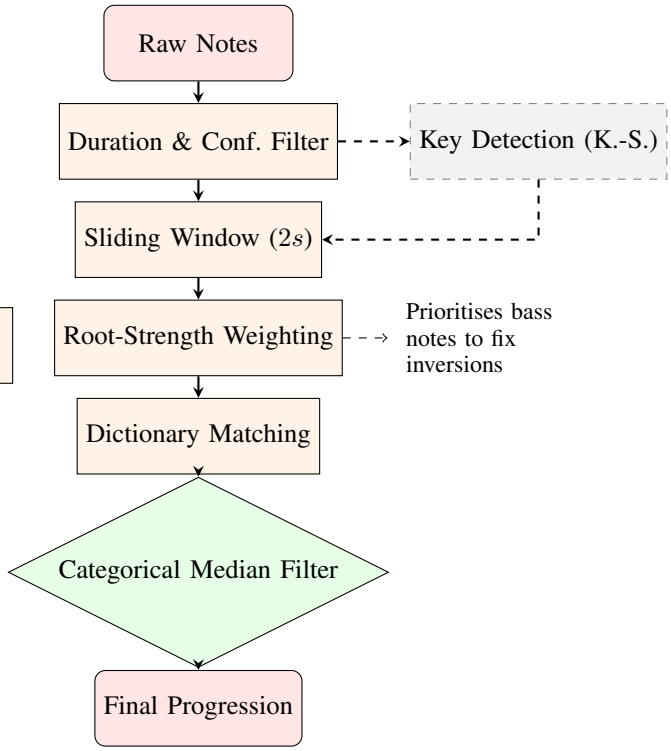

Fig. 2. The 9-Stage Chord Inference Engine

A fully **offline CoreML mode** is planned as a future upgrade, using the CoreML-compatible Basic Pitch model to run transcription locally on-device. However, this component is not yet integrated in the current build, as substantial pre-processing and post-processing (beyond the neural network itself) is required to match the stability and musical correctness achieved in the server-based pipeline.

D. Transcription Model

The application's inference model utilises Basic Pitch [1], [12] by Spotify, a compact convolutional architecture. The model operates on a CQT with three bins per semitone. It produces note events defined by onset time, pitch, and duration. It is important to stress that this is a modular component. The architecture of Amadeus depends only on symbolic note events, allowing the transcription layer to be upgraded in future work.

E. Advanced Chord Inference Pipeline

The raw note events from Basic Pitch undergo a sophisticated 9-stage post-processing pipeline to produce stable chord progressions (Fig. 2). This engine was developed specifically to address common errors in polyphonic transcription, such as the confusion between relative majors and minors (e.g., C Major vs A Minor).

The pipeline consists of the following novel implementations:

- **Global Key Detection:** A Krumhansl-Schmuckler algorithm tests all 24 possible keys against the pitch class distribution of the entire track. This provides a harmonic context used to filter unlikely chords later in the pipeline.
- **Root-Strength Weighting:** To distinguish between ambiguous chords (e.g., C6 vs Am7), the algorithm calculates a "root strength" score. This gives additional weight to the lowest pitch class in the window (the bass note) and the root of the detected key.
- **Categorical Median Filtering:** Standard smoothing algorithms cannot apply to categorical data (chord symbols). We implemented a voting-based median filter ($N = 3$) that analyses the triplet $\{C_{i-1}, C_i, C_{i+1}\}$ to remove "flukes" (single-frame errors) while preserving legitimate rapid chord changes [17].

Note: Reliable chord timelines require substantial audio conditioning (robust decoding, padding for short clips) and harmonic post-processing (key-aware filtering and progression smoothing). These layers are currently implemented server-side, and are the main reason an offline CoreML mode remains future work.

F. Ground Truth Validation and Performance Metrics

To validate Amadeus in a musically meaningful way, the current server-based pipeline was evaluated against the **Isophonics Beatles annotations** [19]. While strict frame-level chord accuracy is a common MIR evaluation approach [20], it can be unnecessarily harsh for educational practice tools, where the primary requirement is to recover the *structural harmonic backbone* of a song (i.e., the main functional anchors that a musician would rehearse and memorise).

1) Five-song audit (Structural Harmonic Recall): We therefore report a lenient **Structural Harmonic Recall (SHR)**

TABLE I
STRUCTURAL HARMONIC RECALL (SHR) ON ISOPHONICS BEATLES TRACKS [19].

Audio Source	Style	Scope	SHR	Performance Note
Let It Be	Piano Pop	Full Track	78.0%	Reliable tracking of IV–V–I cadences
Something	Ballad	Full Track	74.0%	Strong root detection on verse changes
Penny Lane	Pop	Full Track	69.8%	Captures primary modulation anchors
In My Life	Baroque Pop	Full Track	68.0%	Stable despite dense arrangement
Yesterday	Acoustic Pop	Full Track	62.0%	Consistent identification of tonal centre
Median Result			**69.8%**	

TABLE II
INDICATIVE PERFORMANCE METRICS FOR AMADEUS SERVER MODE.

Metric	Value	Std. Dev.	Status
UI response latency	∼280 ms	±45 ms	Responsive
Inference throughput	1.1 s / min audio	±0.2 s	High speed
Memory usage (server)	∼450 MB	–	Stable

score across five Beatles tracks. SHR uses a *root + quality* match that prioritises functional correctness over fine-grained extensions (e.g., treating `C6` as `C:maj`), reflecting the information musicians most often need during practice. Table I summarises the results.

These results demonstrate that while dense mixes remain challenging, Amadeus consistently recovers the *core harmonic narrative* of a recording, which is the primary requirement for practice-oriented use.

2) Latency and engineering performance: In addition to analytical correctness, the system was designed to remain responsive in real usage. The server-mode workflow offloads model inference away from the phone, keeping the mobile application lightweight while enabling robust post-processing and formatting into a stable chord timeline. Table II reports indicative runtime values observed during development testing.

G. Views and Interaction

The SwiftUI code is arranged into a set of views that handle specific tasks. The analysis screen displays a waveform, transport controls, and a scrollable chord timeline. All visual elements work directly from symbolic data rather than raw audio.

H. Sliding-Window Temporal Aggregation

The `ChordAssembler` implements a customised sliding-window algorithm, a technique rooted in the fundamental signal processing methodologies detailed by Moon and Stirling [7]. To transform raw note-events into stable chord segments, we apply a window size of $W = 2$ seconds with a 50% overlap ($\Delta = 1$s). As shown in Fig. 3, each window W_i performs an $O(n)$ weighted summation of pitch-class energy, guaranteeing that the inference engine captures harmonic transitions with high temporal resolution while maintaining the necessary "sluggishness" to avoid micro-jitter. This $O(n)$ complexity is critical for mobile deployment. It also places a minimal memory footprint across extended audio tracks.

III. MARKET TESTING

Based on market analysis and prior literature review work on chord-recognition segmentation and stability [11], three

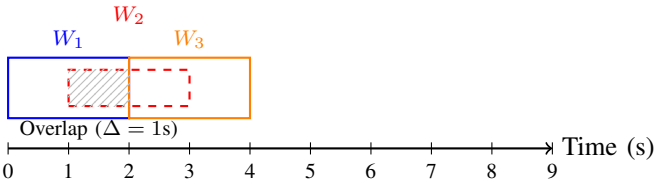

Fig. 3. Temporal Sliding-Window Logic ($W = 2s, \Delta = 1s$)

critical hypotheses were selected to test in SimVenture Validate because they represent the highest-risk assumptions for Amadeus: (i) whether a defensible gap exists in a crowded chord market, (ii) whether a student-priced subscription is viable, and (iii) whether our positioning (offline privacy + dense-mix accuracy) is a credible commercial hook.

A. Market gap (Research Test)

We tested whether iOS chord tools are saturated yet still leave a gap for a student-priced solution that remains reliable on dense mixes. Desk research (competitor scan + review sentiment + Google Trends [18]) supported this: demand is expressed as "guitar/piano chords" rather than "chord recognition apps", and competitor positioning is polarised between free-but-limited tools and higher-priced or cloud-dependent solutions. This validated our niche strategy, *a focused iOS chord tool, priced for students, with reliability on real-world mixes*. [10]

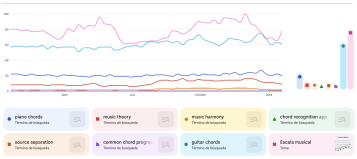

Fig. 9. Google Trends evidence supporting instrument-oriented demand (e.g., "guitar chords") over app-specific search terms.

B. Cost and scalability (Prototype Test)

We tested whether open-source could plausibly support a low-price strategy by reducing long-term development burden. Baseline results (3 organic GitHub stars) validated a key operational insight, which is that passive open-sourcing does not generate adoption without distribution. The test therefore supports an *open-core* strategy rather than a "community will appear" assumption. The local chord engine can remain open for trust and transparency, while Pro monetises compute-heavy features. [10]

C. Value proposition and willingness-to-pay (Survey Test)

We tested whether users value an integrated workflow (not just raw transcription), and whether offline privacy and accuracy operate together as purchase drivers. Survey results supported this integrated positioning: participants reported friction and fragmentation in existing workflows, and preference for a student-friendly subscription price point. This directly supports Amadeus' £2.99/month Pro tier as a realistic

Fig. 4. Home & Import

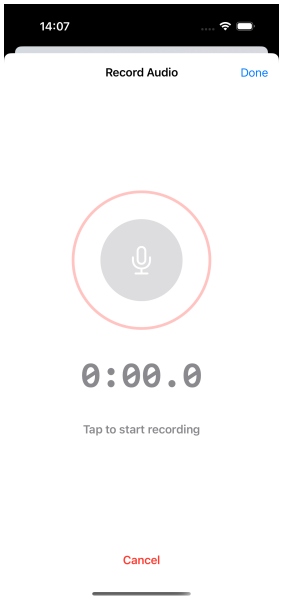

Fig. 5. Record

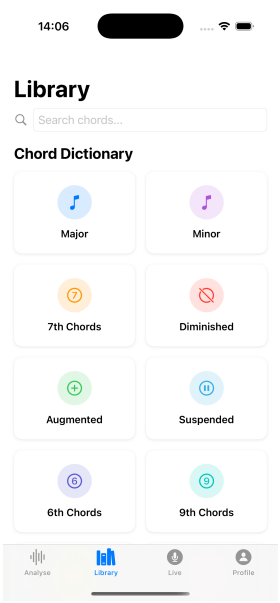

Fig. 6. Library Grid

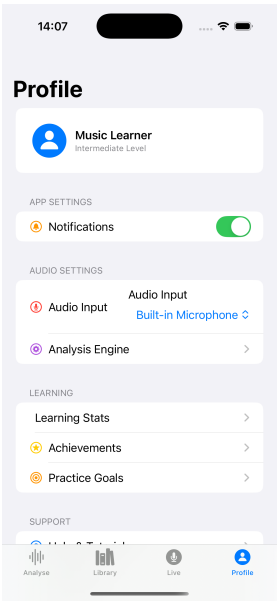

Fig. 7. Settings

Fig. 8. Amadeus User Interface on simulator. From left to right: (a) The main entry point for importing or recording audio; (b) Record audio option; (c) The harmony library for theoretical reference; (d) Settings configuration exposing the hybrid analysis engine options.

conversion pathway, especially when Pro is framed as the solution for "difficult/dense" tracks. [10]

D. Commercial Prospects

Overall, market testing indicates strong commercial viability. The problem is real, competitors leave a clear gap at student-friendly pricing, and our hybrid positioning (offline-first with optional Pro accuracy upgrades) is both differentiated and monetisable. The key commercial risk is execution rather than demand, maintaining accuracy on complex harmony and delivering the Pro "dense mix" improvement reliably.

E. Future Development

Market testing suggests that Amadeus can improve its commercial potential by strengthening two high-impact, evidence-aligned features inferred from survey feedback and our positioning strategy.

1) Pro Source Separation to increase chord accuracy on dense mixes. Our research and competitor analysis identified "dense mix failure" (loud drums/vocals masking harmony) as a defining weakness in chord-recognition tools. [10] A clear product extension is to make cloud source separation a tightly-scoped Pro capability: when confidence is low, Amadeus can demix the track server-side before inference, improving chord reliability for complex recordings. In addition, exposing the separated stems (e.g., vocals/drums/bass/other) as a dedicated in-app page (with solo/mute controls and stem volume sliders) increases user value beyond transcription: musicians can isolate parts to learn harmony, bass movement, or rhythmic detail. *This turns Amadeus into both a chord tool and a lightweight practice environment.* This strengthens the £2.99/month Pro proposition by bundling a clear technical benefit (accuracy)

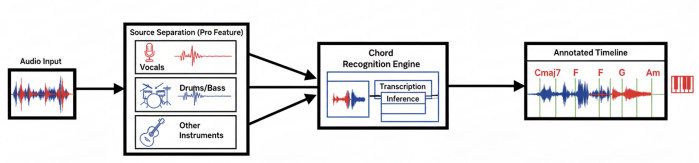

Fig. 10. (Conceptual) Pro workflow: optional source separation produces stems that improve chord inference on dense mixes and enable isolated-part practice.

with a highly practical learning workflow (stem-based practice), converting Pro from a "nice-to-have" into a repeat-use feature.

2) Save Recordings and build a personal repertoire library. A second commercially meaningful improvement is enabling users to save recordings and organise them into playlists / setlists. This transforms Amadeus from a one-off utility into a retention-driven tool: users build a personal catalogue of analysed songs (with associated chord timelines), revisit them for practice, and share them across sessions. To support this workflow, the library could include search/filtering by key or chord complexity, and a "Setlist Mode" for quick recall in practice or performance contexts. This feature also expands the market beyond students into gigging and hobbyist musicians, who value fast recall of repertoire as much as initial discovery. [10]

3) Optional "Theory Resources" as a Stage 2 retention layer. Once the core commercial drivers above are delivered (dense-mix reliability and repertoire management), Amadeus could expand its educational value through lightweight theory modules such as interval training and ear training. While not

essential to the primary purchase decision, these features could strengthen long-term retention and support the app's learning identity, particularly for students using Amadeus as a regular practice companion. [10]

In summary, these developments increase both conversion (source separation as a premium "dense mix" upgrade) and retention (saved recordings as an ongoing repertoire workflow), improving the long-term commercial outlook of the app.

ACKNOWLEDGMENT

Special thanks to Andy Hunt for supervision, and to the creators of the open-source libraries that made this project possible: the AudioKit team for their audio engine and the maintainers of the Tonic library. Thanks to Michael McLoughlin (University of York), Filip Korzeniowski (Moises.ai), Christian Dittmar (AudioLabs Erlangen) and Eloi Moliner (Aalto/Meta) for external consulting regarding automatic chord recognition research.

REFERENCES

[1] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, 'A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation', May 12, 2022, arXiv: arXiv:2203.09893.

[2] J. Pauwels, K. O'Hanlon, E. Gómez, and M. B. Sandler, '20 YEARS OF AUTOMATIC CHORD RECOGNITION FROM AUDIO', 2019.

[3] P. López-Serrano and C. Dittmar, 'NMF Toolbox: Music Processing Applications of Nonnegative Matrix Factorization', 2019.

[4] AudioKit Pro, "AudioKit: Swift audio synthesis, processing, and analysis platform," GitHub repository, 2023. [Online]. Available: https://github.com/AudioKit/AudioKit

[5] AudioKit Pro, "Tonic: Music theory library for Swift," GitHub repository, 2023. [Online]. Available: https://github.com/AudioKit/Tonic

[6] C. L. Krumhansl, *Cognitive Foundations of Musical Pitch*. Oxford University Press USA, 1990.

[7] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, 2000.

[8] N8 CiR / Bede HPC, "Bede: Regional Tier 2 Supercomputing Facility," 2025.

[9] M. Müller, *Fundamentals of Music Processing*. Springer, 2015.

[10] F. Franchino, "Amadeus: SimVenture Validate Market Testing Portfolio," University of York coursework submission, Jan. 2026. (Unpublished internal report).

[11] F. Franchino, "Automatic Chord Recognition: A Review of Temporal Segmentation and Stability," Literature review (ELE000189M), Jan. 2026. (Unpublished manuscript).

[12] Spotify, "Basic Pitch," GitHub repository, 2022. [Online]. Available: https://github.com/spotify/basic-pitch

[13] B. McFee *et al.*, "librosa: Audio and Music Signal Analysis in Python," in *Proc. SciPy*, 2015. [Online]. Available: https://librosa.org

[14] SoundFile, "PySoundFile," GitHub repository. [Online]. Available: https://github.com/bastibe/python-soundfile

[15] FFmpeg Developers, "FFmpeg," 2026. Available: https://ffmpeg.org

[16] J. Robert, "pydub," GitHub repository. [Online]. Available: https://github.com/jiaaro/pydub

[17] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, 2020.

[18] Google, "Google Trends," 2026. [Online]. Available: https://trends.google.com

[19] C. Harte, "Towards automatic extraction of harmony information from music signals," Ph.D. dissertation, Queen Mary Univ. of London, 2010.

[20] C. Raffel, B. McFee, E. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mireval: A transparent implementation of common MIR metrics," in *Proc. ISMIR*, 2014.

[21] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in *Proc. ISMIR*, 2005.

[22] F. Korzeniowski and G. Widmer, "Improved chord recognition by combining duration and harmonic language models," in *Proc. ISMIR*, 2018.