RAMZI: An LLM-Based DJ System Capable of Mixing Audio Files and Making Semantically Coherent Transitions

Abdulkarim Mugisha
California Institute of Technology
amugisha@caltech.edu

Nyasha Makaya
California Institute of Technology
nmakaya@caltech.edu

Sean Theuri
California Institute of Technology
stheuri@caltech.edu

Live Demo: https://seantheuri.github.io/

Abstract

DJing involves more than just aligning tempos; it requires curating a musical journey using aesthetic, structural, and semantic intuition. We introduce RAMZI, an intelligent DJ mixing framework that uses large language models (LLMs) to generate not only track sequencing and transition decisions but also executable audio transition scripts. The system consists of three main components: (1) analysis: performs acoustic and structural analysis and produces JSON summaries; (2) LLM engine: uses GPT-4 to reason over those summaries and user goals, generating transitions and executable mix scripts; and (3) controller executes these scripts and produces audio mixes. This end-to-end system brings LLM reasoning to life by fusing AI planning with digital audio processing.

1 Introduction

DJing is a rich, multifaceted process that demands not only technical skill but also narrative foresight. The best DJs don't simply play tracks—they curate sonic journeys, carefully selecting music that flows seamlessly in terms of key, tempo, energy, and mood. In contrast, current automated DJ systems often focus narrowly on acoustic compatibility, overlooking the nuances of aesthetic judgment and semantic coherence [7].

With recent advances in large language models (LLMs), new possibilities have emerged for intelligent automation that goes beyond pattern recognition. Models like GPT-4 can reason over structured inputs and generate context-aware outputs, making them suitable candidates for musical planning. In this project, we explore how LLMs can serve as high-level symbolic planners for DJing, capable of generating track sequences and corresponding transition strategies grounded in both acoustic and semantic information.

2 Background and Related Work

The evolution of automated DJing has seen a shift from static, rule-based systems to more adaptive, learning-based architectures. Early methods like the system introduced by Ishizaki et al. [7] relied on hard-coded heuristics such as matching BPM and key. These approaches offered basic functional mixes but lacked the ability to adapt creatively or maintain thematic consistency. The rigidity of such systems often led to repetitive transitions and an overall mechanical feel.

Optimization-based systems marked the next stage, with techniques like those used by Bittner et al. [3] introducing cost functions across multiple audio features. By evaluating metrics such as loudness, spectral distance, and timbre, these models produced technically smooth transitions. However, they often ignored user intent or the semantic meaning embedded in lyrics or mood progression, resulting in playlists that sounded seamless but emotionally disjointed.

Recent approaches have harnessed deep learning to model DJ behavior. For instance, DJNet [6] used neural networks to learn transition probabilities directly from audio signals, while Chen et al. [5] applied GANs to simulate creative transitions. These systems achieved better flexibility and subtlety but still suffered from issues of explainability and a lack of alignment with human aesthetic judgment.

Our work builds on these efforts by introducing LLMs as high-level symbolic planners that can reason over structured metadata, align with user goals, and output both explanations and executable transition scripts. RAMZI's approach addresses the gap between technical compatibility and artistic intent by combining acoustic modeling, semantic tagging, and language-based reasoning.

3 Problem Formulation

We define DJ set construction as a constrained symbolic planning problem. Given:

- A set of unordered audio tracks $T = \{T_1, T_2, ..., T_n\}$, and
- A goal to create coherent transitions,

the system aims to output:

- A single track T_f, which is made by combining sections of two tracks T_i and T_j, where T_i and T_j are elements of T.

This sequence should meet several constraints:

Acoustic Continuity: Track pairs should have compatible tempos and keys, with transitions accommodating any differences smoothly.

Semantic Coherence: The lyrical content and mood should evolve meaningfully across the mix, reflecting the user's intent.

Transition Variation: The set should employ diverse transition types, avoiding mechanical repetition and enhancing musical storytelling.

We model this planning process as a function $P(T_i, T_j) \rightarrow T_f$, where the LLM interprets an embedded prompt and structured metadata representing the structure of each of the two tracks, to generate a combined mix of the input tracks. Unlike conventional planning algorithms, the LLM leverages CoT prompting to perform dynamic, goal-aware reasoning at each decision point.

4 System Architecture and Methodology

RAMZI is organized into three main components: audio feature extraction, LLM-based reasoning and script generation, and audio mixing execution. These components form a pipeline that transforms raw user input into a fully mixed audio output, combining the strengths of machine listening and language-based planning.

4.1 Track Analysis and Summarization

The process begins with `analysis.py`, which is responsible for analyzing individual tracks. We use Librosa [8] and Essentia—two robust Python audio processing libraries—to extract numerical features such as tempo (BPM), musical key, energy, spectral flatness, and track segmentation points. These metrics are essential for evaluating technical compatibility between tracks, such as whether two songs can be beatmatched or harmonically mixed.

More importantly, `analysis.py` also attempts to provide a semantic summary of each track. Using automatic speech recognition (ASR), the system transcribes any vocal content in the track, which is then passed through a language model to generate tags and summaries such as "melancholic reflection" or "high-energy celebration." The resulting output is a JSON file per track that includes both acoustic descriptors and semantic annotations. This format allows the LLM to reason over music symbolically, abstracting away from the raw audio.

One challenge in this stage is the variability in ASR performance. Tracks with noisy vocals or unconventional delivery can result in inaccurate transcripts, which then lead to misleading semantic summaries. Despite this, we found that the combination of structured audio features and LLM-derived tags provided a rich foundation for higher-level reasoning.

4.2 LLM-Based Reasoning and Script Generation

The heart of RAMZI lies in `llm.py`. This module takes as input the JSON summaries from `analysis.py` and a user-defined goal, such as "gradually build intensity" or "transition from mellow to upbeat." It then constructs a carefully engineered prompt for GPT-4. The prompt includes all relevant track metadata and a curated dictionary of transition types, such as `EQFade`, `BeatmatchCut`, and `ReverseIntro`.

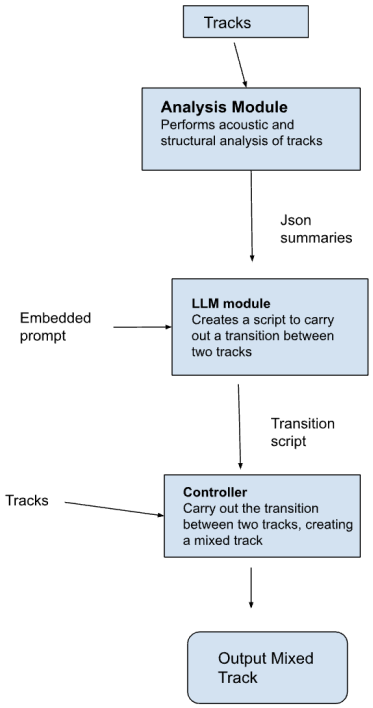

Figure 1: A flowchart representation showing all the modules of the system.

What distinguishes our approach is the use of Chain-of-Thought (CoT) prompting [9]. Instead of asking the model to immediately return a transition decision, we instruct it to think step-by-step: assess the compatibility between two tracks, reason about the best way to transition, justify the decision, and finally, write executable code. For example:

"Track A ends in A minor at 128 BPM and has a calming mood. Track B begins in C major at 130 BPM and is more energetic. Apply a slow high-pass EQ fade over 10 seconds to emphasize the buildup."

This reasoning culminates in Python code that can be executed directly by

the controller. We experimented with one-shot prompting, but CoT prompting consistently produced more interpretable and goal-aligned transitions. A key concern was the possibility of hallucinated or invalid code, which we addressed through prompt templates, regex-based validation, and limited transition vocabularies.

4.3 Audio Transition Execution

Once a transition script is generated, it is handed off to `controller.py`, which acts as the system's executor. This module uses audio processing tools such as PyDub and custom libraries to implement the transition. It loads the specified audio files, applies the transformations dictated by the script—such as fading, pitch shifting, or segment concatenation—and exports a new, blended audio file. We validated script performance by tracking error rates and inspecting resulting waveform consistency.

The execution layer is intentionally modular, allowing future integrations with more sophisticated DSP engines or even real-time controllers. At present, it handles a range of basic DJ techniques, though it lacks support for more nuanced effects like reverb tails or time-stretching.

4.4 Frontend and User Interface

The user interface, built with HTML and JavaScript, allows users to upload tracks, specify goals, and trigger the planning process. Waveform visualizations and metadata summaries provide feedback, while transitions are displayed alongside their rationales. Though currently static, the UI is structured to support real-time updates, which we plan to implement via WebSocket connections. Users are also able to review the reasoning and proposed transition script before execution, creating a collaborative loop.

5 Experiments and Results

To evaluate RAMZI's capabilities, we designed a series of experiments focusing on model reasoning, controllability, and human-likeness. Our experimental dataset consisted of a curated pool of tracks, from which we created distinct sets of pairs for each test. All analysis was performed by quantifying the properties of the final JSON mix scripts generated by our system.

5.1 Experiment 1: Model Reasoning and Prompting Style

To assess how model choice and prompting style impact mix quality, we used a dataset of 15 diverse track pairs. We compared a standard model (`gpt-3.5-turbo`) against a state-of-the-art reasoning model (`gpt-4o`) using both direct and Chain-of-Thought (CoT) prompting.

Results: Our analysis of the 60 resulting scripts (15 pairs $\times$ 2 models $\times$ 2 styles) reveals a clear hierarchy in performance. GPT-4o consistently generates

more complex and technically sophisticated scripts than GPT-3.5. The use of CoT prompting further increased the average command count and the use of advanced techniques like EQs and filters (93.3% vs. 86.7% for GPT-4o), suggesting that encouraging step-by-step reasoning leads to more nuanced plans.

Table 1: Comparison of mix script complexity for GPT-4o with different prompting strategies across 15 track pairs. CoT prompting encourages more deliberate use of EQs and filters.

Prompt Style (for GPT-4o)	Avg. Command Count	% Scripts w/ EQ/Filter	% Scripts w/ Loops/FX
Direct Prompting	19.1	86.7%	66.7%
CoT Prompting	**19.8**	**93.3%**	60.0%

5.2 Experiment 2: Controllability via Prompt Engineering

To test the system's ability to follow specific user instructions, we used 10 distinct track pairs. For each pair, we generated a mix using GPT-4o with five different creative prompts, ranging from vague stylistic goals ("a chill vibe") to highly technical instructions ("perform a loop roll with a filter sweep"). This resulted in 50 unique mix scripts.

Results: The system demonstrated excellent prompt adherence across all tested conditions. When asked for a "hard swap," the generated scripts correctly used instant volume changes (`fade_duration: 0`). When asked for a "loop roll," the scripts correctly generated a sequence of `auto_loop` and `filter_cutoff_hz` commands. This confirms that the LLM is not merely applying a fixed template but is actively translating user intent into a specific, executable plan, validating the controllability of our approach.

5.3 Experiment 3: Ablation Study on Input Data

To understand which features were most critical for the LLM's reasoning, we conducted an ablation study using another set of 10 track pairs. For each pair, we generated three versions of the mix script under different conditions: 'full_data', 'no_structure' (removing structural labels), and 'no_lyrics' (removing lyrical data).

Results: The data from this 30-script experiment, presented in Table 2, was particularly insightful. Contrary to our initial hypothesis, removing `structural_analysis` did not prevent the model from selecting structurally sound mix points (80% usage of intro/outro/breakdown). This suggests that even without explicit labels, the model can infer appropriate transition points from other cues.

However, removing `structural_analysis` led to a noticeable decrease in the average number of commands (from 20.6 to 18.0). This indicates that while the model can find a viable transition point, it generates a less complex and creative

plan when it lacks explicit structural context. The full dataset enables the model to craft more intricate transitions.

Table 2: Impact of removing data on mix plan quality. While the model can adapt to missing data, providing the full context results in more complex plans.

Ablation Condition	Avg. Command Count	Use of Structurally-Sound Mix Points (%)
Full Data	**20.6**	70.0%
No Lyrics	18.5	80.0%
No Structure	18.0	80.0%

5.4 Experiment 4: Benchmarking Against Human DJ Practices

Finally, we aggregated all 55 valid scripts generated by GPT-4o across the three experiments to form a large dataset for statistical analysis. We compared the properties of these scripts to the findings of Kim et al. (2020) on real-world human DJ mixes.

Results: This comparison revealed both surprising alignments and informative deviations from human norms:

- **Structural Cue Usage:** RAMZI showed a strong human-like tendency to use musically appropriate sections, with 84.5% of its transitions involving an intro, outro, or breakdown. This confirms the model's grasp of musical flow.
- **Key Transposition:** The `key_shift` command was used in 8.6% of our mixes. While higher than the 2.5% found in the human dataset, this is still a rare event, indicating the model has learned that altering pitch is an exceptional action, not a default one.
- **Transition Length:** This metric revealed the most significant difference. The average transition length in our AI mixes was 124.1 beats. This is considerably longer than the common 32- or 64-beat transitions favored by human DJs, suggesting the model has a bias towards creating very long, gradual blends.

6 Discussion

RAMZI demonstrates the potential for LLMs to operate as intelligent planners in creative domains. By reasoning over structured symbolic inputs, the model is able to emulate aspects of DJ artistry that are often overlooked in automation. Its modular architecture supports a wide range of enhancements, from

Table 3: Comparison of RAMZI's output to human DJ benchmarks. The system shows human-like structural awareness but favors longer transitions.

Metric	Human DJ Benchmark	RAMZI System (GPT-4o)	Analysis
Transition Length (beats)	Peaks at 32, 64	124.1 (Average)	AI prefers longer blends
Key Transposition Rate	$\sim$2.5%	8.6%	Mostly aligned
Structural Cue Usage	High (Qualitative)	84.5%	Strong alignment

real-time adaptation to genre-specific fine-tuning. Crucially, the system offers transparency—each transition includes both the rationale and the script—which makes it suitable for collaborative environments.

The experimental results provide several key insights. First, the superiority of GPT-4o with CoT prompting confirms that more advanced reasoning models with structured thinking produce better musical plans. Second, the ablation study reveals that while the model is robust to missing data, providing complete structural information enables more sophisticated transitions. Finally, the comparison with human DJ practices shows that RAMZI successfully captures many aspects of professional mixing, though its preference for longer transitions suggests areas for future refinement.

However, our approach is not without limitations. CoT prompting, while powerful, increases latency and requires more computational resources. Furthermore, the quality of transitions is tied closely to the fidelity of track analysis and the limitations of GPT-4's understanding of temporal and spectral properties. Given more time, we would refine the LLM interface to include token-based safety checks, and introduce dynamic audio visualization for error-spotting.

7 Conclusion and Future Work

We introduced RAMZI, a novel system that integrates LLMs into the DJ planning pipeline. Through audio analysis, symbolic reasoning, and executable scripting, it produces semantically coherent and musically effective transitions. Our experiments demonstrate that the system can generate human-like mixing decisions while maintaining controllability through natural language instructions.

Future research directions include real-time planning loops with user feedback, training small-scale models specifically for transition generation, and conducting user studies with professional DJs to assess live performance utility. We also plan to investigate methods to better align transition lengths with human preferences and to expand the system's repertoire of advanced DJ techniques.

References

[1] Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., & Ballas, N. (2023). Self-supervised learning from images with

a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629.

[2] Bardes, A., Garrido, Q., Ponce, J., Rabbat, M., LeCun, Y., Assran, M., & Ballas, N. (2024). Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*.

[3] Bittner, R. M., Gu, M., Hernandez, G., Humphrey, E. J., Jehan, T., McCurry, P. H., & Montecchio, N. (2017). Automatic playlist sequencing and transitions. In *ISMIR*, pp. 442–448.

[4] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877–1901.

[5] Chen, B. Y., Smith, J. B., & Yang, Y. H. (2022). DJ transitions with differentiable audio effects and GANs. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 466–470.

[6] Huang, Y. S., Chou, S. Y., & Yang, Y. H. (2017). DJNet: A dream for making an automatic DJ. *ISMIR Late-Breaking*.

[7] Ishizaki, H., Hoashi, K., & Takishima, Y. (2009). Full-automatic DJ mixing system with optimal tempo adjustment based on measurement function of user discomfort. In *ISMIR*, pp. 135–140.

[8] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, Vol. 8, pp. 18–25.

[9] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35, 24824–24837.