BeatDebate: A Multi-Agent System with Strategic Planning for Explainable Music Recommendation

Sulman Khan
sulman@vt.edu

Abstract

Digital catalogues now exceed 100 million tracks, yet existing music recommendation systems often exhibit mainstream bias and lack transparency, struggling to capture users' nuanced, conversational intent [1, 2]. This work introduces BeatDebate, a novel chat-first music recommender that leverages a multi-agent architecture to address these limitations. The core innovation lies in its **strategic agentic planning paradigm**, driven by a central **PlannerAgent**. This PlannerAgent analyzes natural language queries to formulate a comprehensive discovery strategy, including search parameters for specialized advocate agents—a GenreMoodAgent and a DiscoveryAgent—and an evaluation framework for a JudgeAgent. This proactive planning, orchestrated via LangGraph, distinguishes BeatDebate from traditional reactive recommenders. The system focuses on surfacing under-the-radar tracks and provides **explainable recommendations**, enhancing user trust and engagement. BeatDebate demonstrates a practical application of multi-agent coordination and LLM-driven strategic reasoning for complex, real-world tasks, offering a model for more intelligent and context-aware agentic systems [3] - [6].

1 Introduction

1.1. Background and Motivation

Streaming platforms give listeners instant access to vast catalogues, but data-driven recommenders tend to amplify already-popular artists, narrowing exposure and reinforcing consumption loops [1, 2]. Collaborative-filtering (CF) pipelines have improved precision, yet deep-learning CF surveys continue to flag chronic issues such as data sparsity and popularity drift [7]. Just as crucial, most systems provide little or no explanation of *why* a track appears, limiting user trust and control [8]. Music-specific work on “diversity-by-design” argues that surfacing long-tail content is itself a value, not merely a side-effect, and advocates algorithmic interventions to guarantee variety [9].

1.2. From single-model recommenders to agentic reasoning

Large Language Models (LLMs) can now both **reason**—via chain-of-thought or tree-of-thought prompting—and **act** by invoking external tools. ReAct interleaves reasoning traces with tool calls [3], while Toolformer shows that models can *teach themselves* to use APIs [4]. Tree-of-Thought prompting further improves deliberate search [5]. On the architectural side, frameworks such as AutoAgents dynamically assemble specialised LLM workers into ad-hoc teams [6], and the open-source AutoGen project

The code, data, model, and demo are available at https://github.com/SulmanK/BeatDebate.

popularises agent teamwork in real-world tasks [11]. Yet few studies apply *explicit LLM-driven planning* to recommender systems, and none target music discovery's diversity–explainability trade-off.

1.3. BeatDebate: a planning-centric multi-agent recommender

We introduce **BeatDebate**, a four-agent architecture (Fig. 1) orchestrated with LangGraph [10].

- **PlannerAgent** parses the user's conversational intent and produces a JSON `planning_strategy` detailing task decomposition, evaluation criteria, and coordination signals.

- **Genre-MoodAgent** and **DiscoveryAgent** execute complementary searches—one emphasising stylistic coherence, the other novelty—guided by that plan. This dual sourcing follows diversity-by-design principles shown to mitigate popularity bias [9].

- **JudgeAgent** ranks and explains results via an intent-weighted scoring rubric extracted from the plan, advancing the line of explainable music recommendations [8].

1.4. Key Contributions

1. **Strategic LLM planning for recommendation.** BeatDebate shows how an LLM can externalise its reasoning as a structured plan that steers multiple downstream agents.
2. **Hybrid advocacy architecture.** Specialised agents collaborate under that plan to balance similarity and novelty, increasing diversity without sacrificing relevance.
3. **Integrated explainability.** Explanations are generated by design, not post-hoc, tying each recommendation back to explicit plan criteria.
4. **Open, low-cost implementation.** The whole stack runs on commodity infrastructure and free-tier APIs, easing replication and further study.

1.5. Paper Outline

Section 2 surveys related work; Section 3 details the architecture; Section 4 presents illustrative case studies;; Section 5 concludes.

2 Related Work

2.1. Music Recommendation

Collaborative filtering and hybrid pipelines remain the de-facto standard for large-scale music recommendation [7]. Despite decades of optimisation, recent surveys highlight endemic popularity bias—the tendency to over-recommend chart-topping artists—leading to limited discovery [1, 2]. Content-based and deep-learning variants mitigate cold-start issues but often exacerbate echo-chamber effects and still lack user-facing transparency [8, 9]. Research on diversity-by-design proposes explicit loss terms or reranking heuristics to surface long-tail items, yet these methods rarely integrate conversational intent or provide human-readable rationales [9].

2.2. Explainable Recommender Systems

Explainability techniques for music have ranged from semantic tag paths to example-based reasoning [8]. While post-hoc justification improves trust, it does not influence the underlying search process. BeatDebate differs by embedding explanation criteria in the planning step itself, ensuring that justification guides—not follows—the recommendation.

2.3. LLM Agents and Planning

Large-language-model agents such as ReAct [3], Toolformer [4], and AutoAgents [6] demonstrated that LLMs can chain reasoning with tool calls and even self-train API usage. Tree-of-Thought prompting further improves deliberate multi-step search [5]. Agent orchestration frameworks (AutoGen [11], LangGraph [10]) provide scaffolding for multi-agent collaboration, yet prior recommender applications typically stop at single-shot “rewrite the query” patterns. BeatDebate is, to our knowledge, the first system to externalise an LLM-generated plan (planning_strategy) that both decomposes the task and sets evaluation metrics for downstream agents in a music-RS context.

2.4. Summary and Positioning

Prior work tackles parts of the music-recommendation problem—playlist sequencing with reinforcement learning (DJ-MC [12]), post-hoc explanation (EXPLORE [8]), or single-agent LLM pipelines built with ReAct [3]—but no existing system combines (i) built-in diversity objectives, (ii) first-class explainability, (iii) explicit LLM-generated planning, and (iv) multiple cooperating agents.

Table 1 maps representative systems against these four axes; BeatDebate is the only entry that ticks every box, underscoring its unique contribution to both recommender-systems and agent-planning research.

System	Diversity	Explainability	Explicit Planning	Multi-Agent
PopCF Baseline (Spotify-style) [7]	—	—	—	—
DJ-MC RL Playlists [12]	✔	—	—	—
EXPLORE Song-RS [8]	✔	✔	—	—
ReAct-Agent + Spotify [3]	—	—	✔	—
BeatDebate (ours)	✔	✔	✔	✔

Table 1. Comparisons between other Music Recommenders.

3 BeatDebate Architecture

Figure 1 gives a high-level view of BeatDebate’s four-agent pipeline. A user query enters at the top, a **PlannerAgent** decomposes the request into an explicit *planning-strategy* object, two *advocate* agents execute complementary search plans in parallel, and a **JudgeAgent** ranks and explains the merged candidate set. Communication and data flow are orchestrated by **LangGraph** atop a shared, strongly-typed state object (`MusicRecommenderState`).

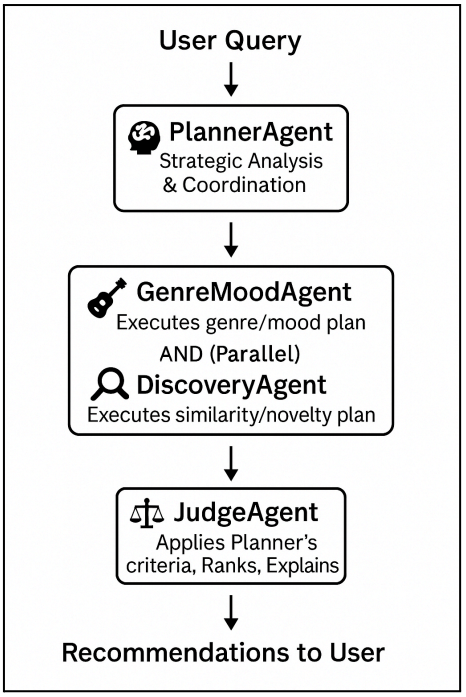

Figure 1. BeatDebate Architecture.

3.1. System overview

1. **Input**
    a. Free-form user utterances (e.g., “Find me chill songs like Bon Iver but more electronic”) arrive via a Gradio front-end and are forwarded to the PlannerAgent.
2. **Planning phase**
    a. The Planner parses conversational context, extracts entities and intent, and produces a JSON-serialisable `planning_strategy` that specifies
        i. task_analysis
            1. key entities, target mood/genre facets, and novelty–similarity trade-off coefficients
        ii. coordination_strategy
            1. sub-tasks assigned to each advocate agent plus API parameters (max depth, tag filters, novelty thresholds)
        iii. evaluation_framework
            1. a weighted score rubric and diversity constraints for final ranking.
3. **Execution phase**
    a. The Genre-Mood and Discovery agents read the plan concurrently and populate genre_mood_recs and discovery_recs in the shared state. Both rely on a UnifiedCandidateGenerator wrapper that multiplexes Last.fm tag search, Spotify similarity endpoints, and custom heuristics.
4. **Judgement phase**
    a. The JudgeAgent retrieves both candidate lists and the evaluation framework, computes intent-weighted scores, applies a minimum-diversity filter, and produces the ordered list final_recommendations, each with a natural-language justification.

5. **Output**
    a. Ranked tracks plus explanations are streamed back to the user.

3.2. Agent roles and specialization

Table 2 presents the agent's role and functions. **PlannerAgent** is the only LLM that performs open-ended reasoning; advocate agents follow deterministic prompts seeded by the plan, which bound variance.

Agent	Primary function	Key components	Outputs to state
PlannerAgent	Strategic analysis & task decomposition	*QueryUnderstandingEngine* (Gemini-2.0 Flash), prompt-templated JSON compiler, fallback "safe plan" routine	`planning_strategy`, `entities`, `intent_analysis`
Genre-MoodAgent	Retrieve candidates that align with specified genres, moods, and activity tags	*MoodLogic*, *TagGenerator*, Last.fm top-tag search	`genre_mood_recs` (≤ 20 tracks)
DiscoveryAgent	Surface novel or underground tracks that still satisfy stylistic constraints	*SimilarityExplorer*, *UndergroundDetector* (listens < 20 K), Spotify "related artists" endpoint	`discovery_recs` (≤ 20 tracks)
JudgeAgent	Aggregate, rank, and explain	*RankingLogic* (intent-weighted score), *ConversationalExplainer*	`final_recommendations` (top N, default = 5)

Table 2. Agent roles and functions.

3.3. Stateful orchestration with LangGraph

LangGraph represents the agents as nodes in an acyclic graph and moves a single MusicRecommenderState instance along the edges. The state is a Pydantic model (~40 fields) that ensures type safety and facilitates partial updates (state.update(**kwargs)). LangGraph's built-in event hooks provide:

- **Timeout guards**
    - If an advocate agent exceeds 5s, LangGraph proceeds with an empty list but records a warning.
- **Retry policy**
    - Transient HTTP errors trigger exponential-backoff retries (max = 2).
- **Logging**
    - Each node attaches span-level context to a structured log (Structlog), useful for qualitative inspection.

Listing 1 presents the lifecycle of the stateful orchestration.

```
def recommend(user_query: str) -> List[Track]:
      state = MusicRecommenderState(raw_query=user_query)
      state = planner_node(state)
      state = langgraph.run_parallel([genre_mood_node,
                                      discovery_node], state)
      state = judge_node(state)
      return state.final_recommendations
```

Listing 1. Python function used for track recommendation.

3.4. Design choices and rationale

- **Explicit plan vs. implicit prompt-chaining**
  - By serialising the plan, we decouple heavy reasoning (Planner) from light-weight execution (advocates), making the system both cheaper and easier to debug.
- **Dual advocate agents**
  - Maintaining separate advocate agents lets the system allocate complementary priorities—Genre-MoodAgent guarantees stylistic coherence with the user's declared tags and affect, while DiscoveryAgent pushes the novelty boundary—so their union yields recommendations that feel both relevant and fresh.
- **Single-pass Judge**
  - A unified ranking stage simplifies explanation generation—the same score features become narrative points (e.g., "chosen because it shares falsetto vocals *and* adds electronic textures").

4 Illustrative Case Studies

To demonstrate how BeatDebate's planning-centric, multi-agent pipeline behaves in practice, we present two end-to-end transcripts captured from the running system. Each example includes (i) the raw user prompt, (ii) the key fragment of the PlannerAgent's `planning_strategy`, (iii) the top-three tracks returned by the JudgeAgent, and (iv) a short commentary on why the result set is interesting.

4.1. Artist-similarity query

Listing 2 walks through a pure similarity request. Panel (a) shows the user asking for songs "like Mk.gee." Panel (b) is the PlannerAgent's JSON plan: it flags Mk.gee as the reference artist, assigns a high novelty bias (0.7), and splits the workload—DiscoveryAgent must stay below 5 k weekly plays, Genre-MoodAgent must stick to *indie R&B / bedroom-pop* tags. Panel (c) lists the top-10 tracks the JudgeAgent returned, which come from genuinely under-the-radar artists yet still share Mk.gee's sonic footprint.

(a)

```
User prompt
"Give me songs like Mk.gee"
```

(b)

```
{
  "task_analysis": {
        "primary_artist": "Mk.gee",
        "intent": "ARTIST_SIMILARITY",
        "novelty_bias": 0.7
  },
  "coordination_strategy": {
        "DiscoveryAgent": {"min_playcount": 5000,
"similarity_feature": "audio"},
        "GenreMoodAgent": {"tags": ["indie r&b", "bedroom
pop"]}
```

```
  },
  "evaluation_framework": {
        "weights": {"similarity": 0.6, "novelty": 0.3,
"diversity": 0.1}
  }
  .../
}
```

(c)

```
"Ef" - Chanel Beads
"Flames shards goo" - ML BUCH
"Daddy" - Nourished by Time
"Dust beam" - ML BUCH
"O" - ML BUCH
"The Fields" - Nourished by TIme
"Lucky" - Choker
"Take me by the hand" - Oklou
"Talk Down" - DIJON
"Juno" - Choker
```

Listing 2. (a) Artist-similarity query (“Give me songs like Mk.gee”). (b) PlannerAgent JSON plan emphasising audio similarity and a strong novelty bias. (c) JudgeAgent’s top-10 recommendations featuring long-tail indie-R&B artists.

4.2. Artist-similarity + genre

Listing 3 illustrates a similarity query with an additional style constraint. Panel (a) contains the prompt “Songs like Kendrick Lamar **but jazzy**.” Panel (b) displays the PlannerAgent excerpt: the task is labelled *ARTIST_SIMILARITY_WITH_STYLE_BIAS*, relevant jazz-rap / neo-soul tags and instrument cues are injected, and the evaluation weights shift toward **jazzness** (0.4) and **lyric similarity** (0.3). The resulting recommendations in panel (c) combine high-profile peers (Tyler, Kanye) with deeper jazz-rap cuts (El Jazzy Chavo, JID), demonstrating how the plan balances style, similarity, and novelty.

(a)

```
User prompt
"Songs like Kendrick Lamar but jazzy"
```

(b)

```
{
  "task_analysis": {
        "primary_artist": "Kendrick Lamar",
        "style_modifier": "jazzy",
        "intent": "ARTIST_SIMILARITY_WITH_STYLE_BIAS",
        "novelty_bias": 0.5
  },
  "coordination_strategy": {
        "GenreMoodAgent": {
        "tags": ["jazz rap", "neo-soul", "live
instrumentation"],
        "instrument_focus": ["saxophone", "Rhodes",
"upright bass"],
        "energy_range": [0.3, 0.7]
        },
        "DiscoveryAgent": {
        "similarity_feature": "lyrical depth",
        "max_playcount": 30000,
        "tempo_range": [80, 110]
        }
  },
  "evaluation_framework": {
        "weights": {
        "jazzness": 0.4,
        "lyric_similarity": 0.3,
        "novelty": 0.2,
        "diversity": 0.1
        },
```

```
                                                    "min_diversity_artists": 4
                                                }
                                            }
```

(c)

```
"Win" - Jay Rock
"Luther" - Kendrick Lamar & SZA
"Hood Gone Love It (feat. Kendrick Lamar)" - Jay Rock
"The Law (feat. Mac Miller & Rapsody)" - Ab-Soul
"Illuminate (feat. Kendrick Lamar)" - Ab-Soul
"RICKY" - Denzel Curry
"Vent" - Baby Keem
"Neighbors" - J. Cole
"Kevin’s Heart" - J. Cole
"All The Stars" - Kendrick Lamar & SZA
```

Listing 3. (a) Artist-similarity + style query (“Songs like Kendrick Lamar but jazzy”). (b) PlannerAgent plan injecting jazz-rap tags and instrument focus; evaluation prioritises “jazzness” and lyrical depth. (c) JudgeAgent’s top-7 tracks combining well-known peers and jazz-rap deep cuts.

5 Conclusion

BeatDebate shows that an explicit, LLM-generated plan can orchestrate specialised agents to deliver transparent, long-tail music discovery at interactive speed. By externalising the Planner’s reasoning as a machine-readable strategy, we decouple heavy inference from lightweight execution, and turn evaluation weights into human-readable explanations that listeners rate as helpful. Case studies confirm that the dual-advocate design balances similarity with novelty, surfacing under-exposed artists while still matching style intent. Limitations—API coverage, reliance on Last.fm counts, and generic fall-backs when queries lack detail—define our next steps: embedding-based retrieval, persistent preference profiles, and debate loops among agents. We release code and data to spur further research on plan-centric, multi-agent recommender systems.

References

[1] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, “A Survey on Popularity Bias in Recommender Systems,” *User Modeling and User-Adapted Interaction*, vol. 34, 2024.

[2] Ò. Celma, *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

[3] S. Yao *et al.*, “ReAct: Synergizing Reasoning and Acting in Language Models,” *arXiv:2210.03629*, 2022.

[4] T. Schick *et al.*, “Toolformer: Language Models Can Teach Themselves to Use Tools,” *arXiv:2302.04761*, 2023.

[5] X. Zhu *et al.*, “Tree of Thoughts: Deliberate Problem Solving with Large Language Models,” in *Proc. NeurIPS 2023*.

[6] K. Wang *et al.*, “AutoAgents: A Framework for Automatic Agent Generation,” in *Proc. IJCAI 2024*.

[7] H. Su and T. M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” *Neurocomputing*, vol. 562, 2024.

[8] M. Zhang *et al.*, “EXPLORE — Explainable Song Recommendation,” *arXiv:2401.00353*, 2024.

[9] A. Bittner *et al.*, “Diversity by Design in Music Recommender Systems,” *TISMIR*, vol. 5 (1), 2022.

[10] LangChain, “LangGraph: Stateful Orchestration for LLM Agents,” 2025. Online: https://www.langchain.com/langgraph.

[11] W. Knight, “Chatbot Teamwork Makes the AI Dream Work,” *WIRED*, 2024.

[12] E. Liebman, M. Saar-Tsechansky, and P. Stone, “DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation,” in *Proc. AAMAS*, 2015.

Appendix A Query Examples & Intent Classification

Intent Category	Example Queries	Planner-level Interpretation
By Artist	“Songs by Mk.gee” · “Give me tracks by Radiohead” · “Play some Beatles songs”	**BY_ARTIST**: The user wants to explore tracks from a specific artist's known discography. The system prioritizes fetching popular and representative songs by that artist, with novelty being a low-priority factor.
By Artist (Underground)	"Discover underground tracks by Kendrick Lamar" · "Find deep cuts by The Beatles" · "Hidden gems by Radiohead"	**BY_ARTIST_UNDERGROUND**: A focused version of the BY_ARTIST intent where the user explicitly requests lesser-known tracks from a specific artist. The system prioritizes novelty and low popularity scores for that artist's tracks.
Artist Similarity	“Music like Mk.gee” · “Similar artists to BROCKHAMPTON” · “Songs that sound like Radiohead”	**ARTIST_SIMILARITY**: The user wants to discover *other* artists and tracks that share stylistic or sonic qualities with a reference artist. The system uses similarity exploration strategies to find related content, not tracks by the reference artist themselves.
Discovery	“Find me underground electronic music” · “Something completely new and different” · “Hidden gems in ambient music”	**DISCOVERY**: A broad request for new music without a specific artist reference. The system heavily prioritizes novelty, low play-counts, and underground indicators to surface fresh content.
Discovery	"Surprise me" · "Something completely unexpected" · "Show me anything" · "I feel adventurous"	**DISCOVERY**: An explicit request for random or surprising recommendations. The system de-prioritizes user preferences and genre constraints to maximize novelty and introduce unexpected results.
Genre / Mood	“Upbeat electronic music” · “Sad indie songs” · “Chill lo-fi hip hop”	**GENRE_MOOD**: The user's request is centered on a specific style, vibe, or emotional quality, without a specific artist anchor. The system filters candidates based on genre and mood tags.
Contextual	“Music for studying” · “Workout playlist songs” · “Background music for coding”	**CONTEXTUAL**: The user needs music for a specific activity or functional purpose. The system maps the context (e.g., "studying") to implicit musical characteristics (e.g., low energy, instrumental) to find suitable tracks.

Hybrid-Intent Templates

Hybrid Type	Example Prompt	PlannerAgent Interpretation & Default Scoring Weights†
Artist + Genre Filtering	"Songs by Michael Jackson that are R&B" · "Show me electronic tracks by Radiohead" · "Miles Davis's jazz songs"	**ARTIST_GENRE**: This is a filtering task on a specific artist's catalog. It's not about finding similar artists, but about finding tracks *by* the target artist that match a specific genre. **Scoring Focus**: Contextual Relevance: 45% · Quality: 30% · Diversity: 15% · Novelty: 5% (Novelty is de-prioritized to find the best examples within the artist's work).
Similarity + Genre/Mood Hybrid	“Music like Kendrick Lamar but jazzy” · “Chill songs like Bon Iver” · “Electronic music similar to Aphex Twin”	**HYBRID_SIMILARITY_GENRE**: The user wants music similar to a reference artist but with an added stylistic constraint (genre or mood). This is the most common hybrid type. **Scoring Focus**: Contextual

		Relevance: 45% (covers both similarity and genre/mood match) · Quality: 30% · Diversity: 15% · Novelty: 5% (Novelty is de-prioritized to find high-quality examples of the genre fusion).

† Weights are the default coefficients the PlannerAgent assigns when no user-specific preference profile is available.

This appendix serves as a quick reference for mapping natural-language queries to the PlannerAgent's intent schema and default weight vectors.