Flow

Project Brief: AI DJ System

Overview

We want to build an **AI-powered DJ engine** that can generate and mix a curated setlist based on user prompts. The system should accept natural language input (e.g., venue, vibe, time schedule, audience preferences), transform that into a structured setlist, analyze each track, and then propose transitions/mixes between them.

The system is composed of **three engines**:

1. **Track Identification Engine (NLP → Setlist Generator)**
2. **Track Analysis Engine (Spotify API Integration)**
3. **Mixing Engine (Transition Logic)**

User Flow Example

User Input:

"I need a mix between 7pm and 10pm for a Casino. At 8pm there will be dinner, then dancing starts at 9pm. Most of our customers prefer R&B, Bollywood, Afrobeats and these songs specifically [list provided]."

System Output:

- A time-structured setlist (song order)
- Metadata (tempo, genre, energy, transitions, notes)
- Suggested transition/mixing instructions

Engine Specs

1. Track Identification Engine (NLP → Setlist Generator)

- **Input:** Natural language description of event (audience, vibe, time slots, preferred genres/artists/songs).

- **Processing:**
    - Pass prompt to OpenAI API (or equivalent LLM).
    - Generate a structured setlist: ordered list of tracks for each time segment.
    - Ensure maximum BPM difference of **±5** between consecutive tracks.
- **Output:** JSON setlist with:

```
{
"time": "19:00–20:00",
"tracks": [
{ "title": "Song A", "artist": "Artist A" },
{ "title": "Song B", "artist": "Artist B" }
]
}
```

2. Track Analysis Engine (Spotify API Integration)

- **Input:** List of tracks from Engine 1.
- **Processing:**
    - Use Spotify API (or equivalent) to fetch metadata per track.
    - Required attributes:
        - tempo (BPM)
        - key
        - genre
        - energy
        - valence
        - danceability
    - Custom computed attributes:
        - contextual vibe label (e.g., *"Sunset Chill"*, *"Peak Energy"*) from playlist co-occurrence or clustering.
        - recommended transition type between songs.

- **Output:** JSON track analysis with metadata and transition notes, e.g.:

```
{
"track": "Song A",
"artist": "Artist A",
"bpm": 120,
"key": "C#m",
"genre": "Afrobeats",
"energy": 0.8,
"valence": 0.6,
"danceability": 0.9,
"transition": "EQ sweep",
"notes": "Peak energy track for dance floor."
}
```

3. Mixing Engine (Transition Logic)

- **Input:** Ordered tracks with metadata from Engine 2.
- **Logic:**
    - Align mixes at **first chorus** by default (ref: example).
    - Respect BPM/key compatibility.
    - Transition type rules:
        - If BPM difference ≤ 3 → crossfade.
        - If BPM difference 3–5 → EQ sweep or echo-drop.
        - If >5 → fade-out/fade-in.
    - Add contextual mixing notes (e.g., *"build tension before drop at 21:00"*).
- **Output:** Mixing plan per track pair, e.g.:

```
{
"from_track": "Song A",
"to_track": "Song B",
"transition_point": "first chorus",
```

```json
  "transition_type": "EQ sweep",
  "comment": "Smooth handoff into higher-energy section."
  }
```