Emotion Based Playlist Generator: Project Specification (Revised)

1. Project Overview

The **Emotion-Based Playlist Generator** is a web-based application designed to create personalized music playlists based on the user's current emotional state. Users provide a brief textual description of their mood, and the application uses Natural Language Processing (NLP) to analyze the sentiment. Based on this analysis, the application generates a playlist that matches the user's emotions, drawing from a pre-curated collection of songs tagged with emotional metrics such as **valence**, **arousal**, and **dominance**.

2. Users and Stakeholders

Primary Users:

- Music enthusiasts looking to curate playlists that match their current mood.

Secondary Stakeholders:

- **Music therapists**: Interested in exploring how music can impact mood and well-being.
- **Music streaming services**: Services like Spotify may benefit from understanding user preferences for mood-based music recommendations.

3. Problem to Solve

This application aims to create an emotionally responsive music experience, where users can enhance or alter their mood by listening to music tailored to their emotional state. By automatically generating playlists based on the user's current emotions, the app saves users time and effort, providing a more intuitive music selection process.

4. Major Features

Feature 1: Sentiment Analysis of User Input

- Users input a short text description of their mood (e.g., “I feel great today”).
- **Natural Language Processing (NLP)** via **TextBlob** analyzes the mood description, calculating:
    - **Polarity**: Determines if the mood is positive, negative, or neutral (values from -1 to 1).
    - **Subjectivity**: Determines whether the mood is factual or subjective (values from 0 to 1).

Feature 2: Song Selection Based on Sentiment Tags

- The system matches the user’s mood with a pre-curated dataset of songs that include emotional tags:
    - **Valence**: Measures the pleasantness of the song.
    - **Arousal**: Measures the energy or excitement level.
    - **Dominance**: Measures the degree of control or power conveyed by the song.

Feature 3: Web-Based Interface Using Flask

- The application is developed as a web-based system using Flask.
- Users interact with the app through a web interface, entering their mood on the homepage and receiving a playlist on the results page.

Feature 4: Playlist Display

- The playlist displays each song’s **name** and **artist** in a visually appealing format:
    - Song names are displayed in **bold**.
    - Artist names are displayed in **italics**.

Feature 5: Navigation

- Users can easily return to the homepage to generate new playlists by clicking a **“Generate Another Playlist”**button.

5. Technical Flow

The following describes how the data flows through the application:

- **Input**:
  - The user provides a text description of their mood.
- **Processing**:
  - **TextBlob** processes the user's input and calculates the **polarity** and **subjectivity** scores.
  - Based on the polarity score, the application queries the **Musical Sentiment Dataset** to find songs that match the user's emotional state. The system uses the **valence_tags**, **arousal_tags**, and **dominance_tags**columns in the dataset to match songs to the user's mood.
- **Output**:
  - The application generates a playlist of songs that match the mood. The songs are displayed on the results page with their names and artists.

6. Changes Based on Feedback

- **(REVISED)**: Sentiment analysis now relies on **polarity** and **subjectivity** scores from **TextBlob** instead of sentiment-related words.
- **(REVISED)**: The application now uses the **Musical Sentiment Dataset** from Kaggle, which contains pre-tagged songs. This eliminates the need for user-provided music libraries.
- **(REVISED)**: The application has moved from a potential desktop or CLI interface to a web-based system using **Flask**.
- **(FUTURE PROPOSAL)**: Integration with music streaming services (e.g., Spotify) is marked as a "nice-to-have" feature. This will be implemented after the core functionalities, and only if time permits.

7. Task Vignettes (User Activity "Flow")

Task 1: User Inputs Mood Description

1. Bob, a music lover, visits the application's homepage.
2. He is feeling excited and types "I'm super pumped for today" into the text input field.

3. Bob presses the **Generate Playlist** button.

Task 2: Sentiment Analysis and Playlist Generation

1. The application processes Bob's input using **TextBlob**, which identifies the input as highly positive.
2. The system queries the **Musical Sentiment Dataset** and selects songs with high **valence_tags** and **arousal_tags**.

Task 3: Viewing the Playlist

1. Bob is redirected to the results page, where he sees a playlist with three songs.
2. Each song is listed in bold, with the artist name in italics.
3. A button allows Bob to return to the homepage to generate another playlist.

8. Data and Processing

- **Data Source**: The **Musical Sentiment Dataset** from Kaggle is used, containing pre-curated songs tagged with emotional metrics.
- **Processing**: **TextBlob** analyzes the user's mood input, and the system uses this analysis to query the dataset based on **valence**, **arousal**, and **dominance** tags.

9. Workflow

1. The user enters a brief description of their mood on the homepage.
2. The app analyzes the input using **TextBlob** and retrieves a sentiment score.
3. The system queries the **Musical Sentiment Dataset** and generates a playlist based on the sentiment tags.
4. The playlist is displayed to the user, with the option to generate another playlist.

10. Milestones for Next Steps

1. **Full Sentiment Tag Integration (Version 2 Progress Report A)**:
    - Integrate the remaining sentiment tags (**arousal_tags** and **dominance_tags**) for more refined playlist generation.

2. **Error Handling and Optimization (Version 2 Progress Report B)**:
    - Implement error handling and optimize playlist generation logic for edge cases.
3. **UI and Minor Backend Adjustments (Version 2 Progress Report C)**:
    - Focus on refining the user interface, making minor backend adjustments, and polishing the front-end experience.

11. Future Features (Nice to Have)

- **Spotify Integration**:
    - If time permits, the app will integrate with the Spotify API to allow users to listen to generated playlists directly in the app.