Emotion-Based Playlist Generator - Project Specification

1. Project Overview

Description:
The Emotion-Based Playlist Generator is a web-based application designed to curate 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 sentiment analysis, the application generates a playlist that aligns with the user's emotions, drawing from a pre-curated collection of music categorized by emotional tags (e.g., valence, arousal, dominance).

External Mechanisms:

- **Natural Language Processing (NLP):** The application will utilize the TextBlob library for sentiment analysis, specifically analyzing polarity (ranging from -1 to 1) and subjectivity (ranging from 0 to 1).
- **Music Dataset:** The app will use the **Musical Sentiment Dataset** from Kaggle, which contains songs tagged with **valence**, **arousal**, and **dominance** scores.
- **Web Framework:** The application will be developed using Flask to provide a lightweight, web-based interface.
- **Music Integration:** In future iterations, the project may involve API integration with music streaming services like Spotify or Apple Music for playlist playback.

Graphical User Interface (GUI):
The application will feature a simple, browser-based interface powered by Flask. Users will input their mood description via a web form and view the generated playlist on a results page. Flask will handle routing and data processing, and the front-end will be built using HTML and CSS.

Remote Control:
The application could potentially offer an API in the future, allowing remote or automated playlist generation based on mood descriptions provided via HTTP requests.

2. Task Vignettes (User Activity "Flow")

Task 1: User Inputs Mood Description
Vignette:
Sarah, a music enthusiast, is feeling nostalgic and wants to listen to music that matches her mood. She opens the Emotion-Based Playlist Generator in her browser. The home page greets her with a text box where she describes her feelings: "I feel sad and nostalgic." Sarah then clicks the "Generate Playlist" button to submit the form.

Technical Details:

- The text input is captured as a form submission using Flask's `POST` method.
- Sentiment analysis is performed using the TextBlob library to determine **polarity** and **subjectivity**.
    - **Polarity:** Indicates the degree of positivity/negativity.
    - **Subjectivity:** Measures how subjective or factual the statement is.
- The app will map these values to the appropriate **valence**, **arousal**, and **dominance** tags from the pre-curated song dataset.

Task 2: Playlist Generation and Display
Vignette:
After Sarah clicks "Generate Playlist," the application processes her input and Flask renders a results page displaying the playlist. The playlist includes songs like "Someone Like You" by Adele and "Yesterday" by The Beatles. Sarah can listen to the tracks using links or explore more recommendations by adjusting her mood description.

Technical Details:

- The application queries the **Musical Sentiment Dataset** to match the sentiment values (polarity and subjectivity) with **valence**, **arousal**, and **dominance** tags in the song dataset.
- Flask renders an HTML template displaying the playlist, with options to export the list to a music streaming service (future feature).

Task 3: Refining the Playlist
Vignette:
Sarah wants to refine her playlist. She goes back to the homepage and adjusts her mood description to "I feel deeply nostalgic," which generates a more fitting playlist. The new playlist matches her updated emotional intensity.

Technical Details:

- Flask allows users to resubmit their mood description through the form, triggering a new analysis.
- A reanalysis is performed, and the updated playlist is rendered on the results page.

3. Technical Flow

Data Flow:

- **Input:** The user inputs a text description of their mood via a web form.
- **Processing:**

    - Sentiment analysis is performed using TextBlob to extract **polarity** (sentiment score) and **subjectivity**(degree of subjectivity).
    - The application queries the **Musical Sentiment Dataset** to find songs tagged with matching **valence**, **arousal**, and **dominance** scores that align with the polarity and subjectivity values.
- **Output:** Flask renders a results page showing the generated playlist, with options for playback or future export.

Blocks:

- **Sentiment Analysis Block:**
    - Processes user input and determines mood using TextBlob's polarity and subjectivity values.
    - Maps sentiment values to the valence, arousal, and dominance tags of songs from the pre-curated dataset.
- **Playlist Generation Block:**
    - Matches mood with pre-tagged songs based on valence, arousal, and dominance.
    - Generates a playlist based on the sentiment analysis.
- **Flask Web Interface Block:**
    - Handles routing for the homepage, form submission, and results page.
    - Displays the input form and playlist results using HTML templates.
- **API Integration Block (Future Work):**
    - Interfaces with music streaming APIs for playback and exporting playlists.

Data Types:

- **Input Data:** User mood description as a string (submitted via form).
- **Processing Data:** Polarity and subjectivity scores as floats, music library as a dataset with valence, arousal, and dominance tags.
- **Output Data:** Playlist as a list of song objects, rendered as HTML.

User Interaction:

- **Input:** Users provide a textual description of their mood via a web form.
- **Output:** The playlist is displayed on a results page, with options for playback or export (future).

4. Changes Based on Feedback

- The TextBlob sentiment analysis will now be based on **polarity** and **subjectivity** values, not sentiment-related words. These values will be used to map to **valence**, **arousal**, and **dominance** tags in a song dataset.

- The application will use the **Musical Sentiment Dataset** from Kaggle, instead of relying on user-provided songs.
- Flask will be used for the web interface, replacing the desktop interface or CLI.
- Future plans include integrating music streaming services such as Spotify for playlist playback or purchasing recommendations.