Phase 2: Architecture & Design

In this phase, we'll lay the foundation for your AI music streaming platform by designing the system architecture, database, and API structure, as well as planning the mood detection algorithms and frontend design.

1. Database Design

Core Tables:

- Users:
  - id: Primary key, unique user identifier.
  - username: User's display name.
  - email: User's email address.
  - password_hash: Encrypted password.
  - created_at: Timestamp of account creation.
  - preferences: JSON field storing user preferences (genres, artists, etc.).

- Songs:
  - id: Primary key, unique song identifier.
  - title: Song title.
  - artist_id: Foreign key linking to the artist.
  - album: Album name.
  - genre: Song genre.
  - mood_tags: Array of moods associated with the song (e.g., "Energetic", "Calm").
  - tempo: Numeric value representing the song's tempo (BPM).
  - duration: Song duration in seconds.
  - spotify_id: Identifier for linking to Spotify's metadata.

- Artists:
  - id: Primary key, unique artist identifier.
  - name: Artist's name.
  - genre: Primary genre of the artist.
  - popularity: Popularity score (derived from Spotify or calculated within the app).
  - spotify_id: Identifier for linking to Spotify's metadata.

- Playlists:
  - id: Primary key, unique playlist identifier.
  - user_id: Foreign key linking to the user who created the playlist.
  - name: Playlist name.
  - mood_tag: Mood associated with the playlist (optional).
  - created_at: Timestamp of playlist creation.

- Listening History:
  - id: Primary key.
  - user_id: Foreign key linking to the user.
  - song_id: Foreign key linking to the song.
  - played_at: Timestamp when the song was played.
  - mood: User-selected mood at the time of listening (optional).
  - device: Device used for playback (optional).

Relationships:

- Users can create multiple playlists.
- Songs can be associated with multiple moods.
- Artists can belong to multiple genres.

2. API Development

Core API Endpoints:

- User Management:
  - POST /api/register: Register a new user.
  - POST /api/login: Authenticate a user and return a JWT token.
  - GET /api/profile: Retrieve user profile details.
  - PUT /api/profile: Update user preferences and settings.

- Song Management:
  - GET /api/songs: Fetch a list of songs, with optional filters (genre, artist, mood).
  - GET /api/songs/{id}: Retrieve details of a specific song.
  - POST /api/songs/recommendations: Fetch mood-based song recommendations.

- Playlist Management:
  - GET /api/playlists: Retrieve user playlists.
  - POST /api/playlists: Create a new playlist.
  - PUT /api/playlists/{id}: Update playlist details.
  - DELETE /api/playlists/{id}: Delete a playlist.

- Listening History:
  - GET /api/history: Retrieve user's listening history.
  - POST /api/history: Log a song playback event.

Third-Party Integration:

- Spotify API: Integration for retrieving song metadata, artist information, and linking songs to their Spotify counterparts.

- Machine Learning API (optional): If leveraging an external ML service for mood detection or recommendations.

3. Mood Detection Algorithm

Dataset Preparation:

- Song Metadata: Use Spotify API to gather detailed song metadata (tempo, energy, valence, etc.).
- Mood Labels: Manually label or source pre-labeled data for songs based on mood.

Model Selection:

- Feature Engineering: Extract features like tempo, key, mode, energy, and valence from songs to serve as inputs for mood classification.
- Modeling: Train a classification model (e.g., Random Forest, SVM, or a neural network) to predict song moods based on these features.
- Evaluation: Validate the model using a test set and adjust hyperparameters for accuracy.

Integration:

- Real-time Mood Classification: Use the trained model to classify songs in real-time as they are added to the platform.
- User Feedback Loop: Allow users to provide feedback on mood classification accuracy to further refine the model.

4. Frontend Design

Wireframing:

- Mood Selector Interface:
  - Create a simple, intuitive interface where users can select their current mood.
  - Include options like "Energetic," "Relaxed," "Focus," and "Uplifted."

- Playlist & Song View:
  - Display playlists with associated moods.
  - Highlight recommended songs based on the selected mood and user preferences.

- Explicit Input Handling:
  - Provide a textbox or voice input for users to express specific music needs (e.g., "Play something energetic for my workout").

UI/UX Considerations:

- Consistency: Ensure design consistency across different components.
- Responsiveness: Make sure the design is responsive for mobile and desktop users.
- Accessibility: Implement accessibility features for visually impaired users.

Prototyping:

- Use tools like Figma or Adobe XD to create interactive prototypes.
- Conduct usability testing to gather feedback on the interface and flow.

By completing these tasks in Phase 2, you?ll have a solid architectural foundation and design blueprint for the platform, enabling smooth development in the subsequent phases.