Spring Boot Final Project Proposal

Project Participants:
Nathan Aguilar

Title:
Music Playlist Manager API

Executive Summary:
In the front-end bootcamp, I built a mock music playlist application. For my final Spring Boot project, I plan to develop a back-end API to support that same concept with RESTful operations. The system will allow multiple users, each managing their own playlists filled with various songs. Users will be able to create, update, and delete their playlists and add or remove songs. This back-end service will provide full CRUD support for all main entities, with appropriate relationships and REST API endpoints.

Initial Features:
Database Design (ERD):
- **User** (One-to-Many with Playlist)
- **Playlist** (Many-to-Many with Song via Playlist_Song)
- **Song**
- **Playlist_Song** (Join Table)

Planned Entities:
- User
- Playlist
- Song
- Playlist_Song (Join Entity)

Planned Features:
- Users can create and manage their own playlists.
- Songs can be added to or removed from playlists.
- Full CRUD operations for:
  - Users
  - Playlists
  - Songs
- Manage song membership within playlists via join table.
- Relationship mapping:
  - One-to-Many: One User → Many Playlists
  - Many-to-Many: Playlist ↔ Song

API Endpoints:
Users
- GET /api/users – Retrieve all users

- GET /api/users/{id} – Retrieve a user by ID
- POST /api/users – Create a new user
- PUT /api/users/{id} – Update an existing user
- DELETE /api/users/{id} – Delete a user

Playlists

- GET /api/playlists – Retrieve all playlists
- GET /api/playlists/{id} – Retrieve a playlist by ID
- POST /api/playlists – Create a new playlist
- PUT /api/playlists/{id} – Update a playlist
- DELETE /api/playlists/{id} – Delete a playlist

Songs

- GET /api/songs – Retrieve all songs
- GET /api/songs/{id} – Retrieve a song by ID
- POST /api/songs – Create a new song
- PUT /api/songs/{id} – Update a song
- DELETE /api/songs/{id} – Delete a song

Playlist-Song (Join Table)

- GET /api/playlists/{id}/songs – Get all songs in a playlist
- POST /api/playlists/{id}/songs – Add a song to a playlist
- DELETE /api/playlists/{id}/songs/{songId} – Remove a song from a playlist

Client Testing Tools:

- Postman
- Front-end React client

Stretch Goals:

- User validation
- Like/favorite system for songs and playlists
- Playlist/song filtering and sorting options
- User profile picture and playlist cover image upload
- Include lyrics for songs