**Real-Time Emotion-Based Music Player - Project Plan**

**Project Overview:**

The “Real-Time Emotion-Based Music Player” is a web application that detects user emotions via facial recognition and recommends music from Spotify based on the user’s emotional state in real-time. The backend is developed using Spring Boot to ensure scalability and robustness.

**Task Division:**

The project is split into five tasks, each assigned to a team member for clear responsibility and efficient development.

### Task 1: Set Up the Spring Boot Project and Configure Dependencies

**Objective:** Create the foundational Spring Boot project with all necessary dependencies.

**Responsibilities:**

* Use Spring Initializr to generate a Spring Boot project.
* Include dependencies: Spring Web and Thymeleaf.
* Add Spotify Web API Java library to pom.xml:

<dependency>  
 <groupId>se.michaelthelin.spotify</groupId>  
 <artifactId>spotify-web-api-java</artifactId>  
 <version>8.0.0</version>  
</dependency>

* Verify the project builds and runs.

**Deliverable:** A working Spring Boot project setup.

### Task 2: Implement Spotify API Integration

**Objective:** Connect the application to the Spotify API to fetch music data.

**Responsibilities:**

* Register the app on the Spotify Developer Dashboard to get client\_id and client\_secret.
* Store credentials in application.properties:

spotify.client.id=your\_client\_id  
spotify.client.secret=your\_client\_secret

* Create a SpotifyService class to:
  + Authenticate using the Client Credentials Flow.
  + Fetch playlists or tracks based on emotion keywords (e.g., “happy”, “sad”).

**Deliverable:** A SpotifyService that retrieves music based on emotions.

### Task 3: Develop the Music Recommendation Logic

**Objective:** Build logic to map emotions to music recommendations.

**Responsibilities:**

* Create a MusicRecommender service that:
  + Takes an emotion (e.g., “happy”, “sad”) as input.
  + Uses SpotifyService to fetch relevant tracks.
* Map emotions to genres/moods (e.g., “happy” → upbeat pop, “sad” → slow ballads).

**Deliverable:** A MusicRecommender service providing emotion-based track suggestions.

### Task 4: Create the Web Controller and API Endpoints

**Objective:** Manage HTTP requests and responses for the application.

**Responsibilities:**

* Implement a MusicController with endpoints:
  + / : Serve the main page.
  + /recommend?emotion={emotion} : Return music recommendations.
* Use Thymeleaf to render the UI.

**Deliverable:** A controller handling user requests and responses.

### Task 5: Design the Frontend with Emotion Detection

**Objective:** Build the UI and integrate real-time emotion detection.

**Responsibilities:**

* Create an index.html Thymeleaf template.
* Integrate face-api.js for facial recognition:
  + Load models for face detection and emotion analysis.
  + Use the webcam to detect emotions in real-time.
* Use JavaScript to send detected emotions to the backend and display recommendations.

**Deliverable:** A frontend that detects emotions and updates music dynamically.

**Sample Artifact: Spring Boot Project Configuration**

pom.xml example:

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.example</groupId>  
 <artifactId>emotion-music-player</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Emotion-Based Music Player</name>  
 <description>A real-time emotion-based music player using Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.1.5</version>  
 <relativePath/>  
 </parent>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>se.michaelthelin.spotify</groupId>  
 <artifactId>spotify-web-api-java</artifactId>  
 <version>8.0.0</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Timeline:**

* **Week 1:** Task 1 - Project setup.
* **Week 2:** Task 2 - Spotify API integration.
* **Week 3:** Task 3 - Recommendation logic.
* **Week 4:** Task 4 - Web controller and endpoints.
* **Week 5:** Task 5 - Frontend with emotion detection.
* **Week 6:** Testing and deployment.

**Collaboration:**

* **Git:** Use branches for each task (e.g., task1-setup, task2-spotify).
* **Meetings:** Weekly updates to sync progress.
* **Documentation:** Document each component clearly for future reference.