Below is a detailed plan for the "Real-Time Emotion-Based Music Player" project using Spring Boot, divided into five tasks for each team member. This application will detect a user's emotions via facial recognition and recommend music from Spotify in real-time.

Project Overview

The "Real-Time Emotion-Based Music Player" is a web application that uses facial recognition to detect emotions and fetches music recommendations from Spotify based on the user's current emotional state. Built with Spring Boot, it ensures a scalable and robust backend.

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 the Spotify Web API Java library to `pom.xml`:

```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`:

```text
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

Here's an example of the `pom.xml` file for Task 1 to get you started:

Show inline

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.
- **Docs**: Document each component clearly.

This structure ensures each team member has a focused role, leading to a successful project completion. Let me know if you need more details!