Wiley Edge Final Project Team 2

Melodymap - Full Stack web application

Mahir Khan Nargis
Ruaraidh MacLennan
Nikoleta Koleva
Abdullah Khan

Team Introduction

Responsibilities:

- Mahir Khan Nargis - Database design, Backend Development, REST API Development and Spring Security
- Ruaraidh MacLennan - Frontend Development, Integration with Spotify API, Flowchart Creation
- Nikoleta Koleva - Unit testing, Backend Development, Spring Security, Class diagram
- Abdullah Khan - Frontend Development, CSS styling

Project Introduction

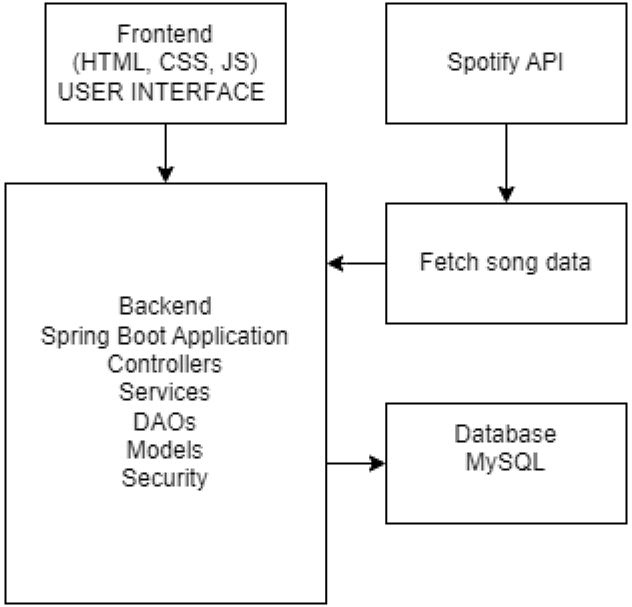
- Project Goal:
  - Develop a comprehensive music management system.
- Objective:
  - Create a full-stack application that allows users to execute CRUD operations on songs and playlists retrieved from the Spotify API.
- User Operations:
  - Create / Login with user accounts
- Song Operations:
  - Create, Read, Update, Delete (CRUD) operations with songs retrieved from the Spotify API
- Playlist Operations:
  - Create, Read, Update, Delete (CRUD) operations with playlists
- Artist Operations:
  - Create, Read, Update, Delete (CRUD) operations with artists

Technologies

- Frontend: HTML5, CSS3, JavaScript
- Backend: Spring Boot with JDBCTemplate, MySQL Workbench
- Database: SQL
- External API: Spotify Web API
- Security: Spring Security
- Testing: JUnit

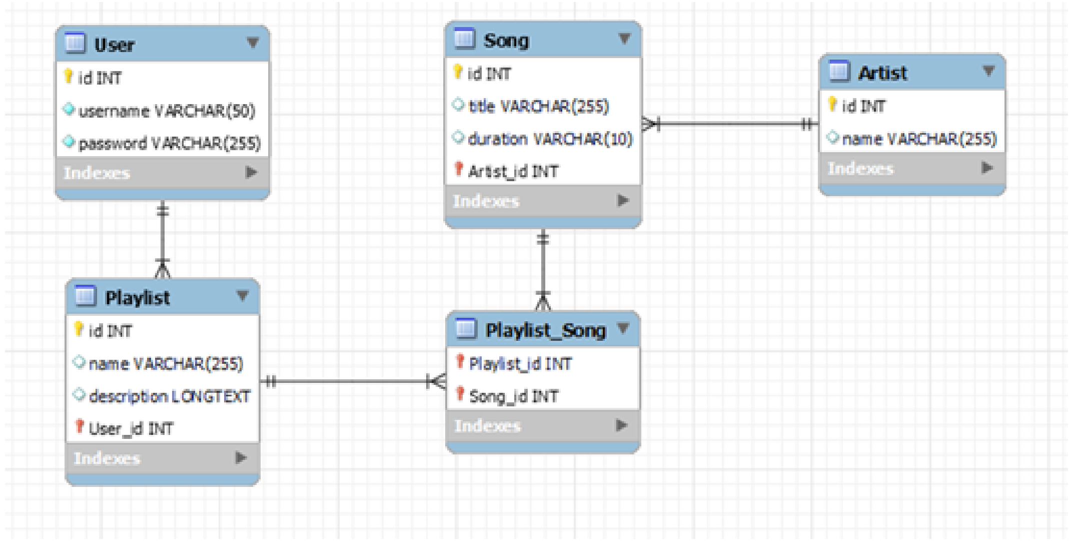
System Architecture

- Frontend
  - User Interface for managing users, songs, and playlists
- Backend
  - Controllers: Handling HTTP requests
  - Services: Business logic for CRUD operations
  - DAOs: Data access objects for interacting with the database
  - Security: Authentication and Authorization



Database

- User: Stores user credentials
- Playlist: Stores playlist details linked to users
- Song: Stores song details linked to artists
- Artist: Stores artist details
- Playlist_Song: Many-to-many relationship between playlists and songs



API Endpoints

- List key API endpoints and their functions:
    - POST /user/login: User login
    - POST /song/add: Add a new song
    - GET /playlist/{id}/songs: Retrieve songs in a playlist
    - POST /playlist/add: Create a new playlist
    - POST /playlist/{playlistId}/addSong/{songId}: Add a song to the playlist
    - DELETE /playlist/{id}: Delete a playlist
    - PUT /artist/{id}: Update artist
    - DELETE /artist/{id}: Delete artist

Unit Testing

- Stateful unit testing
  - DAO and service layer
  - Testing framework - JUnit 5
    - Test database
    - Set up methods
    - Clean up methods
    - Methods for testing CRUD operations and exception handling

Spring Security

- Password Encoder Bean: BCryptPasswordEncoder algorithm is used to encode passwords for secure storage and verification.
- Spring security is used to configure the security settings for the application.
- For development purposes we are allowing all requests without authorization.

Integration with Spotify API

- After creating a spotify account, ClientId is used for authentication
- Retrieve song and artist data: Query spotify database
- Update local database with Spotify data
- Example get song API call:

```
`https://api.spotify.com/v1/search?q=${query}&type=track&limit=10`
```

Frontend Development

Html:

- register.html
- login.html
- dashboard.html
- playlist.html

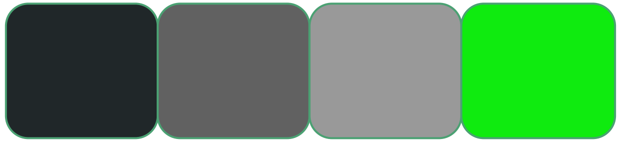
Css:

- registerstyles.css
- loginstyles.css
- dashboardstyles.css
- playliststyles.css

Javascript:

- registerscript.js
- loginscript.js
- dashboardscript.js
- playlistscript.js

Colour scheme:



Project Demo

- Users can create new accounts by registering with a username and password
- Users can log in with their credentials to access the dashboard
- Users can create a new playlist by providing a name and description
- Users can search for songs from the Spotify API
- Display search results with song details (title, artist, duration)
- Users can add selected songs from the search results to their playlist
- Ensure the song details are saved along with the playlist
- The playlist view displays all songs added to the playlist
- Show song details (title, artist, duration) within the playlist
- Users can remove songs from their playlist
- Provide an option to delete a song and update the playlist view accordingly
- The dashboard provides an overview of all playlists created by the user
- Upon returning to the dashboard, users can select a playlist to view its details

Future work

- Filter Spotify songs by genre/artist
- Adding genre table to the database
- Restrict access to certain endpoints, requiring authentication and proper authorization
- Login to Spotify and retrieve personal playlists
- Update existing playlists in Spotify

Questions and Answers

The End

Thank you for your attention!