MAD-2 Project Report

Music Streaming App- PlayIT

Author

Name: Ayan Nayyer
Email: 22f3000961@ds.study.iitm.ac.in
ID: 22f3000961
About me: I'm currently pursuing BS in Data Science and Applications as my standalone degree and currently in Diploma level. I have gained a lot of coding skills and insights of JavaScript,VueJS,Flask, Redis,Celery etc. upon working on this Application Development Project

Description:

Developed a multi-user web application for a Music Streaming App (PlayIT). Implemented functionalities based on VueJS for frontend and Flask for server side. The features also include scheduled jobs and reminders using redis and celery.

Technologies Used:

VueJS: Frontend of the application.
Flask: Backend/ Server-Side logic development of the application.
Flask-Restful: To develop the Restful API for the app.
Flask-Security: For Token based authentication.
Sqlite3 Flask-SQLAlchemy: Simplified database interaction with ORM.
Redis and Celery are used for scheduled jobs/daily reminders via Google Chat and Mail
Flask-mail: For sending monthly report emails.
Flask-CORS: To enable CORS for the app.
Matplotlib - to plot the app statistics graphs.
Bootstrap: For clean and visually appealing UI.
Jinja2: For generating monthly activity report at backend.
SQLite: Easy-to-use SQL database engine for data storage.

Database Schema Design:

Database models are created using flask-sqlalchemy.
7 Tables used in the database: User, Role, RolesUsers, Song, Album, Playlist, Playlist_Song_Table.
Roles of Users are stored in the RolesUsers table.
Song and Album have many to one relationship.
Playlist and Song have many to many relationships. This relationship data is stored in Playlist_Song_Table.

Constraints:
Primary keys ensure unique identification of records in each table.
Foreign keys establish relationships between tables, ensuring data integrity.

Reasoning:
The structure efficiently manages music data with relationships between users, artist and admin.
The use of foreign keys helps maintain referential integrity and ensures songs and playlists are linked to specific users.

API Design

CRUD on Albums:
Endpoint: /api/manage_albums, /api/edit_delete_album/<int:album_id>
Description: This API allows performing CRUD operations on Albums. It supports HTTP methods like GET, POST, PUT, and DELETE to manage album data in the database.

CRUD on Songs:
Endpoint: /api/manage_songs
Description: This API allows performing CRUD operations on Songs.

These APIs were implemented using Flask and Flask-Restful to define routes and resources. SQLAlchemy was used to interact with the database and the data is returned as JSON responses by Flask's jsonify.

Architecture and Features

The application follows the MVC structure:
Model(M) is handled by flask - It interacts with database and manages data.
View(V) is handled by vue.js. Vue components are responsible for interactive user interface.
Controller(C) is handled by flask. Flask routes handle all the business logic at the backend.

Project Features Implemented:

- Separate login for both Users/Artist and Admin.
- Admin has special privilege to see the app statistics and manage users, artists and songs.
- Users can filter songs based on likes/views.
- Users can play songs, read lyrics, like/dislike/ flag songs, create/edit/delete playlists.
- Users can register to become Artists.
- Artists can create Albums, Upload songs, Edit songs/Album and delete them.
- Monthly Activity report of Artist is sent to Artist’s email on first day of month.
- Daily notifications on google chat to users to visit the app if inactive for 24 hours.

Additional Features:

- Aesthetic Recommended Songs Layout.

Overall, the project implements core features for playing songs ensuring a smooth streaming experience. Additional features enhance the aesthetics and functionality of the web application, creating an efficient and user-friendly Music Streaming Application..

Video

For viewing the video, click here!

To run the application:

Unzip the project folder, install all the dependencies from requirements.txt, run the commands mentioned in imp_commands.txt and run the app.py file.