Author

Amisha Mishra
22f1000938
22f1000938@ds.study.iitm.ac.in
My name is Amisha Mishra, currently in 4th term of Diploma with MAD2, SC, TDS and MAD2 Project. I'm a student from Lucknow.

Description

We were supposed to make a Music Streaming App for multiple users with user-friendly features for users. Creators can add their own album and music and admin monitors the app altogether. General users have access to all the functionality including but not limited to playing songs, creating playlists and rating.

Technologies used

Python: For writing the backend code
Flask: For creating the backend API's
Flask-security: For Authentication and Authorization
Vue.js: JavaScript Framework for user interface
Bootstrap: For Styling
SQLite: Database used for storing App Data
Celery and Redis: For Batch jobs and Caching
Mailhog: For Sending Scheduled mails

DB Schema Design

User: Stores User Data with columns id(Integer), username(string), email(string), password(string), fs_uniquifier(string), is_creator(Boolean)

Role: Stores the three roles, i.e admin, creator and user id(integer), name(string) and description(string)

RolesUsers: Stores user_id(integer) and role_id(integer) for linking the User with the roles

Song: Stores Song data with columns id(integer), name(string), artist_id(integer), genre(string), duration(string), lyrics(string), date_added(string), is_approved(boolean), album_id(integer)

Album: Stores Album data with columns id(integer), name(string), artist_id(integer), is_approved(Boolean), release_date(string)

Playlist: Stores Playlist data created by users with columns id(integer), name(string) and user_id(integer)

Rating: Stores Rating given by a user for a particular song with columns id(integer), user_id(integer), song_id(integer), rating(integer)

API Design

The API design revolves around user authentication, song management, album management, playlist creation, rating songs, flagging songs, and more. Routes are structured to ensure proper security measures, including authentication and authorization checks.

Architecture and Features

The project follows a Model-View-Controller (MVC) architecture, with a clear separation between the backend and frontend. Key features include:
- **User Registration and Authentication:** Both regular users and creators can register and log in securely.

- **Song Management:** Creators can upload, edit, and delete songs, providing a diverse music catalogue.

- **Playlist Creation**: Users can create and manage playlists, adding their favourite songs.

- **Rating System:** Users can rate songs, contributing to an interactive and engaging community.

- **Monthly Reports and Reminders:** Automated email notifications are sent to users, providing monthly reports and daily reminders

Video

https://drive.google.com/file/d/1pS6_hn7LjCIGJKt7qovWADsAZU6gXPYH/view?usp=sharing