PROJECT DOCUMENTATION

Author

Name: Divyansh Dixit
Roll No.: 22f1001615
Email: 22f1001615@ds.study.iitm.ac.in

Description

Introducing Music Streaming App, a lightweight music app designed for music lovers. With it you can easily discover and enjoy various songs, explore lyrics, and delve into artists and genres. The app also lets you rate your favourite songs, adding a personalised touch to your musical journey.
It simplifies playlist creation, allowing you to gather all your favourite tunes in one place, tailored to your mood or occasion.
But there's more – it lets you become a creator too. With the "Creator" feature, you can upload your own songs and even release albums, sharing your musical creations with others effortlessly.
Designed to be user-friendly, It offers a seamless experience for exploring new music, singing along with lyrics, rating songs, and organising your musical world—all in one app. Immerse yourself in the world of It and enjoy music in a whole new way!

Technologies Used

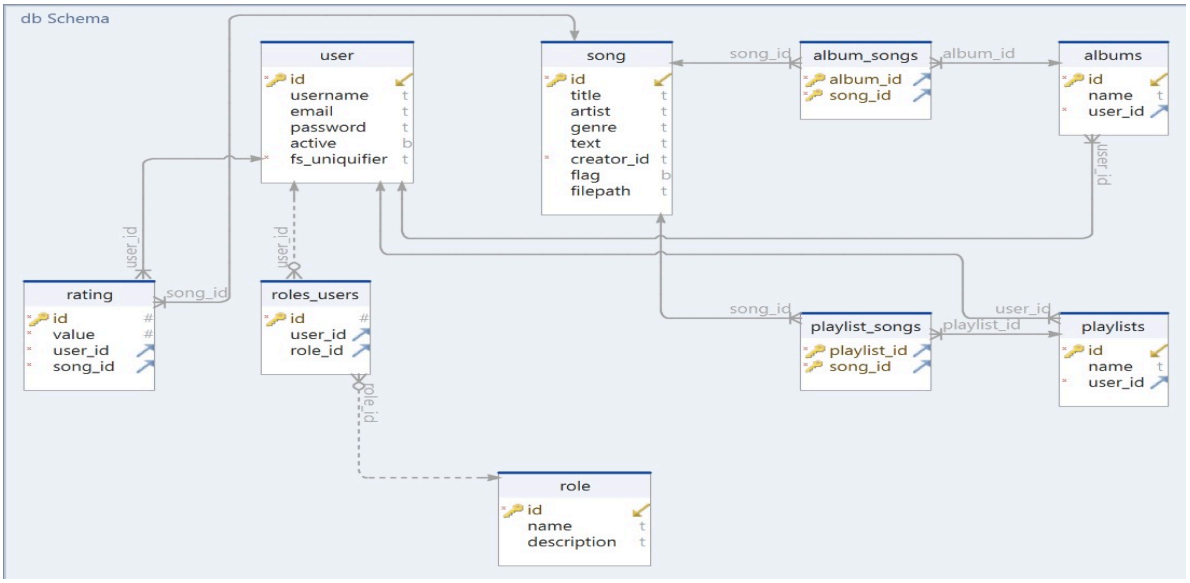
The application harnesses Flask, a potent web framework, for backend development. Key Flask extensions like Flask-Security for user authentication and Flask-SQLAlchemy for database management are seamlessly integrated. For frontend development, VueJS is employed, leveraging CDN for efficient resource loading, while Bootstrap enhances visual aesthetics. Vue-router ensures smooth navigation, while Pinia effectively manages state. Performance optimization is achieved through Redis for caching, supplemented by Flask-Caching. flask_mail facilitates email sending, while flask_cors enables cross-origin resource sharing. HTTP requests are managed using the 'requests' library. Additionally, Redis combined with Celery streamlines batch job handling.

API Design

The Flask-based API architecture utilises blueprints for streamlined functionality management. The Song Data API handles song operations, User Data API manages user tasks, and Playlist API enables playlist management. The Create Playlist and Create Album APIs focus on creating new playlists and albums, respectively. The Update User Role endpoint facilitates user role modifications. Finally, the Search Songs feature enhances user experience with flexible song searches. Leveraging Flask-RESTful and Flask-SQLAlchemy, the API ensures efficient data management and user interaction within the music app.

db Schema Design

The database model for the music app includes tables for User, Song, Playlist, Album, and Rating. Users are defined by attributes such as id, username, email, password, and role, with playlists linked to each user. Songs contain details like name, artist, lyrics, genre, and ratings. Playlists and Albums store user-created collections. The model establishes relationships between users, songs, and playlists/albums. Ratings connect users and songs, capturing user-assigned ratings. This schema forms the foundation for storing and managing user, song, playlist, album, and rating data within the music app, utilising Flask-SQLAlchemy for efficient database management.



Architecture and Features

In the root folder we have the views.py file,graph.py file, models.py, resources.py file,a readme.md file, a requirements.txt file, the instance folder with the SQLite database file, the template folder which contains the JS templates and the static folder with graph and audio file etc.

1. views.py : This file acts as a controller for the app and has all the routes to different pages.

2. models.py : This file contains a Python code for defining database models using SQLAlchemy and Flask-Login for the music streaming web application..

3. resources.py : All the api endpoints have been created and managed here.

4. Mail_service.py: For Sending Monthly Report using Celery.

5. workers.py : For managing Cache using redis.

There's a registration form for users with proper front-end and back-end validation. There's a login form for Listener, Creator and Admin. The Search bar on top helps the user, creator and admin to search for songs. The app has CRUD features for Songs, Playlists and Albums. Creators can edit/delete their songs/albums.Songs can be flagged or deleted by the admin. Also, the admin can Blacklist/Whitelist Creators if they breach any company Policy.

Presentation Video : Mad2_Presentation_Video