Vibezz - Project Documentation

Author

Santhosh G S	21F1001255	21f1001255@ds.study.iitm.ac.in

I am a final-year student pursuing a Bachelor's degree in Electrical and Electronics Engineering at SSN College of Engineering in Chennai. I am passionate about creating solutions that simplify life.

Description

Vibezz is a Music Library Management application where users can compose and add their albums/songs, manage their songs/albums in the library, and engage with their favourite music. When a user uploads a song to the library, their user rights are promoted to artist status. Users can listen to and rate their favourite music, add songs to playlists, create and edit playlists, and search for songs/albums. Users also have the ability to flag songs if they find them harmful. Flagged songs are then examined by an admin, who will make an appropriate decision regarding their status. The admin reserves the right to view the distribution of users or songs in an album, as well as the distribution of albums. Additionally, the admin has the authority to accept or reject deletion requests for flagged songs.

Technologies used

- Flask
- Flask-SQLAlchemy - ORM for SQLite database
- Flask-Caching (Redis) for caching
- Flask-Restful for API management
- Flask-JWT-Extended for authentication
- Celery-Redis for backend jobs
- SQLite - Database
- VueJs - Frontend Framework
- Bootstrap - Basic Styling
- HTML - Markup for skeleton content of the webpage
- Chart.js - Creating graphs on genre and book stats

DB Schema Design

Database schema design can be found in this document. The ER diagram of the database can be found here.

Table Name	Columns
User	id (PK, Int), name (Str), username (Str), email (Str), passhash (Str), role_id (Str)
Role	id (PK, Str), name (Str), description (Str)

Album	id (PK, Int), title (Uni8), artist_id (Int), description (Str), tot_ratings (Int), no_ratings (Int)
Song	id (PK, Int), title (Uni8), artist_id (Int), album_id (Int), lyrics (Uni8), genre (Str), date_created (DateTime), tot_ratings (Int), no_ratings (Int)
Playlist	id (PK, Int), title (Str), user_id (Int), date_created (DateTimel)
Flags	id (PK, Int), user_id (Int), song_id (Int)
Ratings	id (PK, Int), user_id (Int), song_id (Int), rating (Int)

API Design

APIs has been designed based on REST principles using Flask-Restful library

Architecture

Separation of Concerns: API endpoints are segregated into separate files, promoting modularity.
Centralised Configuration: Environment variables are managed in a dedicated configuration file.
Data Modeling: Database models are defined using an ORM, abstracting the database interactions.
Asset Management: Static assets - images, audios are stored in a dedicated directory
Scalability: The API design considers maintainability and scalability as the application grows in complexity.

Features

Every user post request is validated in the background before storing it in the database.
The search functionality can be performed on the HOME page based on album or song title
Users can also upload their own compositions to the library
Users can create their own playlist with all their preferred songs
Users can rate their favourite songs or flag songs with harmful content
Users can read the lyrics of the songs
The songs order in the playlist can be shuffled
The user registration page allows only passwords with a minimum of two LOWERCASE alphabets and two UPPERCASE alphabets and two numbers and two special characters
Admin can view the stats of available albums and songs and also the distribution of roles among users
Admin holds the final decision whether to retain or delete a song, if it has been flagged by an user

Video

[Vibezz - Demonstration Video]