Project Report

Author

Name: Praveen Choudhary
Roll number: 21f2000543
Email: 21f2000543@ds.study.iitm.ac.in

A Data Science student from Jodhpur, Rajasthan, completed my B.E. degree in Electrical and Electronics Engineering and pursuing IIT Madras B.S. in Data Science and Application.

Description

This project involves the development of a comprehensive music streaming application, tailored to manage a wide variety of musical content including songs, albums, and playlists. It features a user-friendly interface for different user roles such as administrators, creators, and regular users, each with distinct functionalities like song uploads, playlist creation, and content management. The application also integrates rating systems, search capabilities, and user management, enhancing the overall music streaming and sharing experience.

Technologies used

The project developed using following technologies:

- Flask: A lightweight and flexible Python web framework used for building the application's backend. It handles routing, requests, and server-side logic.
- SQLAlchemy: An Object-Relational Mapping (ORM) library for Python, used in conjunction with Flask. It provides an efficient way to interact with the application's database by mapping Python classes to database tables.
- SQLite: A lightweight, file-based database management system. It's used for storing and managing all the application data, including user information, song details, playlists, ratings, etc.
- HTML/CSS: Used for structuring and styling the front-end of the application. HTML is used to define the structure of web pages, while CSS is used to style and layout these pages.
- Jinja2 Templating Engine: Integrated with Flask, it is used for rendering dynamic HTML content on the web pages. It allows for the inclusion of Python-like expressions within the HTML, enabling dynamic content rendering.
- Bootstrap: A front-end framework used to design responsive and mobile-first web pages. It provides pre-designed components and classes to ensure the application's user interface is aesthetically pleasing and user-friendly.

IDE: Visual Studio Code

DB Schema Design

This database consists of following tables:

- User - Stores information about users of the application. Includes id (primary key), username, passhash (for password storage), email, is_admin (boolean to indicate if the user is an administrator), is_creator (boolean to indicate if the user is a content creator) and is_blacklisted (boolean to indicate if the user is blacklisted).

- Song - Holds details about each song available on the platform. Consists of id (primary key), name, genre (foreign key linked to the Genre table), creator_id (foreign key linked to the User table to identify the song's creator), artist_name, album_name, lyrics, release_date, and average_rating (for storing the computed average rating of the song).

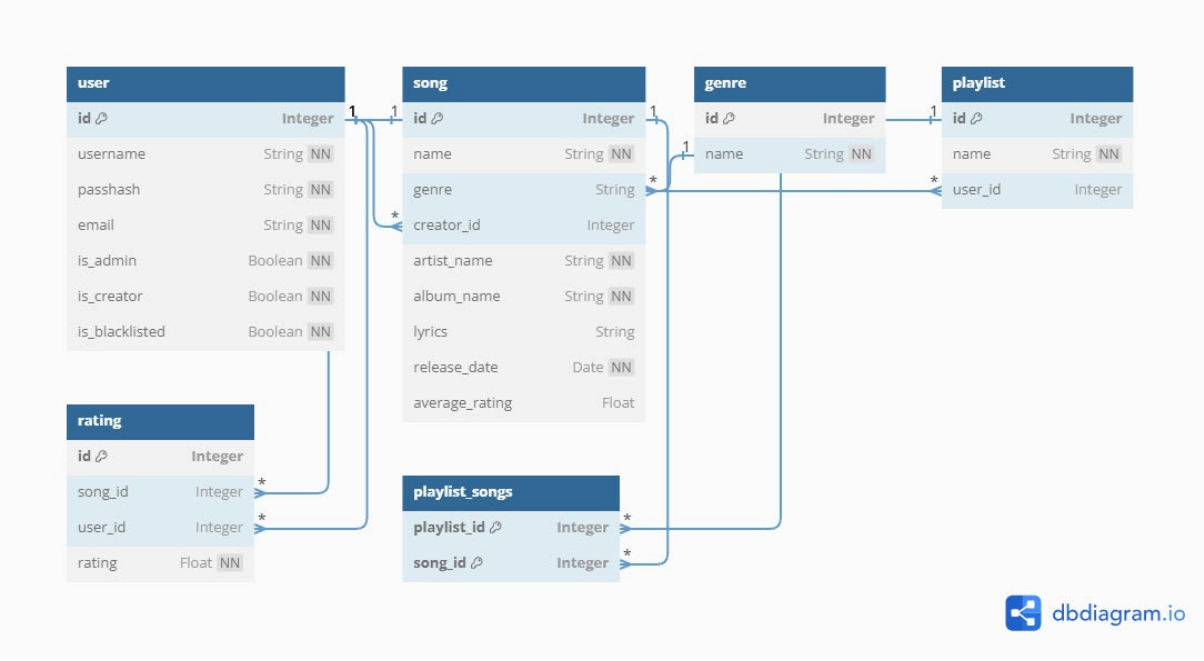
- Genre - Contains different music genres. Includes id (primary key) and name. Each genre can be associated with multiple songs.

- Playlist - Manages user-created playlists. Contains id (primary key), name, and user_id (foreign key linked to the User table to identify the playlist's owner). It is related to the Song table through a many-to-many relationship.

- Rating - Stores ratings given by users to songs. Includes id (primary key), song_id (foreign key linked to the Song table), user_id (foreign key linked to the User table), and rating. This table allows users to rate songs, and these ratings are used to compute the average rating for each song.

- Playlist_Songs – (Association Table) Facilitates the many-to-many relationship between the Playlist and Song tables. Consists of playlist_id and song_id as primary keys, both acting as foreign keys linking to the Playlist and Song tables, respectively.

ER Diagram of Database



Architecture and Feature

The Music Streaming Application is structured based on the Model-View-Controller (MVC) architecture, which organizes the application into three interconnected components. This separation helps manage complexity and facilitates efficient code organization and scalability.

1. Model:
   - Handles data and business logic.
   - Consists of database models (User, Song, Playlist, Genre, Rating).

- The models interact with a SQLite database through SQLAlchemy, which abstracts and simplifies database operations.

2. View:
   - Presents data to the user in a format they can interact with.
   - Implemented using HTML templates, styled with CSS.
   - Jinja2 templating engine is used for rendering the dynamic content.

3. Controller:
   - Processes user inputs and sends commands to the model and view components.
   - Flask routes act as controllers, handling HTTP requests and rendering appropriate views with data from models.

Features of the Application

1. User Authentication and Roles:
   - Secure user authentication system.
   - Different roles with distinct permissions (Admin, Creator, General User).

2. Music Management:
   - Users can search for songs, artists, albums, genres, and ratings.
   - Users can create and manage custom playlists.
   - Add songs to playlists and view them.
   - Users can become creator which gives additional capabilities like uploading new songs and managing their own songs.

3. Rating System:
   - Users can rate songs.
   - Average ratings are calculated and displayed.

4. Admin-Specific Features:
   - Manage users, including blacklisting functionality.
   - Delete songs or entire albums.
   - Oversee all content on the platform.

5. Responsive UI:
   - The application is designed to be responsive and user-friendly, adapting to different screen sizes.

6. Lyrics Viewing:
   - Detailed lyrics page for each song, with average ratings and an option to rate the song.

Presentation Video Link

https://drive.google.com/drive/folders/1IeonIIpm9eEBGj38zMmDFVB0Sp2Bw3Bz?usp=sharing