MODERN APPLICATION DEVELOPMENT – 1 REPORT

Author:

Name: Bhabatosh Acharya
Roll Number: 22f2000802
Email: 22f2000802@ds.study.iitm.ac.in

About: I am currently pursuing dual educational path i.e. BS degree in Data Science and Applications at IITM and B.Com Degree with Commerce Honours from Fakir Mohan Autonomous College. Passionate about continuous learning, I eagerly explore new domains to expand my knowledge.

Description:

The application is an Online Music Streaming App, where users can seamlessly explore and enjoy a diverse range of songs. Navigate effortlessly with our user-friendly interface, making it a breeze to discover your favourite artists or delve into curated Albums. Behind the scenes, our admin tools ensure smooth operations, managing content and ensuring an unparalleled music experience for all users.

Technologies Used:

- **Python** (Used for creating the backend)
- **Flask** (Used to create the web application)
- **Flask-SQL Alchemy** (Used for interacting with the SQLite Database)
- **Jinja2** (Used for templating in Flask)
- **SQLite** (Used for Storing Data)
- **HTML5** (Used to create structure for the front end)
- **CSS3** (Used for custom styling)
- **JavaScript** (Used for only Popup Box)
- **Bootstrap** (Used for basic styling)

Architecture and Features:

The project adheres to the MVC (Model-View-Controller) architecture, emphasizing the essential principle of separation of concerns. Key components include:

1. **Models (models.py):**
   - Defines the database models corresponding to each table.
   - Represents the data structure and relationships within the application.
2. **Controllers (routes.py):**
   - Contains functional methods and routes related to the models.
   - Implements the business logic and serves as a bridge between the models and views.
3. **Configuration (config.py):**
   - Stores configurations essential for the application.
   - Centralizes settings for easy management and modification.
4. **API Endpoints (api.py):**
   - Defines endpoints that allow external services or applications to interact with the application
   - Implements CRUD operations on Songs and Albums

5. **Environment Variables (.env):**
    - Safely manages sensitive information such as Secret keys, database URIs, etc.
    - Enhances security by keeping sensitive data separate from the codebase.
6. **Main Application File (app.py):**
    - Serves as the entry point for the application.
    - Imports and orchestrates various components to run the application.
7. **Database (instance folder):**
    - SQLite database is stored in the **instance** folder.
    - Manages data persistence and retrieval.
8. **Static and Templates (static and templates folders):**
    - **static**: Contains static assets like CSS files for styling, lyrics, songs and thumbnails.
    - **templates**: Holds design templates for rendering dynamic content.

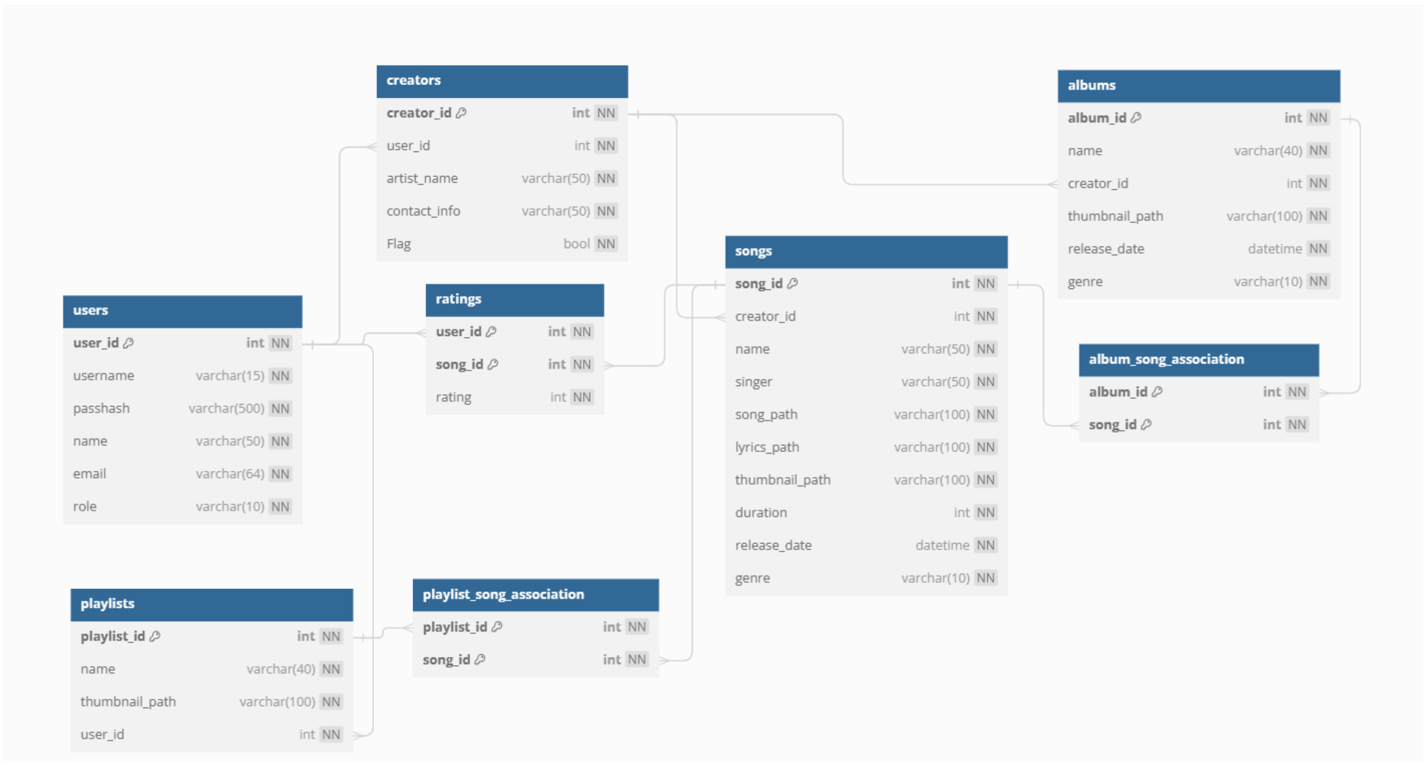
Default features implemented in the application:

- User Login and Admin Login
- Song Management [CRUD], Rate Songs, Play mp3, Read Lyrics, etc.
- Album, Playlist Management [CRUD]
- Admin Dashboard with App Statistics and Performances
- Ability of Admin to blacklist or whitelist a creator, delete a song or album, view song or album
- Search for songs, albums based on Names, Genres, Artists, etc.

Additional features implemented in the application:

- Adding songs to Playlist, Search for Playlist, View info. about creator directly from the album
- Form Validations, Storing password in Hashing manner
- Showing top songs based on ratings of the songs

Database Schema Design:



(*NN denotes Not Null Property)

Relationships Summary:

- **`Users`** and **`Creators`:** *One-to-One relationship***:** each user can have one creator profile, and each creator profile is associated with one user.
- **` Users `** and **`Ratings`:** *One-to-Many relationship***:** each user can have multiple ratings, but each rating is associated with one user.
- **`Users`** and **`Playlists`:** *One-to-Many relationship***:** each user can have multiple playlists, but each playlist is associated with one user.
- **`Creators`** and **`Songs`:** *One-to-Many relationship***:** each creator can be associated with multiple songs, but each song is associated with one creator.
- **`Creators`** and **`Albums`:** *One-to-Many relationship***:** each creator can be associated with multiple albums, but each album is associated with one creator.
- **`Songs`** and **`Albums`:** *Many-to-Many relationship***:** each song can be associated with multiple albums, and each album can have multiple songs.
- **`Songs`** and **`Playlists`:** *Many-to-Many relationship***:** each song can be associated with multiple playlists, and each playlist can have multiple songs.
- **`Songs`** and **`Ratings`:** *One-to-Many relationship***:** each song can have multiple ratings, but each rating is associated with one song.
- `**album_song_association**` and `**playlist_song_association**`: *Many-to-Many relationships* between albums and songs, as well as playlists and songs, facilitated by association tables.
- The above DB Schema is developed w.r.t to the given features of the Application and these are developed one by one after some simulations so that all the requirements are satisfied.

Video Link:

Link:- https://drive.google.com/file/d/1b8nrBWykLo-GDUygiNktctYQSqoPJYYR/view?usp=sharing