Author

Pragati Sethi
21f2001021
21f2001021@ds.study.iitm.ac.in

Description

The music streaming app offers users a unique creator role, allowing them to upload songs and manage albums. Admin oversees app policies, promptly blacklisting creators violating guidelines. Additionally, admin has the authority to flag inappropriate songs and albums, ensuring a safe and enjoyable user experience.

Technologies used

- Flask – Web Framework
- Flask-Wtf forms – For creating forms
- Flask-Login – Managing Login Details
- Matplotlib – For Creating Graphs
- Flask-Restful – For Creating Apis
- Flask-SQLAlchemy – For Creating models for database
- Jinja – Templating Engine
- Bootstrap , CSS - For Styling HTML
- VS Code – Writing Code

DB Schema Design

The database has total 7 tables .

- User : For storing user details
  - Columns : id *(Primary Key)* , name, emaiI_id , iscreator (for assigning creator roles) , is_blacklisted , date_created (when the user registered for the first time)
  - Constraints : email_id should be unique
  - Relationships : It has one to many relationship with following tables :
    - Songs , Album , UserRating , Playlists

    Single user can upload and rate more than one songs or album also can create more than one playlists
- Songs : Stores song details
  - Columns : id *(Primary Key)* , song_title ,language , lyrics, genre ,release_date creator_id *(Foreign Key)* , song url , song_pic , is_flag (for flagging song)
  - Constraints : Except song_pic all the columns are mandatory
  - Relationship : It has one to many relationships with following tables
    - SongInPlaylist , SongInAlbum , SongRatings

    Practically Playlists and Albums can have more then one songs . And single song can have multiple ratings
  - Album : Stores Albums Details
    - Columns : album_id *(Primary Key)* , album_name , is_flag , created_by (stores creator id)
    - Relationship : It has one to many relationship with SongInAlbum table.
  - Playlist : Stores Playlist Detail
    - Columns : playlist_id *(Primary Key)* , playlist_name , is_flag , created_by (stores creator id)
    - Relationship : It has one to many relationship with SongInPlaylist table.

- SongInPlaylist : (playlist_id , song_id) together is the *Composite Primary Key* and also both are *Foreign Key*
    - For maintaining uniqueness constraint
- SongInAlbum : (album_id , song_id) together is the *Composite Primary Key* and also both are *Foreign Key*
    - For maintaining uniqueness constraint
- SongRatings : Storing Ratings
    - Columns : song_id *(Foreign Key , Primary Key)* , user_id *(Foreign Key , Primary Key)* , ratings

API Design

I have developed APIs for various functionalities, including uploading, editing, displaying, and deleting songs. Additionally, I created APIs for retrieving songs based on a specific creator ID, managing albums (creating, deleting, and displaying all albums), and fetching song details within a particular album using its ID. Furthermore, I implemented APIs for playlist management, encompassing playlist creation, display, editing, and deletion. To enhance user experience, I included an API for retrieving all songs associated with a given playlist id.

APIs are stored separately and called within **app.py** to integrate with Flask.

Architecture and Features

1. Project Structure:
    - The project folder comprises four main directories: application, static, templates, and instance.
    - Additionally, there is a Python file named "app.py," serving as the main executable for the application, and a requirements.txt file (utilized to manage project dependencies).
2. Application Folder:
    - Subdivided into apis and data folders.
    - apis folder houses all API implementations.
    - data folder contains models
3. Routes and Controllers:
    - The routes.py file within the application folder holds all the controller logic for the application.
4. Static Folder:
    - Contains subfolders such as songs, song_pics, and graphs to store static assets.
5. Templates Folder:
    - Hosts all HTML files for the application's user interface.

The application has a user dashboard showcasing songs, albums, and playlists, along with genre-specific song listings.
Users can actively manage playlists, edit profiles and versatile search options based on song title, album name, and creator name.
Creators have specialized tools, including a dashboard for album management, song uploads.
Admin have admin dashboard with application statistics and graphical representations of app performance. It also has functionality to flagged songs and albums, and also blacklist and whitelist the creator.
Key functionalities like Song CRUD, Playlist CRUD, and Album CRD are executed through APIs.

Video

https://drive.google.com/file/d/1O5u0_d2tDEr7kI1KE-wGB2U5LoAaYMCi/view?usp=drive_link