Author

P Sai Midhun Reddy
22f1001766
22f1001766@ds.study.iitm.ac.in
B.Tech Third year Computer Science student from MGIT, Hyderabad. Currently pursing Diploma in Programming.

Description

MELOMIX is a multi-user music streaming app like spotify, gaana. It is used for reading & listening songs. A user can register as creator to upload songs & albums. Users can rate and report songs, playlists and albums.

Technologies Used

- **Flask**: for basic backend Implementation.
- **Flask_sqlalchemy**: for implementing Database.
- **Flask_Login**: for implementing the login functionality, user session management
- **datetime**: to storing the date and time (date of creation).
- **pytz:** for date-time conversion
- **flash**: to show alerts
- **Flask_Login**: for implementing the login functionality
- **werkzeug.security**: for hashing the passwords
- **werkzeug.utils:** secure_filename to get
- **Flask-Restful**: to create Apis
- **matplotlib:** for ploting graphs
- Some inbuilt libraries like jinja2, render_template, redirect, and url_for displaying HTML content.

API Design

There are 3 APIs

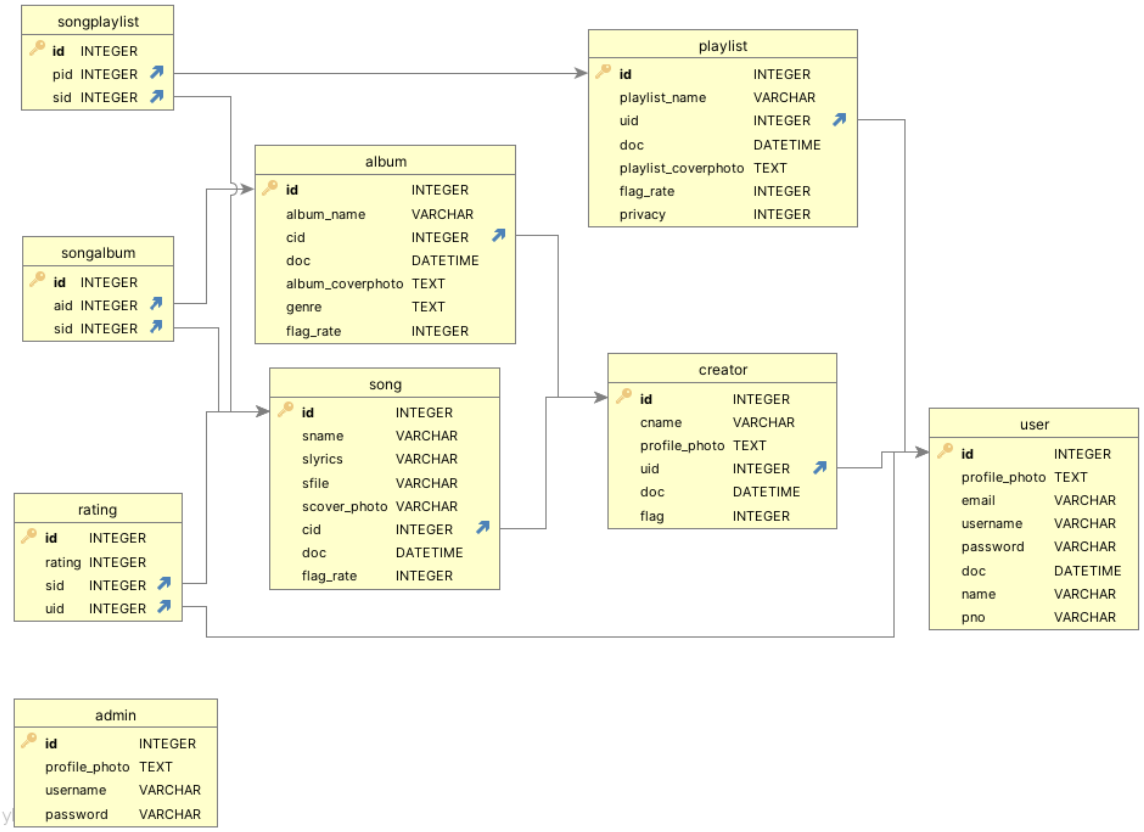
1. **UserAPI :** It has two end points
    - **/api/users :** With this endpoint, we can read all users details in database, creating user.
    - **/api/user/<int:user_id> :** With this endpoint, we can get user with input id, update user with respective input id, delete user with respective input id

2. **SongAPI :** It has four endpoints
    - **/api/songs :** With this endpoint, we can read all songs details in database.
    - **/api/song/<int:song_id> :** With this endpoint, we can read song with respective input id
    - **/api/creator/<int:creator_id>/songs:** With this endpoint, we can read all songs of respective creator and also create songs under respective creator
    - **/api/creator/<int:creator_id>/song/<int:song_id>:** With this endpoint, we can get respective song of respective creator. We can also update and delete respective song associated with respective creator with this endpoint.

3. **PlaylistAPI :** It has four endpoints
    - **/api/playlists:** With this endpoint, we can read all playlists in database
    - **/api/playlist/<int:playlist_id>:** With this endpoint, we can read playlist with respective input id
    - **/api/user/<int:user_id>/playlists:** With this endpoint, we can read all playlists of respective user and also create playlists under respective user

- **/api/user/<int:user_id>/playlist/<int:playlist_id>:** With this endpoint, we can get respective playlist of respective user. We can also update and delete respective playlist associated with respective user with this endpoint.

DataBase Schema Design



Architecture & Features

- There are 2 controllers
    1. **auth**: It is used for authorization purpose
    2. **view**: It is used for all other purpose like viewing ,creating and searching songs, playlists, albums .
- There are 2 folders
    1. **static**: It contains CSS file and images
    2. **templates**: It contains all HTML templates used in Project.
- **login & sign-up system**: Here user should fill in details for creating a new account and after that user will be able to do login.
- **User Home Page:** All songs, playlists & albums will be displayed in this page
- **User Profile Page:** Page to view user's details and update some details
- **Creator - register page:** For registering as a creator.
- **Playlist Page:** To view songs in playlist, reporting playlist
- **Song Page:** To read lyrics, play and rate a song
- **Album Page:** To view songs in Album, reporting album
- **Search:** To search songs, playlists, albums based on names, created by and genres.
- **Admin's Dashboard:** Contains statistics of the App, song/playlist/album management, creator management.
- **APIs:** Three APIs User, Song, Playlist with CRUD functionalities

Video :

https://drive.google.com/file/d/1DLRsSezx_25cbxpehmcKw-Urq78NQqKU/view?usp=drive_link