Music Streaming App
(AppDev1- Project)

Name : Naveen Kumawat

Roll No. : 21f1001711

Student Email : 21f1001711@ds.study.iitm.ac.in

Description:

This app is a multi-user web app for listening to and uploading music.
To create this app, I followed the app wireframe and the guidelines provided by appdev1 instructors and the following steps:-

- Creating the database schema of the app and tables using flask-sqlalchemy.
- Creating the flask app instance and html pages using bootstrap templates.
- Creating all the required routes to link the app to the database. A simple login framework with hashed passwords stored in the database.
- Adding css styling to the web pages.

Frameworks Used:

- Flask - this web application is built on flask.
- Jinja2 Templates - the web pages use jinja2 syntax to generate dynamic content.
- Bootstrap - for templates of the web pages.
- SQLite3 - to create the database structure for the app.
- Flask-SQLAlchemy - to create and manage the relational database for the app.
- Matplotlib - to plot the app statistics graphs for the admin dashboard.

Database:

- Database models/tables for the app are created using flask-sqlalchemy.
- There are 5 Tables used in the database: User, Song, Album, Playlist, Playlist_Song_Table.
- Song and Album have many to one relationship.
- Playlist and Song have many to many relationships. This relationship data is stored in Playlist_Song_Table.
- User table is used to differentiate between user, admin and creator.

System Design:

- This web app follows MVC architecture style.
- Templates Folder - stores all the html pages of the app.
- Instance Folder - stores the database of the app.
- Static Folder - stores all the graphs and has an audio folder which stores the mp3 files.
- App.py - the code for creating the instance of the flask app.
- App_data.py - the code for creating the pre saved data of the app.
- Models.py - the code for creating database tables.
- Routes.py - the code for all the routes and helper functions of the app.
- Requirements.txt - stores names of all the dependencies along with their version number.

Features Implemented:

- Separate login form for users, creators and admin.
- Proper flash/alert messages for the tasks performed.
- Admin dashboard with app statistics on users, creators, songs, albums, most played/liked songs graphs.
- Admin can manage songs, albums and blacklist/delete creators.
- Admin/User can search songs/albums/artists/genres based on their type.
- Users are recommended songs based on the number of likes.
- Users can play songs, read lyrics, like/dislike songs, create/edit/delete playlists.
- Users can register to become creators.
- Users can change their password to a new one.
- Creators can upload/edit/delete songs and albums.
- Creator dashboard with total uploads, total likes received etc.

To run the app:

Unzip the project folder, install all the dependencies from requirements.txt and run the app.py file.

Presentation Video Link:

https://drive.google.com/file/d/1wlXZETXtlbeFRvRqTvqa1BoHLGZmMlfX/view?usp=sharing