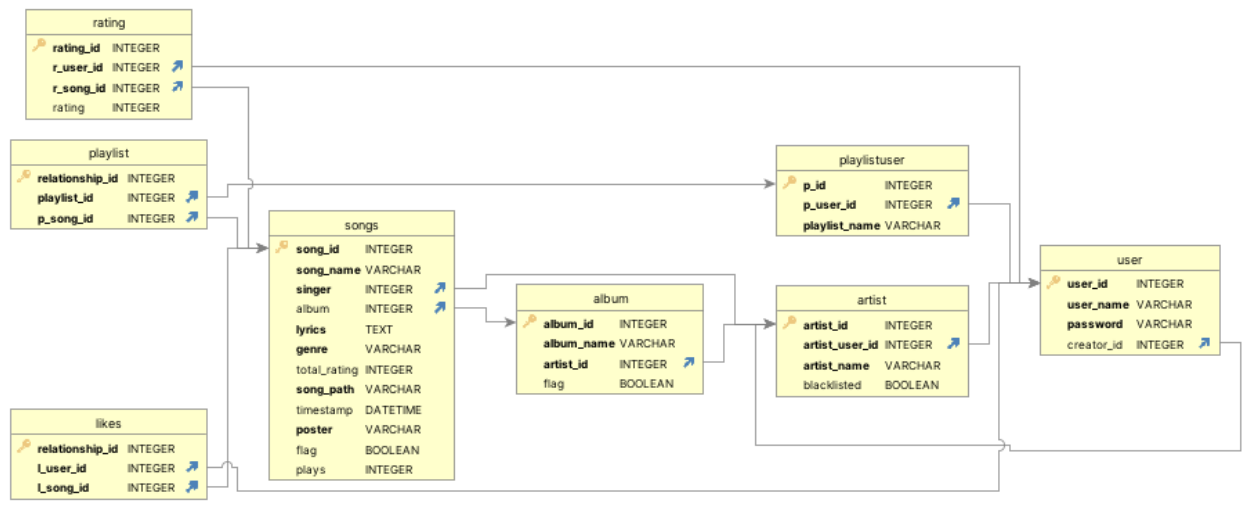
Project Report- Modern Application Development I

Author
Rashika Vinod Kukreja
22f2001412
22f2001412@ds.study.iitm.ac.in
I am a student, currently pursuing a Bachelor of Arts (Mathematics).

Description
The objective of the project is to design a music streaming application which allows users to listen to songs, read lyrics, rate songs, create playlists, etc. The users on the app can register as creators which gives them additional functionality to upload songs, update songs, create albums. The admin of the application is shown statistics about the working of the application and also has the ability to flag songs, albums and creators who violate the company policies.

Technologies Used
The main backend framework that is used to design the application is Flask, a lightweight and flexible web-app framework in python. Flask Login is used for implementing the login framework of the application. The database management system used is SQLite that stores the data in a local file instead of a cloud server whereas SQLALchemy is used as an object relational mapper which enables interaction with the database in python language. Flask RESTful framework is used for developing the API for the application. For frontend design bootstrap is used as the HTML, CSS and Javascript framework.

Database Schema Design



The above diagram describes the structure of the database schema for the application. The primary tables include Users, Songs, Artist and Album. Rating, Likes, Playlist and Playlistuser are relationships defined on the primary tables.

1. User: User table stores the user_id(primary key), user_name and password of the user. The creator_id references the artist_id(Table: Artist) which is used to verify whether a particular user is a creator.

2. Songs: The songs table stores various attributes describing songs like song_id(primary key), song_name, lyrics, genre, number of plays of the song, etc. The attributes singer and album reference artist_id (Table: Artist) and album_id (Table:Album) respectively. The total_rating attribute is calculated as the average of the entries in the Rating table where r_song_id==song_id.
3. Artist: The Artist table stores the artist_id(primary key) and the name of the artist (artist_name). The artist_user_id references the user_id(Table: User) as all artist/creators are general users. Blacklisted is a boolean value which defaults to false but can be changed to true by the admin.
4. Album: The Album table stores the albums created by a particular artist. Album_id(primary key) and album_name are the attributes whereas artist_id specifies the artist of the album. Album.songs can be used to get all the instances of song objects whose song.album==album_id.
5. Playlistuser: It is a table that stores the playlists created by users. p_id(primary key) and playlist_name are the attributes whereas p_user_id references user_id(Table:User). The playlists of a particular user can be accessed as user.playlists which is a list of objects of Playlistuser class.
6. Playlist: This table stores the songs of a particular playlist. It is a relationship between the table Playlistuser and songs. It stores the song_id(p_song_id) and the playlist_id of the playlist the song is put into.
7. Rating: Rating table stores the ratings given by users to various songs.
8. Likes: Likes table is a relationship between songs and users which stores the songs liked by users.

API Design

The API includes 4 resources. Resource SongAPI consists of create.read,update, delete operations on the Songs table. Resource PlaylistApi allows viewing all playlists, creating, updating and deleting a playlist. Resource UserApi allows viewing all users registered on the app. Lastly, AlbumApi allows viewing, creating, updating and deleting the albums.

Architecture and Features

The project folder consists of all the components required for making the application work. Models.py consists of the models required for the application. App.py includes the code for controllers and the API resources. The HTML templates are stored in the templates folder. The audio files and the posters of the songs are stored in the static folder. The static folder also has a folder named ‘admin’ which has the graph and the pie chart displayed on the admin dashboard. The database file will be stored in the instance folder on initialization of the app. ‘Readme’ is a text file that contains basic directions on how to operate the application.

The core features of the application include playing a song, creating a playlist, rating songs, search, etc. An added feature here is that a user can also like songs which will be displayed in the ‘liked songs’ section. Features for a creator include uploading a song, updating a song, creating an album, etc. The admin gets information about the key statistics and is also given the functionality of blacklisting creators whereby the creator will no longer be able to access any creator functionalities and their content will be hidden from other users. Similarly, the admin can flag songs and albums which will give them a tag of ‘explicit content’ to inform the users of flagged content. Among the recommended features, API is implemented as mentioned above.

Video Link

https://drive.google.com/file/d/1mwxjLfhG4LDkg9f06H-a9raZGd7CWY9T/view?usp=drive_link