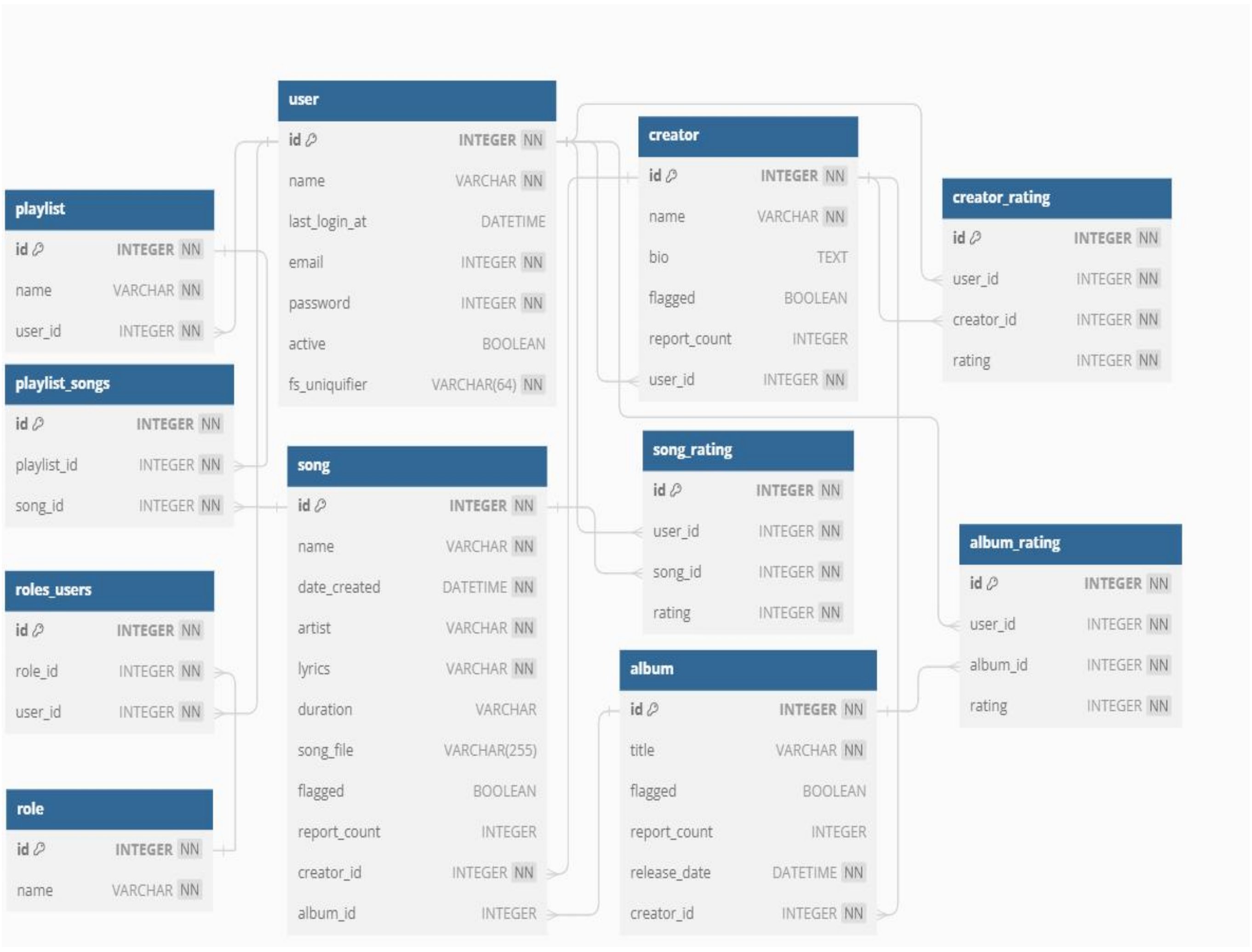
Author:
Arjita Singh Kushwaha
21f3000507
21f3000507@ds.study.iitm.ac.in

Description:
A music streaming app that allows users to listen to songs, create and manage playlists, and rate songs. The app supports multiple user roles, including normal users, creators, and admins.

Technologies Used:
- Vue CLI: For UI
- Bootstrap: For aesthetics
- Flask-CORS: Facilitates management of cross-origin requests from diverse domains.
- Flask RestFul API: Used to develop the RESTful API for the app
- Jinja2: Used for rendering templates for sending emails
- SQLite: Used for data storage
- Flask SQLAlchemy : Used as an ORM (Object-Relational Mapping) tool to interact with the Database
- Flask Celery: Used for asynchronous background jobs at the backend.
- Redis: Used as an in-memory database for the API cache and as a message broker for celery.
- Flask Caching : Used for caching API outputs and increasing performance.

Database Schema Design:



API Design:

- **User Management API**: Register,Login, Logout, Get User Info Feature.
- **Album Management API**: CRUD implementation of Albums.
- **Song Management API**: CRUD implementation of Songs.
- **Creator Management API**: to manage creators whose songs/albums are available on the platform.
- **Playlist Management API**: CRUD implementation of Playlist
- **Song/album/creator Rating API**: Enable users to rate songs/albums/creators on the platform.
- **Cache Management**: Implement caching mechanisms to improve the performance of frequently accessed data, such as popular songs, playlists, etc.
- **Asynchronous Task API:** Implement asynchronous tasks for background processing, such as monthly generating reports and sending daily remainders.

Architecture and Features:

- `\backend`: contains application, app.py, worker.py
- `\backend\application`: contains models.py, jobs.py, views.py
- `\backend\application\static\songs`: contains uploaded songs file
- `\backend\application\models.py`
- `\backend\application\views.py`
- `\backend\application\jobs.py`
- `\frontend`: Contains Vue templates and JavaScript files.
- `\frontend\src`: App.vue, stores, routers
- `\frontend\src\components`: Vue Components.
- `\frontend\src\views`: Vue Views.

Features:

User Management:
Register: Allows users to create new accounts on the platform, providing basic information such as username, email, and password.
Login: Enables registered users to log in to their accounts securely using their credentials.
Logout: Allows users to securely log out of their accounts to protect their privacy.

Album Management :
CRUD Operations: Implements Create, Read, Update, and Delete operations for managing albums on the platform.

Song Management:
CRUD Operations: Provides endpoints for creating, retrieving, updating, and deleting individual songs in the database.

Creator ManagementI:
CRUD Operations: Facilitates the management of creators who contribute songs and albums to the platform.
Flagging and Reporting: Enables users to flag or report creators for inappropriate content, spam, or other violations.

Playlist Management API:
CRUD Operations: Allows users to create, view, update, and delete playlists containing their favorite songs.

Song/Album/Creator Rating :
User Ratings: Allows users to rate songs, albums, and creators based on their preferences and experiences.
Average Rating Calculation: Calculates the average rating for each song, album, and creator based on user ratings.

Asynchronous Task :
Background Jobs: Executes background tasks asynchronously, such as sending monthly reports, sending daily reminders.

Role-Based Access Control:
Admin Role: Grants administrative privileges to manage users and content.
Creator Role: Provides creators with permissions to upload, edit, and manage their songs and albums.
User Role: Standard role for regular users with access to basic features such as listening to music, creating playlists, and rating content.

Video:
https://drive.google.com/file/d/1gSxvfe4Sienql18L04yDnVK_PCUlzLEl/view?usp=sharing