Author

Kabir Jamadar
Roll number: 22f3001138
Gmail:22f3001138@ds.study.iitm.ac.in
Enthusiastic data science student at IIT Madras, focusing on machine learning. Eager to learn and leverage artificial intelligence to solve challenges.

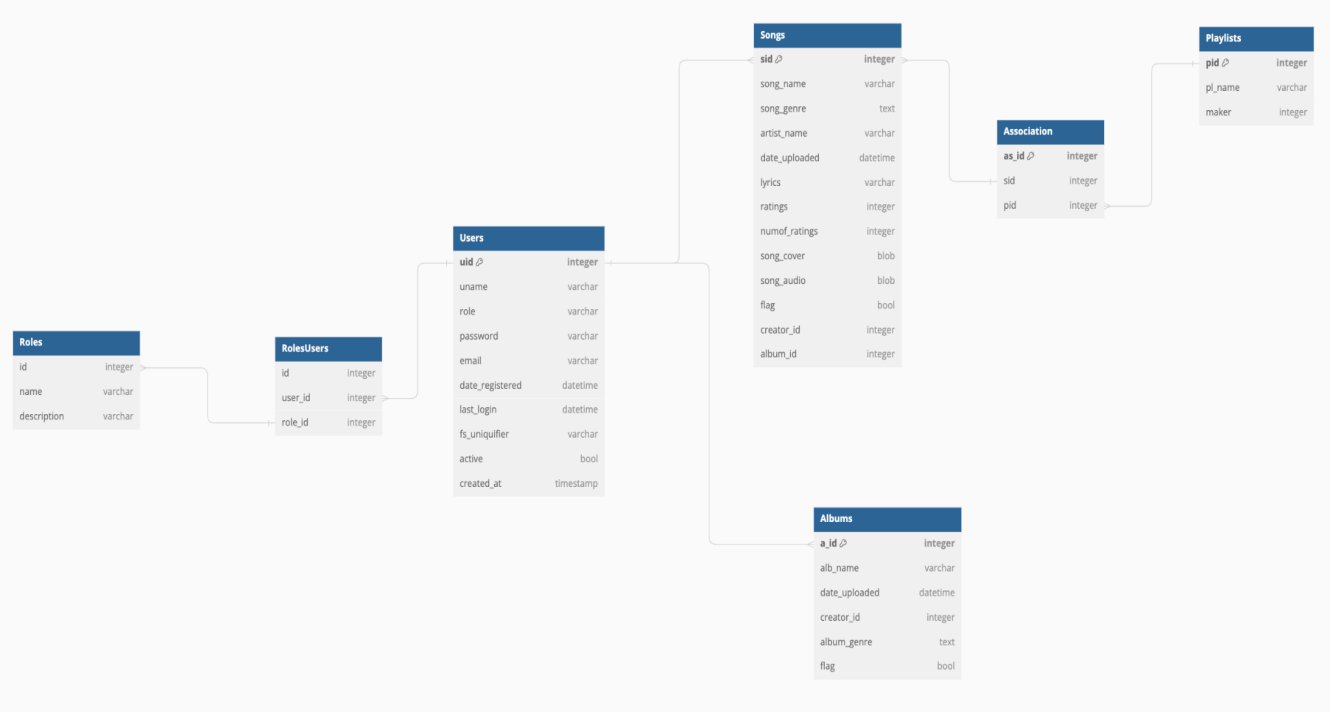
Description

A full-fledged music streaming platform built with the powerful combination of Flask for the backend, SQLAlchemy as an ORM and Vue for the frontend. Users can listen to songs,albums, curate playlists, read lyrics and rate songs. For creators the platform, enables them to upload songs and albums, and enrich the music collection with lyrics. The integration of Flask-RESTful api facilitates smooth communication between the frontend and the backend.Admin can view the statistics of the application, flag songs, Albums, activate/deactivate users. Additionally, Redis & Celery work together behind the scenes to automate tasks like sending daily reminders and generating monthly reports, ensuring a smooth and efficient user experience.

Technologies used

Flask, a micro framework to develop web applications with python.
Flask-SQLAlchemy, an ORM(Object-relational mapper) that allows users to work with databases using python. SQLAlchemy is used to create and manipulate databases.
Flask-security, for authentication and session management. Handles common tasks like logging In, logging out and setting authentication-tokens, as well as securing the application.
Bootstrap, a frontend toolkit, helps create responsive , user friendly design.
Flask-Restful: a Python extension that simplifies the development of Restful APIs using Flask.
Flask-RBAC, provides a Role based Access Control for the application.
Vue, for building user interfaces, builds on top of Html, Javascript, Css for a declarative and component based programming model.
Celery, is a powerful task queue that can be used for simple background tasks as well as complex multi-stage programs and schedules
Redis, an open source in-memory database used as a message-broker.

DB Schema Design



The database schema consists of 7 tables (Users,Roles,Roles_Users, Songs, Albums, Playlist,Association):

Relationships:

(1)One to many between Roles and Users through the Roles_Users table.

(2)One to many between Users and Songs through the creator_id attribute.

(3)One to many between Users and Playlists through the maker attribute.

(4)One to many between Users and Albums through the creator_id attribute.

(5) One to many between Albums and Songs through the song attribute.

(6)Many to Many between Playlists and Songs through the Association table.

The application offers distinct user roles using Flask RBAC, an admin, creators and normal users.

Api Design:

The api directory contains flask-Restful api applications for selecting,updating,adding,deleting Users, Songs, Albums, Playlists. It employs JSON serialization for outputs.

Architecture:

The project is organized as follows:

/Code

/frontend

/src

/components:

/views: contains the vue template views for the application

/store: contains vue-x store template

/router: contains vue-router template

/backend

main.py: to run the application

/instance

/app

__init__.py: to initialize libraries

config.py: contains the configurations

models.py: contains SQLAlchemy models for the applicaiton

views.py: routes for the application

sec.py

Mail_service.py: contains the celery beat functions

tasks.py: contains the celery tasks

instances.py

worker.py: initialization of the workers

/apifn:

api_songs.py : contains the restful routes for songs

api_albums.py: contains the restful routes for albums

api_playlists.py: contains the restful routes for playlists

Api_users.py: contains the restful routes for users

Project Video:

mad2projectvideo.mov