Project Report

Name: Ishika Goel
Roll Number: 21f1006093
Email ID: 21f1006093@ds.study.iitm.ac.in

About Me:

I am an Economic graduate from M.O.P. Vaishnav College for women,Chennai. I have no background in programming apart from the application we were asked to make as part of Modern Application Development 1. I have tried to make the application with the best of my knowledge and ability.

Description

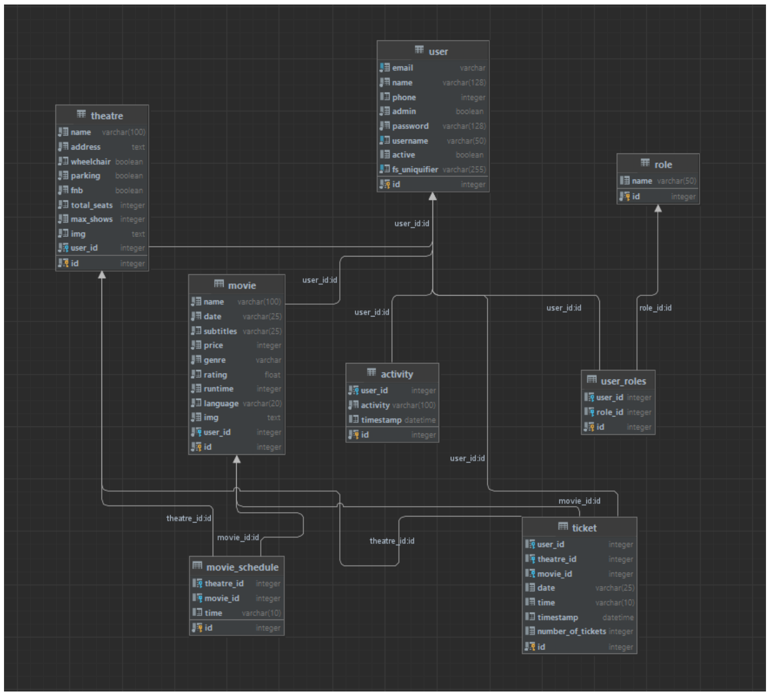
The aim of this project was to build a ticket booking app akin to BookMyShow. The app would differentiate between an admin and a regular user, where the admin would have special permission to add a theater and/or a movie to the app. The booking process will handle housefull scenarios, where it would not allow the user to do so in that case. The user also obtains monthly entertainment reports that depict their monthly bookings and their preferences in movies. It also has an export option for the admin that allows them to extract information of each theater and the movies that run in it.

Technologies used

A variety of Python extensions and modules were used to build the backend of the app. Flask, Flask-RESTful, Flask-CORS, Flask-Security, SQLAlchemy, Matplotlib, Celery and Flask-Cache are a few to name. A complete list of the packages and their versions can be found in the requirements.txt file in the Project directory. Authentication was token based. The frontend was built using Vue-CLI, and is served on another port. JS was used at the frontend to fetch data from the backend. The entire app was built locally on PyCharm, and DB Browser for SQLite3 was used to build the database. Styling and aesthetics were done using CSS from Bootstrap for each of the components. MailHog was used to set up a fake SMTP server to send emails. Redis was used as both the message broker for Celery and to cache the APIs.

DB Schema Design

- There are 8 tables in the database: User, Role, UserRoles, Ticket, Theater, Movie, Movie_Schedule and Activity.
- User keeps track of all the users who have logged in to the app.
- Theater keeps track of all the theaters available in the app.
- Movie keeps track of all the movies available in the app.
- Ticket keeps track of all the tickets booked by different users.
- Roles has the different roles available to the users. In this case, it is only ‘Admin’.
- UserRoles keeps track of the different roles assigned to different users.
- MovieSchedule keeps track of the timings of the different shows at each theater
- Activity keeps track of different activities logged in the app.



All the database schemas are stored in models.py file. It was a very intuitive, simple design for the database. The foreign key constraints had to be imposed to ensure that multiple trackers were still identified by their users, and similarly for the logs by their trackers. In addition to these, we also set up decorator classes for the APIs using Marshmallow to serialize SQLAlchemy objects.

Architecture and Features

The file hierarchy of the project is very similar to what was explained by Thejesh Sir. There is a config.py file for the configuration settings, an init.py file for the initialization of the app, apis, cache, marshmallow and login features. There is an api.py file that takes care of all the APIs. A separate file called database.py was made to initialize the database via SQLAlchemy, and as mentioned earlier, models.py has all the data related to the database schema. To handle the celery tasks, there is a workers.py that defines the worker class, and tasks.py along with the definitions of the various tasks. These 7 files along with the database itself are stored in a folder called backend. The vue components are in a folder called components. The templates folder contains the various HTML templates needed for the body of the emails and the pdfs while the assets folder contains the pdfs, csvs, post images and graphs in their respective subdirectories. Along with main.js, App.vue, routers.js and store.js, all the above are present in the folder src within the folder frontend. Along with these folders, there is an API.yaml, README.md, requirements.txt file, Welcome.pdf and main.py in the root directory. There are also 4 shell scripts, information about which are mentioned in README.md.

Main Features of the app

The main features of the application include creating users, editing or modifying profile by the user. Once the user signs up, he/she will receive a mail welcoming them to use the app. Furthermore, they are now free to browse through movies and theatres. Movies can be accessed by searching based on the name, genre, rating, etc while the theatre can be accessed by searching its name or location. The home view of both the movies and theatre display their respective photos and details added by the admin. The admin has a slightly varying portal than the user that allows him to add, edit or delete a movie or theatre. The admin has the options to validate whether a respective theatre has parking space, allows food and beverages or is friendly for disabled users. In the app default options are taken to be as no unless otherwise chosen like in the case of signing up as admin or validating properties of theatre. There is an export button in the admin's theatre dashboard that provides him details accordingly in csv format. Monthly reports would also be sent comparing the last 2 months; activities of the user and admin. Admin can act as a user but the user can't. There are reminders sent every morning to urge the user to visit the app. There is also caching involved.

Demonstration Video

<Video Demonstration of Ticket Booking App>