Author

Name : - Yazad Mehernosh Pardiwala
Roll Number :- 21F3001724
Student email id :- 21f3001724@student.onlinedegree.iitm.ac.in
I am pursuing this course alongside a full time BMS degree from Mithibai college Mumbai, where I am in my 2nd year. I fully intend to complete till the BSc level & will almost certainly pursue the full BS degree as well.

Description

I have made the ticket show application; as such I understood my requirements were to create an app to allow users to find and book tickets to various shows, & admins to create, modify & delete these shows, theatres, etc., all while making use of the specific advanced (relative to MAD1) technologies taught in the course.

Technologies Used

- Flask – To create the app, run the app, & route functions to the various urls.
- Flask_sqlalchemy – To define the models for my database, connect to the database, & query it.
- Sqlite3 – For the actual database itself.
- M-search – To enable user search functionality on the database tables
- Vue-JS – For the entire front-end application.
- Celery & Redis – for task based functionality (partially)
- Json Web Tokens (JWTs) for login & security

Broad System Design

I implemented my app in a split architecture form. That is to say, the front end was its own entirely separate app made with vue cli & running on an entirely different server & port as compared to the backend. Communication between the two was handled via fetch requests, and said communications were authenticated with JWTs.

On the backend, being pressed for time, I stuck with traditional routes rather than designing an API, determining that if time (by some odd miracle) was available once I was done with higher priority tasks, I would go back and re-configure it to be an API.

DB Schema/Models used

- USER –
  - UID,
  - U_NAME,
  - PWORD,
  - ADMIN,
  - VISITED,
  - EMAIL
- THEATRE –
  - TID,
  - TNAME,
  - CAPACITY,
  - LOCATION,
  - TIMAGE_ID
- SHOW –
  - SID,
  - SNAME,
  - PRICE,
  - RATING_AGG,
  - SIMAGE_ID,

    - S_DESCR,
    - S_TAGS
- RATINGS –
    - RID,
    - SID,
    - UID,
    - RATING
- BOOKINGS –
    - BID,
    - AID,
    - UID,
    - TICKETS,
    - SID,
    - TID
- AIRINGS –
    - AID,
    - SID,
    - TID,
    - STRT_TIME,
    - STOP_TIME,
    - DATE,
    - VACANCIES
- CREATIONS –
    - CID,
    - UID,
    - SID,
    - AID,
    - TID

My schema had some flaws which I recognized in time; I shall go through them now.
The SHOW should have had a column DURATION, since, while I liked the idea of theatre owners being able to set an AIRING to go on for longer than the SHOW's run time (since they'd know best when the theatre would need to be cleaned, how long it takes in their theatre for the audience to leave, etc.), a theatre owner should not be able to say that Lord of the Rings (a 3 hour movie) is airing for 3:00 to 4:30, for example.

The AIRINGS table should have had the PRICE column, since depending on the theatre, time of day, age of the movie, etc, prices for the same movie would vary. This was an oversight on my part.

Features implemented

Most of the core app-related functionality (in terms of things the app should enable the user/admin to do) is present & accounted for.
The only things not accounted for are the batch jobs with celery (though I very nearly had/have these working, were it not for a few bugs I cannot solve) and the recommended and optional criteria (there is some very minimal styling).

Video
https://drive.google.com/file/d/1Qki61xKX0_2SBbn_X4n3QcbKAZhiYaxe/view?usp=sharing