Author

Shubham
21f1007067
21f1007067@ds.study.iitm.ac.in

Description

In this web development project , A movie ticket booking website is built, in this a admin is created when the database is initialised. Admin can create, delete, update movies, theatres. Users can login and book tickets for movies.

Technologies used

VueJS - Used for User Interface

Flask RestFul API - Used to develop the RESTful API for the app

Jinja2 - Used for rendering templates for sending emails

Bootstrap - Used for HTML and CSS styling

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.

Flask Caching - Used for caching API outputs and increasing performance.

Redis - Used as an in-memory database for the API cache and as a message broker for celery.

DB Schema Design

Database has several models : **User**, **Theater**, **Movie**, **UserTheaterRating**, **TheaterMovie**, **Booking**, **Dynamic**.

Structure and details of the columns :

User: Table stores information about users, as user_id, user_mail, password(hashed), admin, active, fs_uniquifier.

Theater: Table stores information about theaters as theater_id, theater_name, theater_place, theater_location, theater_capacity.
Movie: Table stores information about movies as movie_id, movie_name, movie_tag, movie_language, movie_duration, movie_description, movie_image_path.
UserTheaterRating: Table stores information about theater ratings as user_id, theater_id, rating.
TheaterMovie: Table stores information about theater movie relationship as theater_movie_id, theater_id, movie_id, ticket_price, timing.
Booking: Table stores information about bookings made by user as booking_id, user_id, theater_movie_id, no_of_tickets, total_paid, booking_time.
Dynamic: Table stores information about dynamic info about seat left and current price as theater_movie_id, seat_left, current_price.

API Design

The API design for this project follows a RESTful architecture, which means that it uses HTTP requests to access and manipulate resources identified by URLs. The endpoints are grouped into several categories, each corresponding to a specific feature of the application.

Authentication Endpoints:

- POST /api/login : use to login users.
- POST /api/register : use to register users.
- POST /api/token/refresh : use to refresh token in case of expiration of token.
- POST /api/forget_pass : to rest user password.

Movie Endpoints:

- *GET /api/movie* **:** get all the movies.
- *POST /api/movie* **:** to add a new movie.
- *GET /api/movie/<int:movie_id>* **:** to get one movie with movie id.
- *PUT /api/movie/<int:movie_id>* **:** to update one movie with movie id.
- *DELETE /api/movie/<int:movie_id>* **:** to delete one movie with movie id.

Theater Endpoints:

- *GET /api/theater* **:** to get all theatres
- *POST /api/theater* **:** to add a new theatre.
- *GET /api/theater/<int:theater_id>* **:** to get specific theatre.
- *PUT /api/theater/<int:theater_id>* **:** to update theatre with theater_id.
- *DELETE /api/theater/<int:theater_id>* **:** to delete theatre with theater_id.
- *POST /api/user/rating/theater/<int:theater_id>* **:** to rate theatre by user.
- *GET /api/rating/theater/<int:theater_id>* **:** to get rating of theatre.

TheaterMovie Endpoints:

- *POST /api/link_theater_movie/<int:theater_id>/<int:movie_id>* **:** to link theater and movies.
- *GET /api/theater_movie* **:** to get all movies linked with theaters
- */api/dlt/theater_movie/<int:id>* **:** to delete specific relationship of theater and movie

Ticket Booking Endpoints:

- *POST /api/book_ticket/booking/<int: theater_movie_id> : to book a ticket*
- *GET /api/user/booking :* to get all the booking of user.

Search Endpoints:

- *GET /api/search/filter : To get all the features*
- *GET /api/search/movie/<int:movie_id> : to get searched movie*
- *GET /api/search/theater/<int:theater_id> : to get searched theater*

Additional Endpoints:

- *POST /api/export_csv : for admin to get csv file of theater details on mail.(admin triggered)*

Architecture and Features

1. **User Signup and Login**
   - ☑ Form for username and password (both login and signup)
   - ☑ Use Flask Security or JWT based Token Based Authentication only
   - ☑ Suitable model for user
2. **Admin Login**
   - ☑ Form for username and password (can be same as normal users)
   - ☑ Add an admin user whenever a new database is created
   - ☑ The app should differentiate between an admin and a normal user
3. **Theater Management**
   - ☑ Create a new theatre
     - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
   - ☑ Edit a theatre
     - Change title/caption or image
   - ☑ Remove a theatre
     - With a confirmation from the admin
4. **Movie Management**
   - ☑ Create a new show
     - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
   - ☑ Edit a show
     - Change title/caption or image
   - ☑ Remove a show
     - With a confirmation from the admin
   - ☑ Allocate theatres while creating shows
     - Done separately to achieve different prices for the same move at diff. theatres
5. **Search for Shows/Theater**
   - ☑ Ability to search theatres based on location preference
   - ☑ Ability to search movies based on tags, rating etc.
     - Rating search done using filters
   - ☑ Basic home view for a theatre
6. **Book Show Tickets**
   - ☑ List the shows available for a given timeframe to the users
   - ☑ Ability to book multiple tickets for a show at a given theatre
   - ☑ Ability to stop taking bookings in case of a houseful.
7. **Daily Reminder Jobs**

- ☑ Scheduled Job - Daily reminders on Google Chat using webhook or SMS or Email
    - In the evening, every day
    - Check if the user has not visited/booked anything
    - If yes, then send the alert asking them to visit/book

8. **Scheduled Job- Monthly Entertainment Report**
    - ☑ Scheduled Job - Monthly Entertainment Report
        - Come Up with a monthly progress report in HTML (email)
        - The entertainment review report can consist of bookings made by a user in a given month, shows seen, ratings for the shows etc.
        - On the first day of the month
            - Start a job
            - Create a report
            - Send it as email
9. **User Triggered Async job**
    - ☑ User Triggered Async Job - Export as CSV
        - Come up with an export CSV format for theatres
        - The export is meant for a single theatre (at a time) to export details like number of shows, bookings, rating etc.
        - Have a dashboard where the user can export
        - Trigger a batch job, send an alert once done
10. **Performance and Caching**
    - ☑ Add caching where required to increase the performance
    - ☑ Add cache expiry
    - ☑ API Performance
11. **Responsive UI**
    - ☑ Unified UI that works across devices
    - ☑ Add to desktop feature
12. **Well designed PDF reports**
    - ☐ Done with sending monthly entertainment reports.

Additional Features

- ☑ Dynamic pricing for ticket
- ☑ Use Movie poster
- ☑ User has profile page where user can see bookings

Video

https://drive.google.com/file/d/1WwkMUohtzDbcVIWGi_ivrJSwk-RLQ2a-/view?usp=sharing