Ticket Show

Author:

Raghav Rao Ghanathe

22f1000926

22f1000926@ds.study.iitm.ac.in

I am currently pursuing B.Tech in CSE from KMIT, Hyderabad. I am very much interested in Data Science and looking forward to learning more.

Description:

The Ticket Booking System is a movie reservation platform with user and admin login features. Admins manage movies, theatres, and exports, while users browse, book tickets, and view booking history. The frontend employs Vue components and routing, while the backend utilizes Flask, SQLAlchemy, and Celery for asynchronous tasks. This includes monthly reports, daily reminders, CSV theatre exports, and caching for optimization.

Technologies used:

- Flask 2.2.2 - For app routing, template rendering, and redirecting.
- Flask-SQL Alchemy - For connecting the SQLite database as a server with the flask application.
- Celery [Redis] – For caching and scheduling asynchronous tasks.
- Flask_JWT_Extended – For Tokenization
- VueJS – For frontend tasks
- Bootstrap – For Styling

DB Schema Design Tables

1. Users

- userID – Unique ID for user identification - Integer, PK, Unique, Auto Increment, Not Null
- userName – Username for User - String, Unique, Not Null
- password – Password for User – String, Not Null
- lastAct – Last Activity of User - String

2. Movies

- movieID - Unique ID for Movie - Integer, PK, Unique, Auto Increment, Not Null
- movieName – Name of the Movie – String, Not Null
- movieLang – Movie language – String
- types – Type of the movie – String
- genre – Category of the movie – String
- theatreID – FK Theatres.theatreID (On Delete Cascade) – to identify theatre for the movie

3. Theatres

- theatreID - Unique ID for Theatre - Integer, PK, Unique, Auto Increment, Not Null

- theatreName – Name of the Theatre – String, Not Null
- location – Theatre location – String

4. MovieDates

- movieDateID – Unique ID movie date – Integer, PK, Unique, Auto Increment, Not Null
- movieDate – Movie date – String, Not Null
- movieID - FK Movies.movieID (On Delete Cascade) – to identify movie for a date.

5. MovieTimes

- movieTimeID – Unique ID for movie times – Integer, PK, Unique, Auto Increment, Not Null
- movieTime – Movie time – String, Not Null
- movieID - FK Movies.movieID (On Delete Cascade) – to identify movie for a time

6. MovieShowings

- movieShowingID – Unique ID for movie shows – Integer, PK, Unique, Auto Increment, Not Null
- showCapacity – Number of seats for a movie - Integer, Not Null
- movieID - FK Movies.movieID (On Delete Cascade) – to identify movie
- movieDateID - FK MovieDates.movieDateID – to identify movie for a date
- movieTimeID - FK MovieTimes.movieTimeID – to identify movie for a time
- theatreID - FK Theatres.theatreID (On Delete Cascade) – to identify theatre for the movie

7. userBookings

- bookingID – Unique ID for booking – Integer, PK, Unique, Auto Increment, Not Null
- numTickets - to identify number of tickets booked – Integer, Not Null
- userID – FK Users.userID (On Delete Cascade) – to identify user and his booking
- movieShowingID - FK MovieShowings.movieShowingID (On Delete Cascade) – to identify booking for movie for a date and time

API Design

This API is responsible for handling user authentication, user bookings, adding movies and theatres, retrieving movie and theatre information, and more. The Break down of main elements implemented:

1. **Authentication and Authorization:**
    - The API uses JWT (JSON Web Tokens) for user authentication. It provides endpoints for both user and admin login.
    - The jwt_required() decorator is used to secure certain routes, ensuring that only authenticated users can access them.
2. **User Functionality:**
    - The user-related routes include:
        - /userlogin: For user login.
        - /user/home: User home page.
        - /user/theatres: Retrieve a list of theatres.
        - /user/movies: Retrieve a list of movies.
        - /user/search: Perform searches based on query parameters.
        - /user/search/<string:query>: Perform searches based on a specific query.

        - /user/<string:theatre>/movies: Retrieve movies for a specific theatre.
        - /user/bookings: Retrieve the user's booking history.
3. **Admin Functionality:**
    - The admin-related routes include:
        - /admin/addTheatre: Add a new theatre.
        - /admin/theatres: Retrieve a list of theatres.
        - /admin/addMovie: Add a new movie along with its showings.
        - /admin/movies: Manage movies (update/delete).

Architecture and Features

The project folder named Ticket Booking System consists of all the required files. This consists of two folders named frontend and backend consisting of files such as config.py, controllers.py, database.py, and models.py, tasks.py, components of vue, App.vue, routers etc.

This is an online ticket booking application where users can book movie tickets by logging into their accounts. The application also features an administrative panel where the admin can add, edit, and delete movies and theatres. Additionally, users can view their booking information through the application.

Video

https://drive.google.com/drive/folders/1smAFzek51v6bqgamwr34YITAc-W7az7s?usp=drive_link