Author

Sri Krishna Teja Aradhyula

21f2001389

21f2001389@ds.study.iitm.ac.in

I am pursuing an online degree at IIT Madras and also currently, I am in b tech 4rd year at vignan's foundation for science technology and research. I am passionate about programming and problem solving

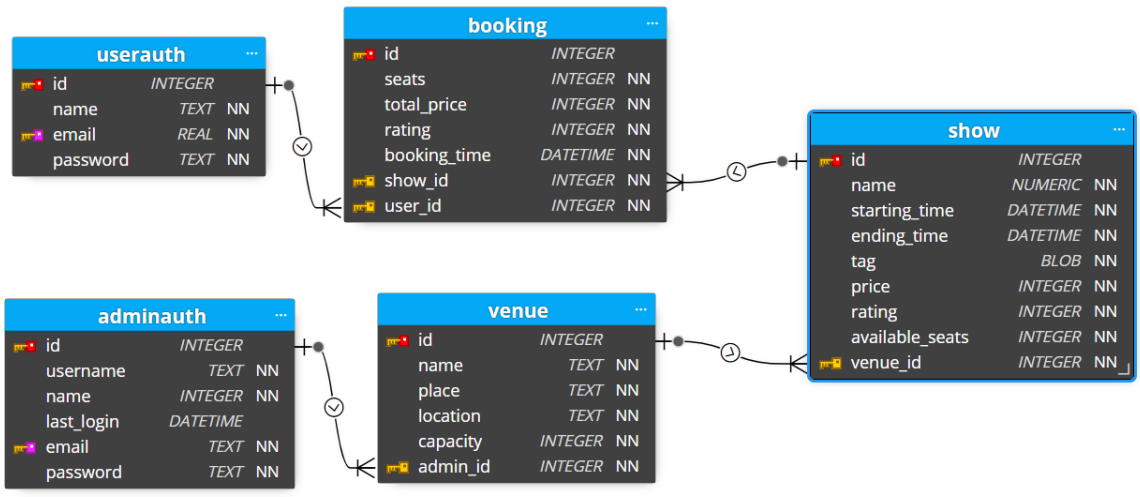
Description

The Multi-User Movie Ticket Booking Application is a comprehensive platform designed to facilitate the seamless booking of show tickets for various movies in different venues. This user-friendly application allows both regular users and an admin to efficiently manage movie bookings, venue details, and show schedules. With its intuitive interface and robust features, this app aims to enhance the movie-going experience for users while simplifying venue and show management for administrators.

Technologies used

- Vue.js
- Flask
- HTML
- CSS
- Bootstrap
- Redis
- Celery
- Chart.js
- JWT
- SQLite

DB Schema Design



The database schema for the Ticket Booking and Event Management System consists of interconnected tables: UserAuth for user authentication, AdminAuth for administrator authentication, Venue for venue info, show for event details, and booking for user bookings. UserAuth stores user credentials, AdminAuth maintains admin data. Venue stores details, linking to admins. Show holds event info, referencing venues. Booking records user bookings, referencing users and shows. This schema ensures secure authentication, streamlined management, efficient booking, and insightful event tracking.

API Design

The Flask API features endpoints for user authentication, venue management, show scheduling, and user bookings. "Auth" endpoints handle secure user and admin login/registration with JWT tokens. "Venue" endpoints allow creating, retrieving, updating, and deleting venues, utilizing caching for efficient data access. "Show" endpoints manage show operations, using date-time handling and caching. "Booking" endpoints let users book shows, retrieve bookings, and update ratings. Endpoints are organized under clear tags for clarity. Token-based authentication secures operations, and the YAML config offers an overview of the structure with security. The API caters to a comprehensive booking platform with user-friendly features and security.

Architecture and Features

The application employs Flask as its core framework and features robust user management, including signup and login via Flask-Security or JWT-based tokens. Role-based access control (RBAC) ensures distinct user roles: Admin, Store Manager, and Customer. The interface clearly differentiates between roles, facilitating seamless navigation.

Admins undergo mandatory RBAC-secured login, while Store Managers follow a similar process. Categories and products benefit from easy CRUD operations, enhancing management for Admins and Store Managers. The purchasing system accommodates single or multiple category selections, supporting cart functionality and streamlined checkout.

Scheduled backend jobs handle tasks like data synchronization and email alerts, with daily reminders and monthly entertainment reports. Theatre and show management, exclusive to Admins, involve creation, editing, and removal with multilingual support and customizations.

Booking functionality allows users to reserve multiple tickets for shows, with Admins capable of halting bookings for full shows. Exporting theatre details via CSV is user-triggered, supported by caching strategies for performance. The API is optimized for efficiency, integrating caching techniques.

In essence, the application provides a streamlined platform for secure user interactions, effective role-based access control, efficient management of categories and products, theatre and show customization, booking capabilities, export options, reporting features, alerts, and API performance enhancements. The architecture ensures modularity and usability while catering to various user roles and needs.

Video

https://drive.google.com/file/d/10FtVMfcva40NrzMPwYRvKx5tDXpxcx8J/view