Author

Samarth Sharma
Roll no- 21f3000758
21f3000758@ds.study.iitm.ac.in
I am a student and my enthusiasm to learn new skills has drawn me towards this degree.
Apart form this degree I am a Mechatronics Engineer.

Description

I have created a multiuser task tracking web application for managing ticket bookings and providing regular updates on latest shows and monthly reports to help users plan their entertainment schedules effectively.

Technologies used

• API using Python-Flask for backend development.
• VueJS with CLI for frontend user interface.
• SQLite for database to store ticket booking information.
• Redis for caching to optimize performance.
• Celery and Redis for batch job management.
• CSS and Bootstrap for styling the application.
• Matplotlib for generating plots and graphs.
• Pandas for exporting data.

DB Schema Design

In the Ticket Booking App, the data is stored using the following three DB Models:

• USER (DB Model) -
The USER model represents individual users and contains the following columns: user_id, which serves as the primary key to uniquely identify each user; username, a unique credential used for user login; name, which stores the full name of the user; password, used for secure user login; and role, indicating the specific role or permissions of the user within the application.

• Theatre (DB Model) -
The Theatre model is responsible for storing information related to theaters. It consists of the following columns: theatre_id, serving as the primary key to uniquely identify each theater; user_id, a foreign key linking back to the USER model, allowing the application to track which user added the theater; name, the name of the theater; place, indicating the location or place where the theater is situated; capacity, representing the maximum seating capacity of the theater; and trend, which stores a relative URL pointing to the plot image that represents the theater's performance trends.

• Show (DB Model) -
The Show model contains details about individual shows. It includes the following columns: show_id, acting as the primary key to uniquely identify each show; theatre_id, a foreign key associating the show with a specific theater through a reference to the Theatre model; name, indicating the name of the show; rating, representing the rating or popularity of the show; tags, providing information about any relevant tags or categories

associated with the show; ticket_price, indicating the cost of a ticket for the show; num_tickets, the total number of tickets available for booking in the show; and datetime, specifying the date and time of the show.

• Ticket (DB Model) -
The Ticket model is responsible for tracking ticket bookings. It comprises the following columns: ticket_id, serving as the primary key to uniquely identify each ticket booking; user_id, a foreign key linking back to the USER model, associating the ticket booking with a specific user; show_id, a foreign key referencing the Show model to associate the ticket booking with a particular show; booked_tickets, indicating the number of tickets booked by the user for the show; price, representing the price of the booked tickets per unit; and total, specifying the total price of all the booked tickets.

API Design

The Ticket Booking App API is implemented using the following classes:
• Login and Signup - Used for user login and signup.
• Dashboard - Displays all the theatres and shows added by the user.
• Theatre - Provides APIs for adding, updating, and deleting theatres, as well as exporting theatre details.
• Show - Provides APIs for viewing specific show details, adding, updating, and deleting shows, as well as exporting show details.
• Summary - Contains API for the summary page, which provides a brief summary of theatres and their respective bar graphs depicting trends.

Architecture and Features

The Ticket Booking App follows a modular architecture with sub-folders for Vue CLI, resource.py for API, models.py for implementing database models, jwt_trial.py for token-based authentication, worker.py, and tasks.py for batch job implementation using Celery and Redis. The key features of the application include:

• User authentication through login and signup functionality.
• Dashboard displaying all the theatres and shows added by the respective user.
• Theatre management allowing CRUD operations on theatres and exporting theatre details.
• Show management allowing CRUD operations on shows and exporting show details.
• Summary page with brief summaries of theatres and bar graphs depicting trends.
• Token-based security implemented for login and signup processes.
• Export functionality for exporting card, list, and dashboard details.
• Data caching for improved performance.
• Daily reminder batch job to send email reminders for deadline and task completion.
• Monthly progress report batch job to send monthly progress reports via email in HTML format.

Video Link

https://drive.google.com/file/d/12IRNNvXwGKiwk8_3XbkEGdc5VZ8t08MP/view?usp=sharing