Author

Devansh Singh Parmar

Roll Number: 21f1001594

Student Email ID: 21f1001594@ds.study.iitm.ac.in

DESCRIPTION

BookIt is a streamlined web application, built using Vue.js for frontend and Flask for creating backend. The web app is designed to let users effortlessly browse movies, theatres, shows, and book movie shows remotely through a web interface.

TECHNOLOGIES USED

Vue.js: A progressive JavaScript framework for building flexible, component-based web interfaces.
Flask: A lightweight Python web framework for rapid backend API development.
Flask-JWT-Extended: Handles JSON Web Tokens in Flask for authentication and protection.
Flask-CORS: Manages Cross-Origin Resource Sharing in Flask, enabling cross-origin AJAX.
WeasyPrint: Converts HTML and CSS content into PDF documents.
Celery: Asynchronous task queue to offload time-consuming operations.
Redis: In-memory data store serving as a backend for Celery's task queue.
SMTP (with smtplib): Protocol and library combo to send emails from the application.
Fetch: Native JavaScript method for making HTTP requests to APIs.
SQLAlchemy: ORM for Python, easing database interactions and queries.

Key Motivations Behind Using These Technologies:

Flexibility & Scalability: Tools like Flask and Vue.js are both flexible and allow for easy scaling, making them suitable for both small projects and medium scale applications.
Asynchronous Operations: For better user experience, tasks that can be time-consuming (like sending emails or generating PDFs) are offloaded to be handled asynchronously using Celery. This ensures that the user isn't left waiting for these tasks to complete.
Separation of Concerns: Using Vue.js for frontend and Flask for the backend allows for a clear separation between the user interface and the business logic.

DB Schema Design:

1. **User Table**:
    - **Fields**: **user_id**, **username**, **name**, **password**, **email**, **last_activity, role**.
    - **Constraints**: **user_id** as primary key, **username** must be unique, **role** not null
    - **Purpose**: Stores registered users' data while the role feature can be ‘user’ or ‘admin’.

2. **Theatre Table**:
    - **Fields**: **theatre_id**, **user_id**, **theatre_name**, **address**, **capacity, city**, **screens**.
    - **Constraints**: **theatre_id** as the primary key, **user_id** as the foreign key
    - **Purpose**: Stores information about theatres.

3. **Movie Table**:
    - **Fields**: **movie_id**, **movie_name**, **genres**, **description**, **release_date, image_url, rating**
    - **Constraints**: **movie_id** as the primary key.
    - **Purpose**: Keeps track of movies in the system.

4. **Show Table**:
    - **Fields: show_id, movie_id, theatre_id, timing,show_capacity,date,ticket_price,screenNumber,seats_booked**
    - **Constraints**: **show_id** as primary key, **movie_id** & **theatre_id** as foreign keys.
    - **Purpose:** Stores shows data of shows added to a theatre by an admin
5. **Booking Table**:
    - **Fields**: **booking_id**, **user_id**, **show_id**, **booking_date**, **ticekt_count**, **total_price**
    - **Constraints**: **booking_id** as primary key, **user_id** & **show_id** as foreign keys.
    - **Purpose**: Whenever a user books ticket/tickets the specific booking is added to the database.

API Design:

1. **User Endpoints**:
    - **/signUp**: registers new users.
    - **/login**: Authenticates existing users and logs them in by sending back a new jwt token.
2. **Theatre Endpoints**:
    - **/api/TheatreData**: Adds, gets, and edits theatres data.
    - **/api/UserTheatreData/<int:theatre_id>**: gets theatre data related to one user.
3. **Movie Endpoints**:
    - **/api/MoviesData, /api/MoviesData/<string:movieName>**: Gets all movies data for user home page
    - **/api/UserMovieData/<int:movie_id>**: gets single movie data based on movie id
4. **Show Endpoints**:
    - **/api/ShowsApi, /api/ShowsApi/<int:theatre_id>**: Adds a show to a given theatre or gets a show detail.
    - **/api/EditShow, /api/EditShow/<int:show_id>**: To edit and delete existing shows.
5. **Booking Endpoints**:
    - **/api/BookTicketsApi**: Allows users to book tickets for a show.
6. **Statistics Endpoints**:
    - **/AdminHomePageStats**: Fetches statistics for the admin dashboard.
    - **/generate_pdf**: Asynchronously generates and emails a PDF report of dashboard.

Some other APIs include **SearchMoviesApi** (returns movies similar to search query), **CurrentBookingsApi** and **BookingHistoryApi**, **ScreensDataApi** (gets screen data of a theatre), **TheatresInCityApi** (gets theatres based on cities)

Architecture and Features:

Architecture:

The movie ticket booking application is structured into two primary directories: Frontend and Backend. The Frontend directory encapsulates all user interface elements and client-side logic. Within it, the src directory contains several sub-directories: assets (for images and static content), components (for reusable Vue components), router (defining navigation rules with index.js), and views (containing primary Vue templates for rendering main pages). Additionally, app.vue serves as the root Vue component, and main.js initializes the Vue application.

Conversely, the Backend directory is dedicated to server-side operations and data management. It incorporates multiple Python scripts responsible for configuring Flask, orchestrating data flow, and defining API endpoints. An integral part of this directory is the SQLite database, ensuring persistent and structured data storage for the application.

Features:

Users on the platform can effortlessly search for movies and book movie tickets, browse local theaters, and see their current shows. Users can also access their current bookings and booking history right from their profile. Moreover, they have the flexibility to update their personal details anytime.

Administrators have additional privileges, that can't be accessed by normal users. The backend can differentiate between normal users and admin through the role feature of the database and the use of jwt authentication. Admins can easily add or list theaters or shows, and modify show details or theatre information. A vital utility for them is the dashboard present on the home page, which provides admins with data visualization and some important business metrics which they can export as pdf.

Video Link: https://drive.google.com/file/d/1Yccd9EIkMScNQz8aFMO0-6yiOFaux_H-/view?usp=sharing