MAD 2 - TICKET BOOKING PLATFORM

PROJECT REPORT

Author

Name: GRISS HARRIS DENNIS
Roll No: 21f3001987
@Email: 21f3001987@ds.study.iitm.ac.in

Description

MAD 2 Project is creating a Ticket Booking Application using Flask as backend and Vue as Frontend.The users can use the application to book their favourite shows and movies from their nearest or favourite theatres.

Technologies used

The application uses Flask as backend server and Vue.js as the frontend framework.The packages installed for the application are:

1. **Flask:** The core framework for building the backend of the application.
2. **Flask-RESTful:** Simplifies building RESTful APIs, handling API endpoints.
3. **Flask-SQLAlchemy:** Integrates the SQLAlchemy library, enabling interaction with the database.
4. **Flask-Security:** Manages user authentication, authorization, and role-based access control.
5. **Flask-CORS:** Enables Cross-Origin Resource Sharing (CORS) support in Flask applications.
6. **bcrypt:** Provides password hashing for secure storage in the database.
7. **csv:** Handles reading and writing CSV files.
8. **smtplib:** Sends email messages from Python scripts.
9. **base64:** Encodes and decodes binary data as base64 strings.
10. **Jinja2:** A template engine for rendering dynamic data in templates.
11. **WeasyPrint:** Generates PDF reports or documents from HTML.
12. **celery.result.AsyncResult:** Allows querying the status and results of asynchronous tasks using Celery.
13. **matplotlib.pyplot:** Provides an interface for creating plots and charts.
14. **Flask-Caching:** Adds caching support to Flask applications.

DB Schema Design

Models used in this application are:

- **User** model represents users with attributes like username, password, email, and more.
- **Role** model represents roles assigned to users with attributes like name and description.
- **role_users** is association table for many-to-many relationship between users and roles.
- **Venue** table represents venues with attributes venue_id, name, place, location, capacity, and a relationship with the **Show** model.
- **Show** table represents shows with attributes show_id, name, rating, tags, show_time, price, image_file, and a relationship with the **Venue** model.
- **Show_Venue** table serves as the association table for the many-to-many relationship between **Show** and **Venue** models with attributes show_venue_id, show_id, and venue_id.
- **Booking** model represents a booking with a unique **booking_id**, associated **user_id**, **show_id**, **venue_id**, and the number of **seats**.
- **UserRating** model represents user ratings with a unique **user_rate_id**, associated **user_id**, **show_id**, and a **rating** (as a string).

API Design

1. **Login API:** Handles login for both admin and users.
2. **User API:** Retrieves and registers user details.
3. **Admin API:** Fetches admin details.
4. **Venue API:** Performs CRUD operations (get, post, put, delete) on venue details.
5. **Show API:** Performs CRUD operations (get, post, put, delete) on show details.
6. **Booking API:** Saves user bookings.
7. **Rating API:** Saves user ratings.
8. **Search API:** Used to search for shows and venues.
9. **Logout API:** Allows users to log out.
10. **Summary API:** Generates summary reports of shows.

Architecture and Features

In the Project folder frontend files like js and vue files are stored in templates folder and .py files and database file are stored in the root folder.

Search feature is implemented for users to search for the shows and venues.

Housefull feature is implemented for shows if seats become full.

Admin and User dashboard have venues and shows they are showing .Admin can delete and edit shows and venues using pages for respective methods. Booking page lets user book the shows.In UserBookings page, user can rate the show.Login Page for login for users and admin.Summary page shows the summary report of the shows.

Video

https://drive.google.com/file/d/1_h196wz1HIJcx_IxJHcQVpZqFCjpfJLq/view?usp=sharing