Author:
Shubham
21f1007067
21f1007067@ds.study.iitm.ac.in

Description:

This project is a Ticket booking Web Application
-where a user can register, login, search by shows or venues or by genre, user can also make use of filter by location or rating or genre, book tickets for multiple shows and manage them.
- and admin can create venues and shows then allocate multiple shows to multiple venues (many to many relationship)

Technologies used:

1. Flask – for application code.
2. Jinja2 templates and Bootstrap for HTML generation and styling.
3. SQLite and SQLAlchemy for data storage.

DB schema design:

a. The database has several models/labels created: ***User, Show, Venue, Show_venue, Ticket_booked, Show_rating, Dynamic***. Each table has different attributes and helper function.

b. The database is designed to store show, venue, which show is allocated to which venue, user, dynamic price, and show rating.
c. Structure and details in the columns:
- **Venue:** This table stores information about Venue with venue_id, venue_name, capacity, place, location.
- **Show:** This table stores information about Show with show_id, show_name, show_tag, show_lang, show_duration, show_description, show_image_path.
- **Show_venue:** This table stores about which show is allocated to which venue with show_venue_id, venue_id , show_id, show_price, show_timing, show_added_timing.
- **Show_rating:** This table stores information about Shows rating with show_id, rating, no_of_rating.
- **Dynamic:** This table stores information about dynamic changes such as show_id, seat_left, current_price.
- **User:** This table stores information about Users such as id, email, password.
- **Ticket_booked:** This table stores information about Ticket booked with booking_id, user_id, show_venue_id, number_of_ticket_booked, cost_at_the_time_ticket_booking, time_of_ticket_booked.

API design:

I have not implemented API functionality (part of recommended section and not core)

Architecture and Features:

This project is organized using Model-View-Controller (MVC) architecture, with controllers handling logic and routing, templates for displaying views, and models for interacting with the database

1. Core features:

a. Admin login and User login:

☑ Admin login

☑ User login

b. Venue Management (only for admin):

- ☑ Create a new venue
- ☑ Edit a venue
- ☑ Remove a venue

c. Show Management (only for admin):

- ☑ Create a new show
- ☑ Edit a show
- ☑ Remove a show
- ☑ Allocate venue to shows

d. Search for Venues and Shows:

- ☑ Ability to search venue based on location (both filter and search)
- ☑ Ability to search shows based on Tags and rating etc.
- ☑ Basic home view for shows with venue

2. Additional Feature

a. Dynamic Pricing

- ☑ Dynamic pricing (ticket fluctuate with ticket booked and time left to show)
- ☑ Different pricing for each venue to shows

b. Proper Login System

- ☑ Login Manager for User with hashed password
- ☑ Session for Admin login and logout

c. Predict popularity of show and venue based on previous trends

d. Dashboard for Admin

- ☑ Top rated and top revenue generated
- ☑ Graph for ticket booked vs Show

Video Link:

https://drive.google.com/file/d/1q2Mr1p8rqwMEosdyob6ZgFsj5z6cAGHr/view?usp=sharing