Author

Sayan Ghosh **21F1006567** 21f1006567@ds.study.iitm.ac.in

Enthusiastic Computer Science and Data Science Student.

Description

TicketShow v1 is the MAD-1 project for January-2023 Term of IITM BS Degree. In this we need to create an online movie ticket booking application with multiple users, and multiple venues. Admin adds venues and shows, user books or cancels, admin sees analysis.

Technologies used

- Flask – for backend server
- Jinja – for templating the front-end VIEW pages
- Flask-Login – for user authentication
- Flask-Migrate (alembic) – for database versioning and migration
- Flask-RESTful – for API routes
- Flask-SQLAlchemy – for database Object Relational Mapper
- FLask-WTF (wtforms) — for forms (user input)
- Python-dotenv – to load config environmental variables from .env file
- SQLite3 – for database storage
- Bash – for run script
- Chart.js – for showing analytics charts

DB Schema Design

User:

column	details
id	Integer, Primary Key
username	String(64), Unique
password_hash	String(128)
name	String(128)
is_admin	Boolean

- Is_admin stores whether the user is a normal user (who books) or an admin user (who creates venues and shows)

Show:

column	details
id	Integer, Primary Key
venue_id	Integer, Foreign Key
name	String(128)
rating	Float
start_time	DateTime
end_time	DateTime
price	Float
tags	String(256)

- Each show belongs to one venue (many-to-one relationship) so each show record stores the primary-key of the venue it belongs to.
- Each User can book tickets to many shows
- Each shows can have many users booking tickets of it
- So there is a Many-to-many relationship between Users and Shows.
- So a join table is required to store this detail.
- This is done in the Booking Table.

Booking:

column	details
id	Integer, Primary Key
show_id	Integer, Foreign Key
user_id	Integer, Foreign Key
booking_time	DateTime
seats	Integer

Venue:

column	details
id	Integer, Primary Key
name	String(128)
address	String(256)
city	String(64)
capacity	Integer

- Each venue has a fixed capacity so it is stored along with other venue details.
- City stored separately to query on it.

API Design

API for CRUD of venues and shows are created as well as API for listing all the venues and all the shows. More details are present in the **openapi.yaml** file.

Architecture and Features

- The **CONTROLLERS** are located in the "**ticketshow/routes**" directory, which contains Python files such as **auth.py, error.py, common.py, user.py, admin.py, venue.py,** and **show.py**, responsible for handling the different aspects of the application's routing logic.
- The **TEMPLATES**, used for rendering the views, are stored in the "**ticketshow/templates"** directory, with subdirectories for each component, such as **"components," "show," "venue," "profile,"** and **"auth."** These subdirectories contain HTML files specific to the various features of the application, like **login, registration, and managing shows and venues**.
- The project also contains a "**static**" folder, housing the "**css**" subdirectory, which holds the **CSS** files for styling the application.
- The main application file, **app.py**, is located in the root directory, along with other essential files like:
  - **config.py** for configuration,
  - **models.py** for **database models**,
  - **forms.py** for form classes, and
  - **util.py** for utility functions.
- Finally, there's a **"ticketshow/api"** directory for API-related functionality, including **__init__.py, venue.py,** and **show.py.**

Features:

- **Authentication -** Login, User Register, Admin Register pages to perform necessary authentication. Admin users can only be added if no admin exists, or by a pre-existing admin user to keep the app safe. Auth is handled using Flask-Login (Session Based Authentication)
- **Admin Panel -** Admin can add/edit/delete venues and shows. Shows show ratings with custom colours. Tags are split up into individual pills. Deleting a venue deletes all the shows in it. Deleting a show deletes all the bookings of it.
- **User Homepage -** User can see all the venues and the shows available in them. Book option is available for shows which are in future and tickets not sold out yet. Otherwise appropriate message shown.
- **Search -** Users can search using **location, tags,** and **rating.** The filters can also be combined to have a very powerful searching capability.
- **Bookings -** Users can book shows on the homepage, those booked shows are visible in bookings, where they can cancel not-yet-over show bookings.
- **Profile -** Both users and admins can view their profile page. A display picture is randomly generated using their username as seed. They can also change their name or username, and update their password.
- **Analytics -** Admin can see detailed analytics of the platform in the analytics tab. Quantities like number of shows per venue, bookings per venue, bookings per show, bookings per user, etc are visible. Admin can also export the data as **csv** and download the **zip of all the csv files** at once. Charts are shown using **chart.js**

Video

https://drive.google.com/file/d/1zVkYXT4M97AWjDrDumDpWMM-P_zaMkPT/view?usp=share_link