Author

Amal Roy
21f1004594
21f1004594@ds.study.iitm.ac.in
I am from Kerala and I currently work with reliance as an operations manager.

Description

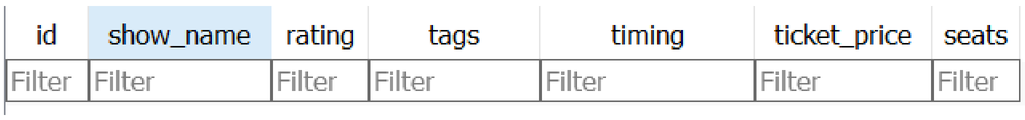
The project should have seprate access for admin and users. The admin must have the power to add ,edit ,delete venues or shows. The user must have the option to book the tickets for their favourite shows in a particular venue and able see their past and current bookings.

Technologies used

The website is completely built upon python flask framework. The extensions used are flask sqlalchemy, redirect, return_template, jinja, return_template_string, request and bootstrap.
Sqlalchemy: For database connection
Redirect: to navigate to a particular router
Return_template: to get the html page in the router
Return_template_string: To write HTML document in return statement.
Jinja: for dynamic use in html.
Bootstrap: for styling

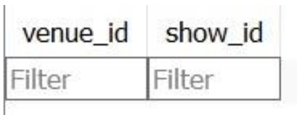
DB Schema Design

A total of 5 database tables were created and they are the following:



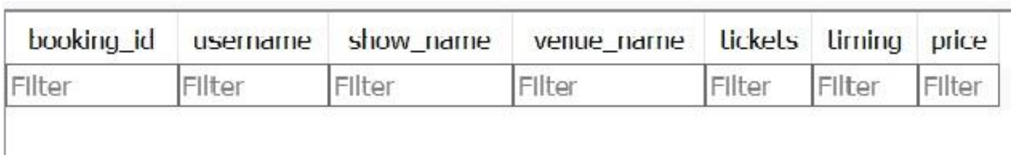
id	show_name	rating	tags	timing	ticket_price	seats
Filter	Filter	Filter	Filter	Filter	Filter	Filter

1.Shows management table, to add shows . the attributes seats takes value venue management table seats automatically. This can only be modified by the admin



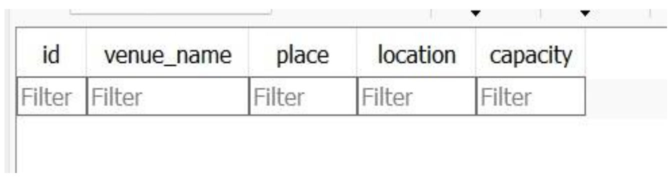
venue_id	show_id
Filter	Filter

2.Association table ,since the problem is a many to many relationship.it takes id of two tables and makes a direct relation out of it



booking_id	username	show_name	venue_name	tickets	timing	price
Filter	Filter	Filter	Filter	Filter	Filter	Filter

3.This table gets populated when the user books a ticket for him or her.



id	venue_name	place	location	capacity
Filter	Filter	Filter	Filter	Filter

4.Venue management table for adding new Venue. Can be only operated by the admin

Reasons for such tables.

1.The show management table and Venue Management table are the core requirements for this project and the attributes given was mandatory for such a project to run smoothy and a user friendly way.

2.The association table was built to make a many to many relationship between shows and venues since , a venue can host many number of shows and same shows can happen in many different venues.

3.The booking table was designed in such a way that the admin can clearly understand how many shows are booked for a particular venue and the total revenue earned on each venues.

Architecture and Features

The backend python code is all written in main.py file. It contains database connections, flask extensions, database structure and all the @app.route definition which takes to different pages in the frontend and also some computations for dynamic pricing.

All the html codes are written in the templates folder. A total of 12 html pages are created for the smooth operation of the website. All the images used in the homepage of user are stored in the static folder.

Features of the Website.

Features of Admin

1.The admin has a seprate login page and the username of the admin is set to a constant variable.

2.The admin will be able to add ,edit and delete a venue.

3.Admin can create a show in a particular venue. Shows can be edited or deleted. The association table helps here to create shows for one particular venue selected.

Features of user.

1.User has a seprate login page. If the user is not a member, he/she has to register and then will be allowed to login.

2.User has the option to select a particular city to see the venues and shows happening for that city.

3.The user has the option to check his/her tickets which are booked in the mybookings page and has the option to cancel the ticket.

Additional features.

1. The option of dynamic pricing is implemented for this project. If more than 75% seats are booked then the price would increase by 200rs and if its less than 30%, price will decrease by 100rs.

Video

https://youtu.be/G3JnCWysYao