Author

Name: Kevin Joshua T
Roll number: 22f1001410
student email: 22f1001410@ds.study.iitm.ac.in
About me: I am a student currently living in Chennai, pursuing an engineering degree, along with with this course, I am into full stack web development, and bleeding edge hardware technology.

Description

In this project a simple web web interface that is communicative of its current state and the ability to retrieve data based on parameters is required. There is also supposed to be forms for creating or deleting the new data passed by the user.

Technologies used

os: To get the paths required for the smooth functioning of relative addressing
Flask:Core requirement for running web applications based on python.
flask_sqlalchemy:For ensuring base is declared automatically taking the issue of fixing the base directly to flask, integrating without any intervention
Sqlalchemy: Using some functions not in flask_sqlalchemy due to weight, like date and func.
flask_bcrypt: To hash the user password for security
flask_login: To authorise the entry and exit using login and logout
Plotly: For plotting graphs based on the event data to show some basic statistics to the admin users
re: to use regular expressions for evaluating the password and email id to check if they match the format
Datetime: Since the shows would use dates to schedule events it has been included
pytz: datetime.now() was not able to compare with time unaware functions, hence it has been used to set everything to IST.

DB Schema Design

user table has user_id as primary key which auto increments to provide unique serialisation of each user not limiting to just email, hence the same user can have both an admin and a user account without resorting to a separate table. An email and and password for forward facing user uniqueness as the primary key will not be shown to them.username for providing some friendliness and customisability, and level to indicate what type of user they are

admin table has stage_id as primary key a foreign key as user_id from user table which identifies who each venue belongs to, a location for filtering and stage for naming the venue and seats for capacity

show table has show_id as primary_key which auto increments, stage_id to check if times are clashing, user_id to identity the ownership of the show, Starttime and end time for time constrains, show for knowing the name of the show, stage for forward facing simplification of accessing venue name, seats_left that update each time a ticket is booked, cost for setting the price for each show and tags for filtering on the ticket purchaser's side.

Rating has a distinct table show and it uses the show name as primary key to ensure that no entry is repeated twice, rating to show the rating as it is simpler and lesser load to compute and save the data once than to compute it every time.stars to find the total stars given to a show. count to show how many people have rated the show for later calculation

bookings table has booking_id for uniquely identifying booking details for each user as primary key. user_id that has the user that booked the ticket.show_id for knowing which show has been booked.show for knowing the name of the show for easily displaying the data without merging two tables in sql.cost to indicate the amount paid by the purchaser how much they spent for the show and tickets to show how many tickets have been bought.rating to check in backend if a show has been rated by the user.

API Design

No api has been used for this.

Architecture and Features

The project is organised in a way that it is evident that a single person has developed the app, there isn't any extra external packages that have been created. All the functions reside solely in the main.py file, and images have been used sparsely, the only images for the app icon to display in the corner of the webpage. All the jinja2 templates reside inside the templates folder. Bootstrap has been download locally as a .min file for lower size, and for ensuring integrity of the data so that I am not affected by any changes to it. The bootstrap and image file has been saved in the static folder.

Most of the features are basic features that have been implemented are simple CRUD features rendered by simple UI, that has not been bloated by unnecessary code, which trying to maintain a clean experience. For example I noticed that in the login page if I put nested collapsibles, the inner collapsible pops when it finishes it animation, so it would be jarring. The only part that hasn't been modified for aesthetics is the datetime selector as that would necessitate the download of a separate library solely for its function.

other factors that separates this from other apps would be things like its redundancy in tables that add to larger tables while lowering search time, and a time aware table rating feature that shows up once you have finished watching a show, but has been tucked away so that it does not instruct the user when they are solely engaged in looking for a show.

Video

https://drive.google.com/file/d/1wzcy1iLV91bfbfvGQ9lxJluF9_6vsgwn/view?usp=sharing