MODERN APPLICATION DEVELOPMENT-II PROJECT REPORT by:

Name: Chandresh Jagjivanbhai Sutariya
Roll number: 21F3001415
Student email: 21F3001415@ds.study.iitm.ac.in

A short "About me": I'm 22, a boy! I'm a Civil Engineer, born and living in Surat City, Gujarat. I'm also doing a merchandise export business of polished natural diamonds.

Description of project

I have to make a ticket management and booking application for some movie theatres, and the app at least should have features described in the problem statement as "core".

Technologies used

Flask	:	As an app backend.
Flask_sqlalchemy	:	To make the database model
datetime	:	An python inbuilt library to get the current timestamp
flask_security	:	To add security features
celery	:	To create workers(celery workers)
smtplib	:	pyhon inbuilt library to make the connection to SMTP server and send requests
email	:	python inbuilt library to attach data to create a message that is to be sent
jinja2	:	To render data in an HTML file which is to be sent for email

DB Schema Design

- **User:**

1.	Id	:	A unique id to identify user
2.	username	:	a string can be used for login purpose
3.	email	:	to send email and also a required thing for login
4.	password	:	security
5.	active	:	For security purpose
6.	last_logout_at	:	To check when the user visited last time, and to send emails if not visited in 24 hours
7.	fs_uniquifier	:	A constraint to be added to use flask_security
8.	role	:	To add RBAC(Role Based Access Control) to the application
9.	roles	:	Relationship column with Role table

- **Role**

1.	Id	:	A unique id to identify role
2.	name	:	Role title
3.	description	:	A short thing about the role

- **Venue**

1.	Id	:	A unique id to identify venue
2.	venue_name	:	venue name
3.	place	:	place of the venue
4.	location	:	location of the venue
5.	capacity	:	capacity of the venue

- **Show**

1.	Id	:	A unique id to identify the show/movie
2.	venue_id	:	venue_id of the show
3.	show_name	:	show name
4.	rating	:	rating of the show
5.	date	:	date of the show
6.	timing_starts	:	starting time of the show
7.	timing_ends	:	ending time of the show
8.	tags	:	tags of the show/movie
9.	price	:	ticket price of the movie
10.	seats	:	available seats for the show
11.	booked_seats	:	Number of booked seats

- **Bookings**

1.	Id	:	A unique id to identify the booking
2.	user_id	:	user_id of the user who made the booking
3.	venue_id	:	venue_id of the venue of the booking
4.	show_id	:	show_id of the booked show
5.	seats_booked	:	Number of seats booked by the user

API Design

As per the core requirement, I had to use Vue for the front end, and I am communicating the database from the front end via API calls on the back end.
I'm making API calls from the front end using the fetch method in Vue/JS.

I have created API for CRUD operation on Venue, Show, and Bookings tables, and also using API for user creation and for all the tasks where the database comes in the picture. Also using API to trigger celery jobs and to know the status of the job.

Architecture

Requirements are in the root folder with file name "requirements.txt". Configurations for Flask security is in file "flask_security_config.py" in the root folder. All controllers and other configurations are in "app.py" file. All static files are stored in the "static" folder, and all templates are stored in the "templates" folder in the root folder.

Features

I have added all core features given in the problem statement. Here I'm describing the major and extra features I have added.

1. **User signup and login; Admin Login**
   - ➔ For signup I'm using a custom form, and for login, I'm using the default login page given by the falsk-security for both user and admin login
   - ➔ For signup – reactive data validation for email and password. Users will get real-time warnings while writing emails for invalid data.
   - ➔ I'm creating an admin when the app runs the first time(when the database is not created)

2. **CRUD operation on the theatres (only by admin)**
   - ➔ data validation and reactive warnings for wrong data input while creating a venue
   - ➔ Admin can add venue capacity

3. **CRUD operation on the shows (only by admin)**
   - ➔ data validation and reactive warnings for wring data input while creating the show
   - ➔ Admin can add price, and seats capacity for a particular show while creating the show

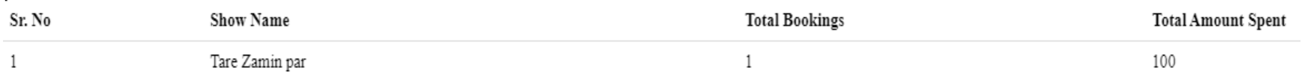
4. **User home view and Search option for show**
   - ➔ User can search shows as per the location, and the tag of the movie. The search is done on the frontend of the browser; I'm not sending any request to the backend for implementing the search.
   - ➔ User can only see the shows which are going to happen in the future – meaning, I am not showing the show, which has an older date or whose date has already passed.
   - ➔ User can see the price, date, and available seats on the home page for each show

5. **Caching**
   - ➔ I have added caching feature for faster UX

6. **Booking**
   - ➔ A user can book as many tickets as she/he wants
   - ➔ While making the booking, the user can see real-time changes in the amount as per the change in the booking numbers
   - ➔ User will get a notification saying the number of available seats if booking numbers exceeds available seats

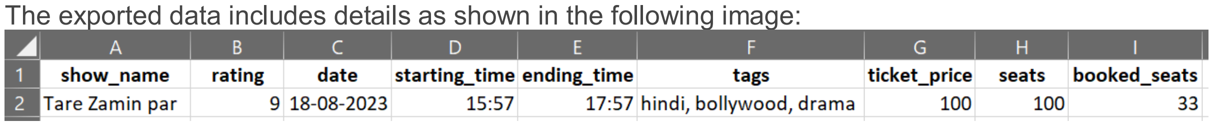
7. **Daily Reminder** – via email if a user has not visited the application for 24 hours.

8. **Monthly entertainment report** – The user gets an email on the first day of the month, with details shown in the picture below via mail.



Sr. No	Show Name	Total Bookings	Total Amount Spent
1	Tare Zamin par	1	100

9. **Export data by Admin only – Async job using celery workers**
   - ➔ An Admin can export data for a particular venue. I have given a special dashboard to export data for a particular venue. The data is exported in CSV format with the name of the file as <<venue_name>>.csv
   - ➔ The exported data includes details as shown in the following image:



	A	B	C	D	E	F	G	H	I
1	show_name	rating	date	starting_time	ending_time	tags	ticket_price	seats	booked_seats
2	Tare Zamin par	9	18-08-2023	15:57	17:57	hindi, bollywood, drama	100	100	33

Other than these, many more features are added. Here I have only described the major/important features and their implementation. The small features like - real-time notification of the status of the celery job etc. are not described here.

Video

https://drive.google.com/file/d/1Gl-diXmjvAhbsPoCGstA5hHLycsBDjWX/view?usp=sharing