BMS

Name: Abhishek Prasad Prajapati
Roll number: 21f1003546
Email-id: 21f1003546@ds.study.iitm.ac.in
I am graduate mechanical engineer and currently pursuing this course.

This is a modern multi-user application that allows users to book show tickets, while the admin can manage theatres and shows. The app utilizes Flask for RestAPI, VueJS for UI, SQLite for the database, Redis for caching, and Redis and Celery for batch jobs among other frameworks and technologies.

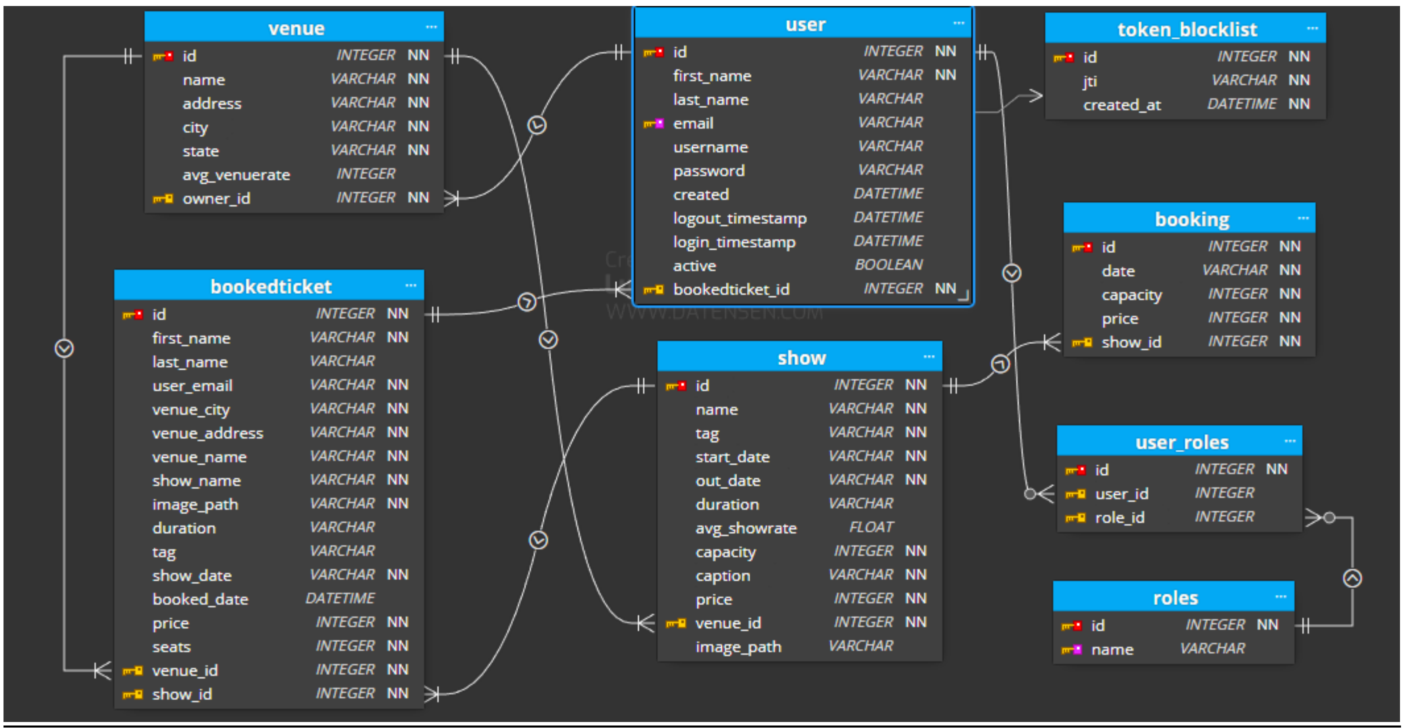
Functionalities of TicketShow app are following:

(1) **Theatre and Show Management:** The admin can create, edit, and remove theatres and shows, including their details such as name, rating, ticket price, capacity, and tags.
(2) **User Signup and Login:** Users can sign up with a username and password and log in to access the ticket booking platform.
(3) **Show Search and Booking:** Users can search for shows and theatres based on location preference, tags and book tickets for available shows.
(4) **Daily Reminders & Monthly Entertainment Report:** The system sends daily reminders for user engagement if used has not booked show within 3 days and generates monthly reports with booking details, show ratings, etc., sent via email.
(5) **User Triggered Async Job**: Export as CSV: Admins can trigger batch jobs to export theater details, bookings, and ratings in CSV format, with an alert sent upon completion.
(6) **Performance Enhancement through Caching:** Caching is implemented using Redis to boost the application's performance, reducing response times and optimizing API calls for a seamless user experience.
(7) **Dynamic Ticket Pricing based on Show Ratings:** The app has feature of dynamic pricing, where ticket prices for shows can vary based on their popularity and ratings.
(8) **Downloadable PDF Tickets:** Users can download well-designed PDF tickets for their booked shows, containing all essential details for a seamless ticket confirmation experience.

Technologies used:

• Application code: Flask RestAPI, SQLAlchemy, Redis, Celery, Flask JWT
• Database: SQLite3
• Style: VueJS, Jinja2, Bootstrap

DB Schema Design:



Folder Structure:

Root Structure:

```
.
└── ProjectFolder/
    ├── Project/
    │   ├── API
    │   └── Frontend
    ├── Project_doc.pdf
    ├── README.md
    ├── requirements.txt
    └── api_yaml_file.yaml
```

Backend Structure:

```
.
└── API/
    ├── auth/
    │   ├── auth.py
    │   └── role_selection.py
    ├── admin/
    │   ├── shows.py
    │   └── venues.py
    ├── user/
    │   ├── rating.py
    │   ├── search.py
    │   ├── shows.py
    │   └── venue.py
    ├── env
    ├── static/
    │   └── shows_image
    ├── templates/
    │   ├── daily_mail_temp.html
    │   ├── monthly_mail_temp.html
    │   └── ticketPDF.html
    ├── app.py
    ├── caching.py
    ├── celery_worker.py
    ├── config.py
    ├── models.py
    ├── tasks.py
    └── test.sqlite3
```

Frontend Structure:

```
.
└── Frontend/
    ├── node_modules
    ├── public
    ├── src/
    │   ├── assets/
    │   │   └── images
    │   ├── components/
    │   │   ├── admin_has_venue.vue
    │   │   ├── admin_no_venue.vue
    │   │   ├── adminNav.vue
    │   │   └── navBar.vue
    │   ├── mixins/
    │   │   └── myMethods.js
    │   ├── router/
    │   │   └── index.js
    │   ├── views/
    │   │   ├── adminShows.vue
    │   │   ├── Booking.vue
    │   │   ├── HomeView.vue
    │   │   ├── Login.vue
    │   │   ├── Register.vue
    │   │   ├── search.vue
    │   │   ├── showDetails.vue
    │   │   ├── showForm.vue
    │   │   ├── userBooking.vue
    │   │   └── venueForm.vue
    │   ├── App.vue
    │   └── main.js
    ├── index.html
    ├── package.json
    └── README.md
```

1. <u>API End-Point Directories :</u>

- **auth/:**
  - **auth.py:** Registration and token-based authorization login.
  - **role_selection.py:** Role-based access control (RBAC) implementation.

- **admin/:**
  - **shows.py:** Show management by admin (POST/GET/PUT/DELETE).
  - **venues.py:** Venue management by admin (POST/GET/PUT/DELETE).

- **user/:**
  - **rating.py:** Rating implementation on venues and shows.
  - **search.py:** Search functionalities implementation for shows and venues.
  - **shows.py:** Show-related interactions for users.
  - **venue.py:** Venue related to interactions for users.

2. <u>Static and Template Directories:</u>

- **static/:** Shows related images.
- **templates/:**
  - **daily_mail_temp.html:** Template for sending daily reminder email.
  - **monthly_mail_temp.html:** Template for sending monthly entertainment report email.
  - **ticketPDF.html:** Template for generating PDF tickets.

3. <u>Root Directory:</u>

- **app.py:** Main application script for running flask app.
- **caching.py:** Implementation of caching functionality using Redis.
- **celery_worker.py:** Implementation of Celery worker for executing asynchronous tasks.
- **config.py:** Configuration settings for the app.
- **models.py:** SQLAlchemy models for the database.
- **tasks.py:** Implementation of Celery tasks for background jobs.
- **test.sqlite3:** SQLite database for storing app data.

<u>Project Video Link:</u> https://drive.google.com/file/d/1K72hscP8JYF0FlFz0xZ6ZYWugAj1MK-6/view?usp=sharing