Author

Satyam Kumar
21f2000243
21f2000243@ds.study.iitm.ac.in

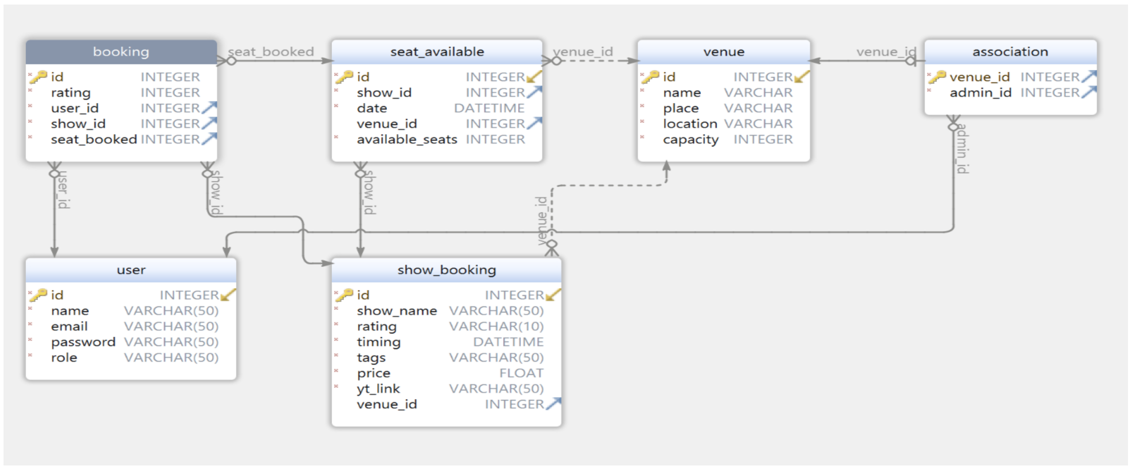
Description

This project allows hosts to manage shows and venues, and users to book tickets. It has a search function with filters, and displays shows based on their timeframe. When seats are full, it displays that the show is houseful. Users can rate and write reviews about the show.

Technologies used

The app is built using the Flask web framework for the backend. Key Flask extensions include Flask-JWT-Extended for user authentication, and Flask-SQLAlchemy for database management. VueJS is used for the frontend UI, along with Bootstrap for styling, vue-router for routing and Vuex for state management. Redis is employed for caching, Flask-Caching for caching, smptlibl for sending emails, flask_cors for enabling cross-origin resource sharing, requests for making http requests, and Redis along with Celery are utilized for handling batch jobs.

DB Schema Design



A Venue can host multiple ShowBookings, while a ShowBooking can only have one Venue. A User can make many Bookings, and a Booking can only belong to one User. A User can have multiple roles, and a Venue can have multiple Users with different roles, which is implemented through the Association class. A ShowBooking can have one

SeatAvailable, and a SeatAvailable can belong to one ShowBooking. Finally, a Booking can be associated with one SeatAvailable, and a SeatAvailable can have many Bookings.

API Design

The API supports CRUD for shows and venues, with two Flask-RESTful resources: VenueResource and ShowBookingResource. VenueResource supports GET, POST, PUT, and DELETE requests for venues, while ShowBookingResource supports the same for shows, along with creating a new seat available object. Both resources use Flask-RESTful's reqparse and fields modules for data validation and serialization, and the abort function for raising HTTP exceptions when necessary.

Architecture and Features

The project architecture includes key components such as Python-based scripts "app.py" and "main.py," which interact with the data model in "model.py." Celery is used for task scheduling, and a RESTful API is likely outlined in "api.yml." "Chart js" is employed for data visualization, while a potential Vue.js-based frontend resides in "v2-fronted." Dependencies are listed in "requirements.txt," static assets in "static," and the "run.sh" script handles execution.

Core features:

- Admin login and User login through JWT based token authorization
- Venues & Shows Management
- Booking of show tickets. Shows housefull if capacity is full.
- Search for shows/venues . Also , user can search shows /venues based on tags ,location ,ratings etc
- Different price for each venues
- Export venue/show engagement (number of tickets booked,venue performance) ,
  Caching and other backend jobs.
- Dynamic Pricing for top 5 shows which have a high number of bookings and time is less than 6 hrs.
- Rating and Review - The rating feature is available for users who have booked tickets to a show, and the show time has already passed. User can rate and review
- Responsive Ui
- Analytics tab for shows and venues
- Api documentation with flask_swagger_ui

Video : projectVideo