Author

Vibha S
22F2001677
https://app.onlinedegree.iitm.ac.in
I am a second term diploma student and this is my fourth term in this degree. I am also pursuing a B.E. degree in offline college. This course has been a welcome challenge and the skills I have learnt here have been have been highly invaluable.

Description

I have gone through the problem statement and taken the wire-frame as a reference for this project. There needs to be three main entities – the user, venue and shows. There is a many-to-many relationship between the user, venue and shows which must be suitably represented by efficient database model to avoid redundancy of data and perform suitable CRUD operations.

Technologies used

The technologies used in this project include Flask extensions like Flask, request, redirect, url_for, render_template, current_app, Flask-SQLAlchemy extensions like SQLAlchemy, flask_cors for cross origin resource sharing, Flask-JWT-Extended for Role Based Authentication, celery for executing backend jobs, redis for caching and queuing backend jobs, Flask-Caching for caching information and data from backend, matplotlib and Vue.js framework for frontend.

DB Schema Design

The database here has the following tables: booking, show, user, venue, venue_show, RolesUsers, role

1. user - has the columns: id – index ; username – username of user ; password – password used by user to log in ; last_active – when the user/admin logged in; email – email id provided by user
2. venue - has the columns: vid – index ; vname – name of theatre ; vplace – location of theatre ; vcapacity – the total number of seats available at theatre ; vimage – image of the theatre
3. booking – is the relation table between user and show and has the columns: id – index ; uid – user index ; sid – show index ; tickets – number of tickets booked ; price – total price of the tickets booked ; vid – theatre id
4. show - has the columns: sid – index ; sname – name of the show ; srate – rating given ti show ; stags – tags associated with the show ; sprice – price of one ticket to the show ; time – start time of the show ; seats – total number of seats available to the show ; booked – number of seats booked by users
5. venue_show – is the relation table between venue and show and has the columns: id – index ; vid – venue id ; sid – show index

6. RolesUsers – is the relation table between role and user and has the columns: id – index ; user_id – user id ; role_id – role id either 1(admin) or 2(user)
7. role – is the table which describes the roles available : id – index ; name – name of role ; description – role description

This database designing was found to be an efficient way to hold the data andquery them in need. Therefore this designing method has been adopted.

API Design

Te functions of API for backend are implemented in the file named controllers.py residing in
the applications folder with the aid of sessions.py for database querying , while retaining the basic status codes such as 200 (successful request), 304 (not modified), 503 (service unavailable), etc; whereas the API for backend are given in detail in the yaml file

Architecture and Features

The project folder has two main folders, namely backend and frontend. Inn the backend, the main python file – main.py and a file – requiremets.txt which has the names python library dependencies on which the project runs. The folder constraints db_directory which has the .sqlite3 file which contains the database, application which has all the related code in python .py files, static folder which has a folder having all the image file in the format .png/.jpeg. The frontend folder has two main folders – public ans src. All .vue files are present in the src folder which has the following – assets, components, router, views and App.vue .

The project takes us to the url “/” which is where we choose to login as user or admin and the respective login happens. The CRUD for admin is with respect to managing shows and venues whereas for user is to make new registrations, book shows, give ratings and view past bookings. The user has an option to search for shows based on the location of venue and the tags associated with the show.

Video

https://drive.google.com/file/d/180TpGWwRKbHxgBi9tzD93AMuNO9ZsEbC/view?usp=drive_link