MAD1 Project

Author

Abhay Bairagi
22f1000829
22f1000829@ds.study.iitm.ac.in

I am a student at the Indian Institute of Technology Madras (IITM) pursuing my degree in Data Science. I am passionate about developing applications and exploring new technologies.

Description:

The ticket booking app allows users to view venues and shows and book shows with seats also allows users to search for movies and various venues . The app also provides an admin interface to create venues and shows. The app has a login system for both admin and user.

Technologies used:

• Flask: A Python web framework used for building web applications.
• Flask-SQLAlchemy: A Flask extension that provides a SQLAlchemy database for Flask applications.
• Werkzeug: A WSGI (Web Server Gateway Interface) utility library for Python that provides routing, request handling, and response handling.
• Flask-RESTful: A Flask extension for building REST APIs.

Purpose:

• Flask: Used for building the web application and creating routes and views.
• Flask-SQLAlchemy: Used to create and manage the SQLite database for the ticket booking app.
• Werkzeug: Used for secure file uploads.
• Flask-RESTful: Used for building an API to interact with the database.

DB Schema Design:

The database schema consists of five tables: Venue, Shows, VenueShow, User, and ShowBooks. These tables are created using SQLAlchemy ORM in Python.

Venue Table:

- venue_id: Integer, Primary Key, Unique ID for each venue
- name: String, Not Null, Name of the venue
- location: String, Not Null, Location of the venue
- place: String, Not Null, Place where the venue is located
- capacity: Integer, Not Null, Total capacity of the venue
- shows: Relationship, Many-to-Many Relationship with Shows table, List of all the shows being held at this venue

Shows Table:

- show_id: Integer, Primary Key, Unique ID for each show
- title: String, Not Null, Name of the show
- image: String, Not Null, Default: logo.png, Name of the image file for the show
- desc: String, Not Null, Description of the show
- release_dt: String, Not Null, Release date of the show
- rating: Integer, Nullable, Rating of the show
- price: Integer, Not Null, Price of the show
- tag: String, Not Null, Tag of the show
- stock: Integer, Not Null, Stock of the show
- venue_id: Integer, Foreign Key to Venue Table, ID of the venue where the show is being held
- venue: Relationship, One-to-Many Relationship with Venue table, The venue where the show is being held

VenueShow Table:

- venue_id: Integer, Primary Key, Foreign Key to Venue Table, ID of the venue
- show_id: Integer, Primary Key, Foreign Key to Shows Table, ID of the show
- This is a join table used to represent the many-to-many relationship between the Venue and Shows tables.

User Table:

- uid: Integer, Primary Key, Unique ID for each user
- Name: String, Not Null, Name of the user
- Email: String, Not Null, Email of the user
- Pass: String, Not Null, Password of the user
- shows: Relationship, Many-to-Many Relationship with Shows table, List of all the shows the user has booked
- venue: Relationship, Many-to-Many Relationship with Venue table, List of all the venues the user has booked

ShowBooks Table:

- uid: Integer, Primary Key, Foreign Key to User Table, ID of the user
- venue_id: Integer, Primary Key, Foreign Key to Venue Table, ID of the venue
- show_id: Integer, Primary Key, Foreign Key to Shows Table, ID of the show
- seats: Integer, Not Null, Number of seats booked by the user for the show

I have designed the database schema this way because:
A venue can have multiple shows and a show can be held at multiple venues. Hence, I have created a many-to-many relationship between Venue and Shows table, with the VenueShow table serving as a join table.
A user can book multiple shows and venues, hence I have created a many-to-many relationship between User, Shows, and Venue table, with the ShowBooks table serving as a join table.

API Design

1. Venue API: This API allows CRUD operations for a particular venue. It provides endpoints to get, create, update and delete a venue. The GET endpoint retrieves a venue by ID, the POST endpoint creates a new venue, the PUT endpoint updates an existing venue by ID, and the DELETE endpoint deletes a venue by ID.
2. Venue Show API: This API retrieves all shows of a particular venue. It provides an endpoint to get all shows for a particular venue. It returns a JSON object that contains the venue ID, name, place, capacity, and a list of shows for that venue.
3. Venues API: This API retrieves all venues. It provides an endpoint to get all venues. It returns a JSON object that contains the venue ID, name, place, capacity, and location for each venue.
4. Shows API: This API retrieves all shows. It provides an endpoint to get all shows. It returns a JSON object that contains the title, price, stock, rating, and release date for each show.

Architecture and Features

This project is a ticket booking app and is organized in a structure where the instance folder contains the database, the static folder contains all media files and static files, and the templates folder contains all HTML files. The **api.py** file contains code for the API, and the **models.py** file contains all models. The main file is **main.py**, which is where the views and controllers are defined. In this project, controllers for creating venues, creating shows, logging in users and administrators, and registering users and administrators have been defined. For the login system, the **sqlalchemy** module has been used for the database, and **session** has been used to make the user remain logged in.
The features implemented in this project are a login system using SQLAlchemy and session, CRUD functionality for Venues, Shows and Admins and API to get the list of venues and shows and for CRUD operation on Venues. Additionally, users can see the list of available venues and shows and search functionality for shows and venues, and they can book tickets for shows. The project has additional features like file uploading for show images, user profiles, view booked show and user authentication to ensure only authorized users can access certain parts of the app.

Video

https://drive.google.com/file/d/1_5-RO9MSD5IyiBhkgYwbFYCj5PfXVoMz/view?usp=sharing