Author

Sarfaraz Ahmad Ali Ahmad Khan
21f3002965
21f3002965@ds.study.iitm.ac.in
I'm 24 years old, hold a bachelor's degree in mechanical engineering, and work as a developer in the telecom domain.

Description

The project requires the development of a multi-user web application using Flask, Jinja2 templates, Bootstrap and SQLite.
The application is a ticket booking platform for shows, where users can book tickets for different shows, and the admin can manage venues and shows.
The application also includes login functionality, a search for shows/venues, and the ability to book multiple tickets for a show at a given venue.

Technologies used

Flask provides a flexible and lightweight approach to building web applications in Python.
Flask-Login simplifies user authentication and session management.
Flask-SQLAlchemy provides integration between Flask and SQLAlchemy, making it easy to interact with a database.
Werkzeug handles HTTP requests and responses.
WTForms provide form validation and rendering.
The email_validator library is used to ensure that email addresses entered by users are valid.
All of these technologies work together to create a powerful and efficient web application that can handle a variety of tasks and interactions with users.

DB Schema Design

The database schema design consists of four tables: User, Venue, Show, and Booking.

User table: This table stores information about users who have registered with the system. It includes columns for the user's unique ID (primary key), their name, email address (which must be unique), a hashed password for their account, a Boolean field indicating whether they are an administrator, and a one-to-many relationship with the Booking table that allows us to access all bookings made by the user.

Venue table: This table contains information about the venues that are available for shows. It includes columns for the venue's unique ID (primary key), name, location, capacity (the total number of seats available), the number of seats that have been booked, and a one-to-many relationship with the Show table that allows us to access all shows hosted at the venue.

Show table: This table stores information about the shows that are available for booking. It includes columns for the show's unique ID (primary key), name, average rating, a list of tags associated with the show, the price of a ticket, a foreign key linking to the venue where the show will be hosted, and a one-to-many relationship with the Booking table that allows us to access all bookings made for the show.

Booking table: This table stores information about bookings made by users for specific shows. It includes columns for the unique booking ID (primary key), a foreign key linking to the show being booked, a foreign key linking to the user who made the booking, and the number of seats that were booked for the show. This table allows us to keep track of all bookings made through the system.

The reason for designing the database schema in this way is to create a normalized and efficient database structure that enables easy querying and manipulation of data. Each table represents an entity in the application, and the relationships between tables allow us to access related data easily. For example, the Booking table allows us to see which users have booked tickets for which shows, and the User table allows us to access all bookings made by a particular user. Additionally, the use of foreign keys ensures referential integrity between tables and prevents data inconsistencies.

Architecture and Features

The project is structured using the Flask web framework and follows the Model-View-Controller (MVC) architecture pattern. The app.py file contains the Flask application instance and defines the routes (controllers) for handling requests from clients. It also contains the database models for the application's data entities, including User, Venue, Show, and Booking. The templates directory contains HTML templates (views) for rendering pages to the client. The base.html file is a template that is extended by other templates to provide a consistent layout across all pages.

The application features user authentication, with separate login routes for users and administrators. Users can sign up for an account, while the admin can access a dashboard to manage the application's data entities. The application also allows the admin to create and manage venues, shows, and bookings. Users can search for shows and venues using the search functionality and can book tickets for different shows. The application also provides forms for creating, editing, and deleting venues and shows.

Video

https://drive.google.com/file/d/1gpwJHzmvHDDFvB0OuvYRi7fy7Xq7dMzg/view?usp=sharing