TicketBooking project Report

Author:

Name : Arunrajan R V
Student ID : 22f1000888
Mail id : 22f1000888@ds.study.iitm.ac.in
About me : I am from Trivandrum, Kerala. I completed my B.E Mechanical engineering in 2017 at Anna University,chennai. I worked as a quality engineer in EVERUP batteries, chennai.

Description:

The TicketBooking project aims to provide users with a convenient and efficient way to book tickets for shows at various venues. It also helps admins create venues and shows for people to enjoy.

Technologies used:

- Flask for application code
- Jinja2 templates + Bootstrap for HTML generation and css styling
- SQLite for data storage
- Flask_login is used for secure user login
- Flask_restful and Flask_cors for building RESTful APIs

DB Schema design:

The Admin table schema comprises four columns, namely admin_id, admin_name, mobile, and password. The admin_id serves as the primary key and is auto-incremented for every new admin added to the table. The admin_name and mobile columns are marked as unique and not null, thus ensuring that no two admins share the same name or mobile number. Lastly, the password column is marked as not null to ensure that every admin account has a secure password.

The Venue table schema includes five columns, namely venue_id, venue_name, place, location, and Screen. The venue_id serves as the primary key and is auto-incremented for every new venue added to the table. The venue_name, place, and location columns are marked as not null to ensure that no venue lacks these vital pieces of information. Lastly, the Screen column is marked as not null, indicating the number of screens present at each venue.

The VenueCreated table schema comprises three columns, namely vc_id, cadmin_id, and cvenue_id. The vc_id serves as the primary key and is auto-incremented for every new venue created. The cadmin_id and cvenue_id columns are marked as foreign keys and are not null, referencing the admin_id and venue_id columns, respectively. This table tracks which admin has created which venue.

The Show table schema includes ten columns, namely show_id, show_name, rating, timing, tags, screenNumber, seats, price, amount_received, and created_at. The show_id serves as the primary key and is auto-incremented for every new show added to the table. The show_name, rating, timing, tags, screenNumber, seats, and price columns are marked as not null, ensuring that all shows have this information. The amount_received column is not null, indicating the total amount received from all bookings for the particular show. Lastly, the created_at column is marked as not null, indicating the date and time the show was added to the table.

The ShowCreated table schema includes three columns, namely sc_id, cshow_id, and cvenue_id. The sc_id serves as the primary key and is auto-incremented for every new show created. The cshow_id and cvenue_id columns are marked as foreign keys and are not null, referencing the show_id and venue_id columns, respectively. This table tracks which show is created at which venue.

The User table schema comprises four columns, namely id, user_name, mobile, city and password. The id serves as the primary key and is auto-incremented for every new user added to the table. The user_name and mobile columns are marked as unique and not null, ensuring that no two users share the same name or mobile number. Lastly, the password column is marked as not null to ensure that every user account has a secure password.

The Showbooked table schema includes nine columns, namely sb_id, seats_booked, showname, showtime, venuename, venueplace, venuelocation, total_price, bshow_id, and buser_id. The sb_id serves as the primary key and is auto-incremented for every new show booked. The seats_booked, showname, showtime, venuename, venueplace, and venuelocation columns are marked as not null, ensuring that all show bookings have this information. The total_price column is also marked as not null, indicating the total price paid for the booking. The bshow_id and buser_id columns are marked as foreign keys and are not null, referencing the show_id and user_id columns, respectively. This table tracks which user booked which show.

The Rate table schema contains the details of the ratings and reviews given by users for specific shows. It has columns such as rating, review, show_name, rshow_id and ruser_id. These columns are used to store the rating given by the user, the review text, the name of the show, and the IDs of the show and user who gave the rating. This information can be used to improve the quality of the shows and provide better recommendations to users.

API design:

The same table schema has been implemented for CRUD operations with API endpoints. This allows for easy manipulation of data using HTTP requests such as GET, POST, PUT, and DELETE. By providing a standardized way to interact with the data, the API ensures that the database is always in sync with the application's needs. This makes it easier to maintain and update the application over time.

Architecture and features:

The project is organized using the Model-View-Controller (MVC) architecture. The controllers are in the project folder named as app.py and responsible for handling user requests and updating the model accordingly. The templates are stored in the templates folder and are used to display the data to the user. The models are used to manage the data and interact with the database.

The system has implemented default features such as login authentication, filtering and search capabilities, and booking management. Additional features such as rating and reviewing shows, generating reports on revenue.. These features have been designed to enhance the user experience and provide administrators with insights to their revenue streams. The features have been implemented using effective approaches to ensure a seamless and secure platform for users to search and book shows.

Click the link to watch the Video: **project video link**