Movie Theatre Reservation System

Kartik Aggarwal, Soumyo Dey, Faiz Khwaja
Khoury College of Computer Science, Northeastern University, Boston, USA
{aggarwal.kart,dey.soum,khwaja.f}@northeastern.edu

Abstract

In this project, MySQL database and Flask framework were employed to build a website for movie theater reservation systems. Watching movies is a stress buster for most people after a very hectic day. Standing in long queues for booking tickets is even worse. In order to solve this problem, the website that we have built allows users to easily view any movie or schedule details and further book the ticket of their choice online. Users can also cancel their ticket at any time. Our project also assists the admin to make changes to the schedule and movies. The admin can also view the details of users who have booked tickets and finally analyze and visualize the revenue generated based on various factors, like revenue generated per date or revenue generated per movie, and thus make important decisions for overall business growth.

Introduction

Our project consists of two independent views-Customer view and Admin view. In the customer view, the customer can register and get their own personal dashboard, where they can view details of the schedule that provides information about specific movies and their time slots. Moreover, they can also view which actors are in particular movies. Finally, they can book their desired tickets and can further cancel their booking if required. In the admin view, the admins have their own personal dashboard, where they can add new movies, view details of users who have booked their tickets, update status of movies and examine the revenue generated per date or movie.

A stored procedure is a code that can be saved and reused over and over again. We used five stored procedures in this project because there are many repetitive tasks like booking a ticket that must be performed in the movie reservation system, so we used stored procedures to save time.

Views are virtual tables that are used to increase SQL security by displaying only the necessary information to the user. Since our database consists of a lot of extra columns that may be confidential or not useful to the user, views can help in hiding such columns.

A function is similar to a stored procedure, the only difference being that it is mandatory for a function to return a value. Functions are mainly used to calculate something from a given input. We used functions to return values like revenue generated.

A trigger is a series of activities that are carried out in reaction to an insert, update, or delete operation on a table. We needed a trigger for our database to ensure certain actions, such as booking confirmation.

A transaction is a logical unit of work that consists of a series of operations executed on a database using one or more SQL statements. All of the SQL statements in a transaction can be committed, or all of them can be rolled back. We used transactions so that there is no partial booking or cancellation.

Database Design

A database is a collection of data items. For this project, we used a relational database because we wanted a pre-defined relationship between the data items. These items are laid out in a table format with columns and rows. Tables store data about the things that will be represented in the database. A field keeps the actual value of an attribute, while each column in a table holds a specific type of data. The table's rows indicate a group of linked values for a single item or entity. A primary key is a unique identifier assigned to each row in a table, and foreign keys are used to link rows from different tables. Without restructuring the database tables, this data can be accessed in a variety of ways.

In our database, we have seven main tables, which are movie, booking, actor, director, schedule, users and confirmation. We also have 2 relations namely actor_in_movie and director_in_movie. The schema diagram of our database is given below.

The two most important tables in our database are users and movie.

The users table consists of five columns, namely: id, email, username, password and admin check (which checks if the user is an admin or not). The primary key of this table is id.

The movie table consists of six columns, namely: movie_id, movie_name, movie_status, movie_release_year, movie_genre and movie_rating. The primary key of this table is movie_id.

`

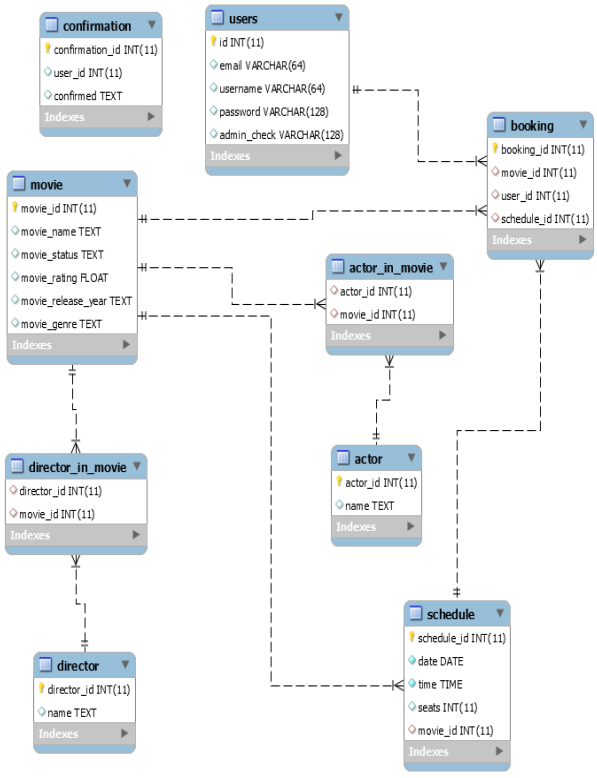

Fig 1. Schema Diagram

Other than these tables, the schedule table consists of five columns, namely: schedule_id, date, time, seats, movie_id. The primary key of this table is schedule_id and the foreign key is the movie_id with reference to the movie table.

The booking table consists of four columns, namely:

booking_id, movie_id, user_id, schedule_id. The primary key of this table is booking_id and the three foreign keys of this table are movie_id, user_id and schedule_id referring to movie, user and schedule table.

The actor table consists of two columns, namely:

actor_id and name. The primary key of this table is actor_id.

The director table consists of two columns, namely:

director_id, and name. The primary key of this table is director_id.

The confirmation table consists of three columns, namely: confirmation_id, user_id and confirmed. The primary key of this table is confirmation_id.

Apart from these we have two many to many relational tables, namely: actor_in_movie and director_in_movie. The actor_in_movie consists the primary keys of actor table(actor_id) and movie table(movie_id) which helps in making a relation between the actor and movie table. Similarly, the director_in_movie table consists of the primary keys of director table (director_id) and movie_table (movie_id) which helps in making a relation between the director and movie table.

In the movie table, movie_name, movie_rating, movie_release_year, and movie_genre and from the actor table, name and from the director table, name were populated using data from IMDB.com/chart/top, where the top 250 movie details were obtained using web scraping. The 'Request' and 'bs4' Python packages were used for the same. Schedule, user and booking table is populated using dummy data.

Our database also consists of three views, five procedures, three functions, two transactions and a trigger. Our database also consists of three views, five procedures, three functions, two transactions, and a trigger.

The three views we created are:

1. The display_actors view provides information about the actors and their movies.
2. The display_schedule view gives the user information about the current schedule of movies that are still playing at the theater.
3. The display_user view provides the administrator with basic information about the users.

The five procedures are as follows:

1. The "BOOKING" procedure gives the user details about all their bookings.
2. The "create_show" procedure helps the administrator add shows into the schedule table.
3. The "delete_status" helps the administrator change the status of movies in the movie table.
4. The "login_proc" procedure inserts the data into the "users" table for any new registration.
5. The "update_status" procedure is used by the administrator to change the movie_status to "active" by entering the movie_id.

The three functions used are:

1. The "active_mov" function returns the number of active movies currently.
2. The "revenue_at_particular_day" function returns the revenue collected according to the date provided by the admin.
3. The "revenue_of_movie" function returns the amount of revenue collected by a particular movie specified by the administrator.

The two transactions performed in the database are:

1. The "add_booking_transaction" decrements the value of the number of seats available for the show the user has booked a ticket for and also inserts the booking details into the booking table.
2. The "cancellation" increments the value of the number of seats available for the show the user has booked a ticket for and deletes the corresponding booking details from the booking table.

The only trigger we created performs insertion on the confirmation table after every new booking detail is inserted into the booking table.

Application Description

In order to create a fully functional website, we have integrated the Flask API in Python with our MySQL database.

Internal Working of the Application

The Flask API, which provides the user interface for the website, was integrated with the MySQL database using the MySQL connector. The MySQL connector is a Python driver for communicating with MySQL servers. After integrating with the MySQL database, the tables in the database were created using Flask. Each database table is defined as a class in Flask. Once these classes were defined in Flask, these changes were committed in order to obtain the tables in the MySQL Workbench.

Since most of the computations in the database were done using SQL queries, and the results of these computations were obtained in Python, in order to display the output of these computations to the user on the webpage, Flask was integrated with the Jinja 2 template engine.

Jinja 2 is a templating language that consists of variables and programming logic that are replaced with actual values when rendered into HTML. Between tags or delimiters, variables and/or logic are placed. For example, Jinja templates employ {% ... %} for expressions and logic (such as loops), while {{...}} is used to show the end user the outcomes of an expression or a variable. When the latter tag is rendered, it is substituted with a value or values that the end user sees. Jinja Templates are nothing more than.html files.

In a Flask project, they are usually found in the /templates directory. Overall, the application was built with the Python Flask API and MYSQL to store various types of information in the database tables, as well as the Jinja 2 template and HTML for the user interface.

While taking inputs from the end users to perform different actions, forms are required. For example, while booking a movie ticket, our application requires the user to input their username and schedule ID, whereas to cancel a booked ticket, the user has to enter the "Booking ID" and the "User ID." In order for the user to be able to view these forms and enter information, we have employed WTforms, which is a flask extension to build forms.

The stored procedures, transactions, and functions in our application require some input arguments. These input arguments are taken in real time by the users using WTF-forms. Once the user fills out the form fields, the values entered by the users are extracted and given as input to these stored procedures, transactions, and functions. The values returned are then displayed on the UI using the Jinda template and HTML. The stored procedures, functions, and transactions are called from the python script after the connection is made with the MySQL database using the Python MySQL Driver.

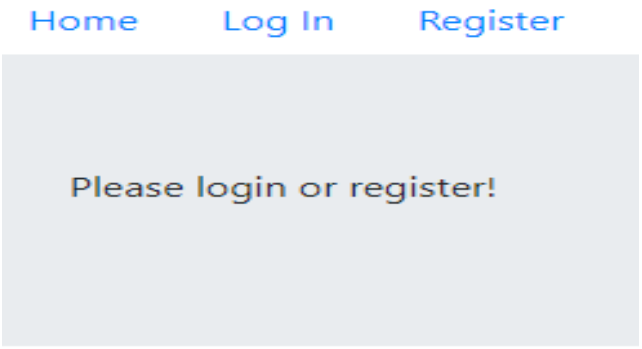

Fig 2. Welcome Page

When the website is accessed for the first time, three buttons are visible to the user. The buttons are the "Home" button, which allows users to redirect to the home page, the "Log in" button, which allows already registered users or admin to log in, and the "Register" button, which allows first-time users to create an account.

User Interface Components

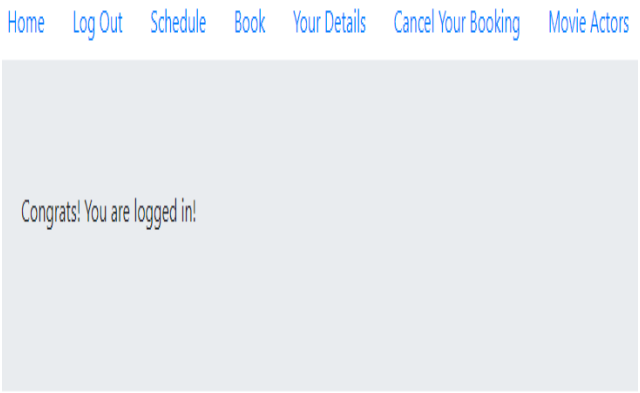

Fig 3. User Home Page

Once the user is logged in, they can see the "Log Out" button which enables them to log out of their account, the "Schedule" button which shows users the schedule of different active movies, the columns of the schedule table are schedule id, movie_name, date, time. The "Book" button which enables the user to book the ticket by entering the schedule ID and their username. Total number of seats initially for each movie is taken to be 100. The total seats remaining gets subtracted each time a ticket is booked by the user. The "Your Details" button which allows users to see their booking details by entering their username, and the "Cancel Your

`

Booking" button which allows users to cancel their booking by entering the "Booking ID" and "Schedule ID". Once a ticket is cancelled by the user, number of seats remaining for the movie gets added. Finally, the "Movie Actors" button allows users to view the actors in the movies.

The admins have their own personal dashboard, which they can access upon entering their admin credentials. Upon logging in, the admin can see the "Add Movie" button, which enables the admin to add movies to the schedule by entering the movie_id, the date and time at which the movie will be played in the theatre. After adding the movie to the schedule, the admin has to update the movie status to "active" using the "Update Status" button so that the users can view the available movies and their schedules on the user dashboards. It is used to update details of the new movies on the schedule.
The admin can also see the "Display-Users" button, which gives information about all the users that have booked the tickets. It shows the username and the movie name of the movie for which the ticket has been booked.

The "Delete Status" button enables the admin to delete the "active" status of the movies, which would in turn remove the details of the movies from the schedule dashboard of the users. It essentially aids in the removal of films that are no longer playing in theaters.

The "Show Active Count" button shows the number of active movies playing in the theatre to the admin. It assists the admin in keeping count of all the active movies. The "Show Revenue" button shows the revenue generated per day. All the admin has to do is enter the date on which the revenue needs to be found. This enables the administrator to keep track of the amount of revenue generated every day and thus help analyze it. The "Movie Revenue" button shows the revenue generated per movie. Upon entering the movie name, the admin can find out how much revenue each movie has made.

Finally, the "Visualize" button shows admin the histogram of the top 250 movies in an interval of 20 years from 1920 to 2020. Moreover, it also shows the pie chart of the most popular genres.

Fig 4. Admin Home Page

Data Analysis

Data analysis is the process of cleaning, inspecting, manipulating, and modeling data in order to extract usable information, draw conclusions, and aid decision-making. In the business world, data analysis is critical for understanding challenges and exploring data in meaningful ways. Data is nothing more than numbers and facts. Data analysis is important in today's business environment since it helps businesses make more scientific decisions and run more efficiently.

There are various types of data analysis that can be performed depending upon what we are trying to achieve. These various types of data analysis are as follows:

1. Descriptive analysis - This is the process of looking back at previous data and deriving conclusions from it. The most typical sort of data analysis is this. This is commonly used to track Key Performance Indicators (KPIs), income, and sales leads, among other things.
2. Diagnostic Analysis - The goal of diagnostic analysis is to figure out why something happened. Once your descriptive analysis has revealed that something negative or good occurred, you can do a diagnostic analysis to determine the cause.
3. Predictive Analysis - Predictive Analysis is concerned with anticipating what will most likely occur in the future. Trends are derived from historical data and utilized to make predictions about the future in this type of research.
4. Prescriptive Analysis - This requires using the knowledge gleaned from the previous three forms of data analysis to develop a strategy for the company to address the problem or make a decision. It is here that data-driven decisions are made.

As a result, the administrator received basic information on the movie theater's performance to assist them in analyzing company demands and future goals. The genre distribution of the movies presented by the movie theater was one of the insights we supplied to the administrator. Figure 5 depicts the above-mentioned distribution as a pie chart. We gathered all of the movies' genres from the movie table and organized them by genre name.

The other insight we showed for the admin included data on the quantity of movies released each year. A histogram was used to illustrate this data. Figure 6 shows the histogram that was used to visualize the previously mentioned data. It is plotted by count of movies per year.

To create the data visualization plots in the admin application we have employed matplotlib and seaborn libraries of python. Matplotlib was used to plot the pie chart and the histogram was plotted using seaborn. These visualization plots

are further rendered to the html so that the admins can view them in their personal dashboard.

The data needed to plot the visualization plot is obtained from our database present in the MySQL workbench. Since data is obtained in real time, the plots will be dynamic, that is, they will change when the data changes.

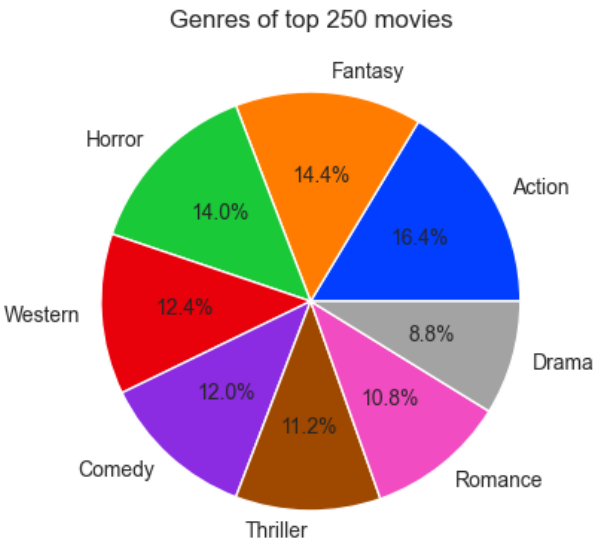

Fig 5. Pie Chart

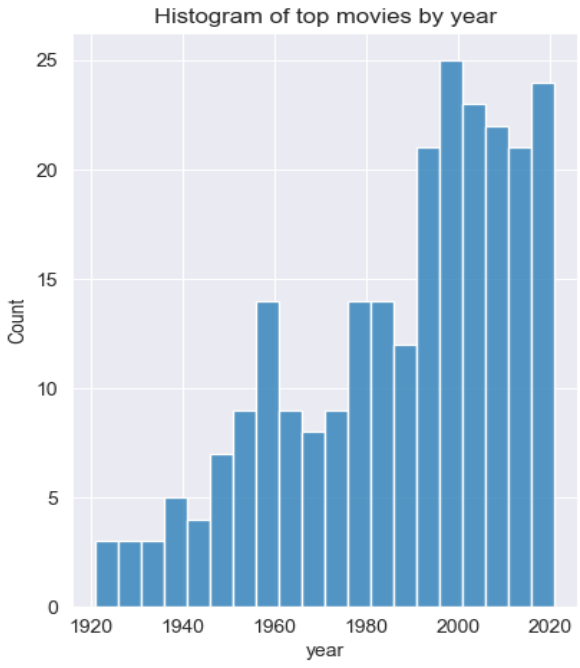

Fig 6. Histogram

Conclusion

Summary of the results

We did it. The website looks great and its working!!
In this project, we were able to successfully create a movie reservation system database along with a supporting graphical user interface (GUI). This GUI is a website that helps the user easily view the schedule of an on-going movie and book tickets. Our website also allows the user to cancel their ticket if required. Moreover, the website allows the admin to add movies to the schedule and get information about the users who have already booked tickets. Furthermore, the website gives the admin useful insights about the business. The admin can view the total revenue generated each day as well as the revenue generated by each movie. The user interface has made it easier for the user and the admin to make changes and retrieve information from the database by providing different tabs and buttons on the user interface which on clicking calls different stored procedures, functions or display views without writing any query in the My SQL workbench.

Learnings

During this course of the project, we learned how to use MY SQL to design and maintain a database. By employing the flask API in Python, we further explored how to integrate a database with a user interface. We mastered how to use Python libraries to plot and visualize different plots, like histograms and pie chart in the user interface.

Future Scope

This project holds great potential for future improvements. There are some gaps that need attention. Since we have provided only two visualizations available to the admin, there could be more data analysis insights that can be integrated into the GUI for the admin. The visualization tab GUI can be improved by providing the admin the option to choose from various insights that he wants to see. Since we have populated the tables with dummy data, we are not capable of providing some insights that could have been possible with real-life data. In our database, we could also add more tables related to the user's payment information and also integrate with a payment portal to allow users to pay online. Currently, we are assuming that every ticket costs the same amount, but in the future, we can also add the feature to provide every show with its own ticket cost. We are also assuming that our theater only has one screen, but in the future, we might incorporate a multiple-screen system into our database.

`

References

1. https://www.imdb.com/chart/top
2. https://www.sqlshack.com/learn-mysql-the-basics-of-mysql-stored-procedures/
3. https://www.guru99.com/functions.html
4. https://www.mysqltutorial.org/mysql-views-tutorial.aspx
5. https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html
6. https://www.mysqltutorial.org/mysql-transaction.aspx
7. https://flask.palletsprojects.com/en/2.0.x/
8. https://flask-login.readthedocs.io/en/latest/
9. https://flask.palletsprojects.com/en/2.0.x/patterns/wtforms/
10. https://seaborn.pydata.org/
11. https://www.kite.com/python/docs/matplotlib
12. https://www.w3schools.com/TAGS/default.ASP
13. https://jinja2docs.readthedocs.io/en/stable/
14. https://programmer.help/blogs/many-to-many-relationship-of-database-in-flask.html/

GitHub Link

1. **https://github.com/SoumyoDey/Movie-Theatre-Reservation-System**
2. **https://github.com/kartikagg/Movie-theater-reservation-system**
3. **https://github.com/faiz2399/Movie-Reservation-System**