Mini project report on

Movie Ticket Booking System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE21CS351A – DBMS Project

Submitted by:

Mahamad Sakeeb M Gadyal
PES2UG21CS266

Mayur Patil
PES2UG21CS286

Dr. Mannar Mannan J
Assistant Professor
Designation
PES University

AUG - DEC 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Movie Ticket Booking System

is a bonafide work carried out by

Mahamad Sakeeb Gadyal	**PES2UG21CS266**
Mayur Patil	**PES2UG21CS286**

In partial fulfilment for the completion of fifth semester DBMS Project (UE21CS351A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2023 – DEC. 2023. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature

Dr. Mannar Mannan J

Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Movie Ticket Booking System** has been carried out by us under the guidance of **Dr. Mannar Mannan J , Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2023.

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Mannan Mannar, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE21CS351A - DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor, ChairPerson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The "Movie Ticket Booking System" mini-project is a web-based application created with Streamlit and MySQL, aiming to optimize the efficiency and user experience of movie ticket management. It incorporates user authentication, ticket booking, and cancellation features. The database design adheres to the Entity-Relationship (ER) model, ensuring a well-structured schema. MySQL is utilized for backend operations, employing Data Definition Language (DDL) and Data Manipulation Language (DML) statements. Various SQL queries, stored procedures, and functions are implemented to facilitate data retrieval, manipulation, and automation. The Streamlit-based front end offers an intuitive and responsive interface, addressing existing limitations for a modern movie ticket management solution.

The Movie Ticket Booking System is a comprehensive Database Management System (DBMS) project designed to streamline and automate the process of booking movie tickets. The system aims to provide a user-friendly and efficient platform for both customers and theater administrators to manage movie bookings, seat allocations, and related transactions.

The Movie Ticket Booking System DBMS project aims to enhance the efficiency and convenience of the movie ticket booking process for both users and administrators. The utilization of a well-designed database ensures data accuracy, security, and scalability, contributing to an improved overall cinema experience.

TABLE OF CONTENTS

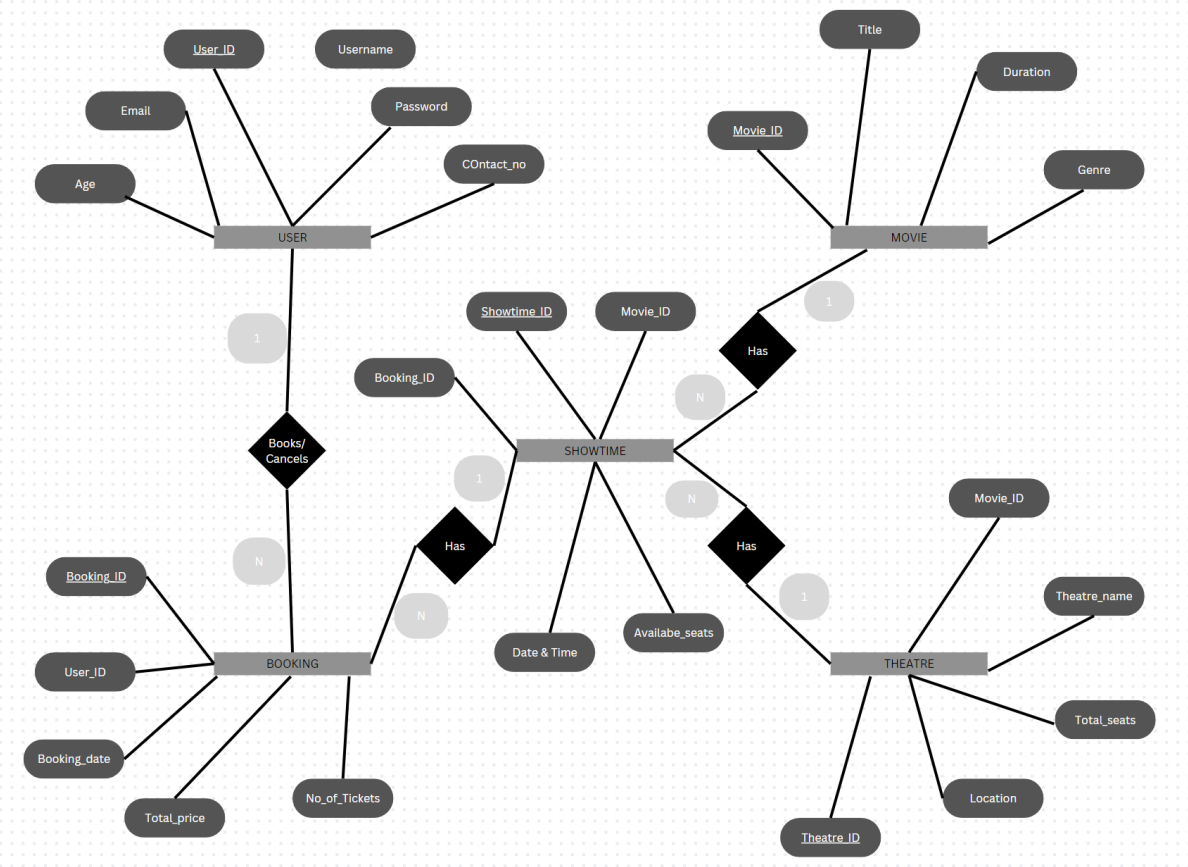
1.INTRODUCTION:

The Movie Ticket Booking System is a web-based application developed with Streamlit for the front end and MySQL for the backend. Focused on delivering a seamless user experience, the system facilitates effortless movie ticket reservations, cancellations, and user feedback. Key features include user authentication, movie search, ticket booking, cancellation, and a movie rating system.

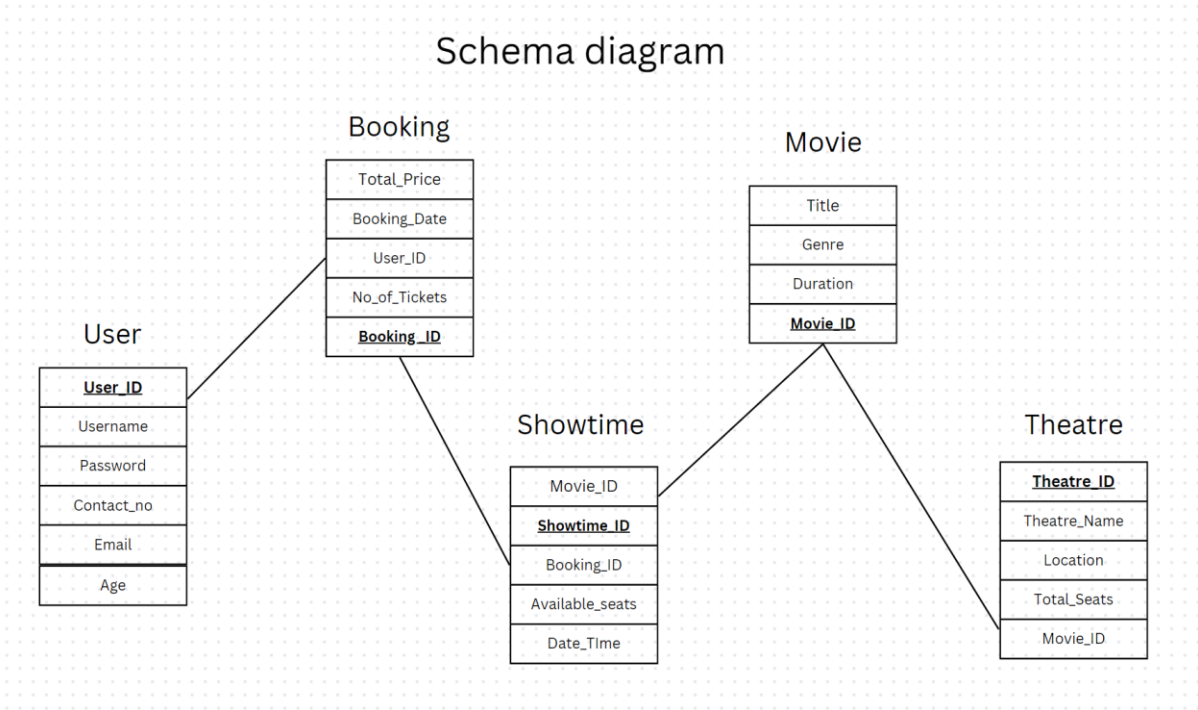
2. PROBLEM DEFINITION:

The Movie Ticket Booking System project addresses the challenges inherent in the movie ticket reservation process, focusing on enhancing the experience for both users and administrators. It aims to streamline the complexities associated with movie ticket bookings, management, and user feedback, addressing any discrepancies encountered. The project's goal is to modernize and optimize the movie ticket booking operations, ultimately improving the overall movie-going experience for users while providing a scalable solution to meet the evolving needs of the entertainment industry.

3.ER MODEL:



4.ER TO RELATIONAL MAPPING:



4.1 STEPS OF ALGORITHM FOR CHOOSEN PROBLEM:

User Registration and Authentication:
Users, including customers and administrators, register securely with unique credentials.

User Login:
Registered users log in securely using their credentials.
Validate and authenticate user information during the login process.

Movie Availability and Booking:
Users can check movie availability for specific dates, select showtimes, and view pricing.
Provide options for users to choose seats, select movie preferences, and confirm bookings securely.

Booking Management:
Enable administrators to view and manage movie reservations.
Confirmation and cancellation of bookings should be handled efficiently.

Payment Processing:
Implement secure payment processing with various options (credit/debit cards).

5.DDL STATEMENTS:


```
create database movie_ticket;
use movie_ticket;

CREATE Table users(
User_ID varchar(5),
First_Name varchar(15),
Last_Name varchar(20),
Email_ID varchar(30),
Age int,
Phone_Number varchar(10) NOT NULL,
Primary Key(User_ID));


Create Table Theatre(
Theatre_ID varchar(5),
Name_of_Theatre varchar(30) NOT NULL,
No_of_Screens int,
Area varchar(30),
Primary Key(Theatre_ID));


CREATE TABLE Screen(
Screen_ID varchar(5),
No_of_Seats_Gold int NOT NULL,
No_of_Seats_Silver int NOT NULL,
Theatre_ID varchar(5),
Primary Key(Screen_ID),
Foreign Key(Theatre_ID) REFERENCES Theatre(Theatre_ID) ON DELETE CASCADE ON UPDATE CASCADE);
```

```
Create Table Movie(
Movie_ID varchar(5),
Name varchar(30) NOT NULL,
Language varchar(10),
Genre varchar(20),
Target_Audience varchar(5),
Primary Key(Movie_ID));


CREATE Table Shows(
Show_ID varchar(10),
Show_Time time NOT NULL,
Show_Date date NOT NULL,
Seats_Remaining_Gold int NOT NULL CHECK (Seats_Remaining_Gold >= 0),
Seats_Remaining_Silver int NOT NULL CHECK (Seats_Remaining_Silver >= 0),
Class_Cost_Gold int NOT NULL,
Class_Cost_Silver int NOT NULL,
Screen_ID varchar(5) NOT NULL,
Movie_ID varchar(5) NOT NULL,
Primary Key(Show_ID),
Foreign Key (Screen_ID) REFERENCES Screen(Screen_ID) ON DELETE CASCADE ON UPDATE CASCADE,
Foreign Key (Movie_ID) REFERENCES Movie(Movie_ID) ON DELETE CASCADE ON UPDATE CASCADE);


CREATE Table Booking(
Booking_ID varchar(10),
No_of_Tickets int NOT NULL,
Total_Cost int NOT NULL,
Card_Number varchar(19),
Name_on_card varchar(21),
User_ID varchar(5),
Show_ID varchar(10),
Foreign Key (User_ID) REFERENCES users (User_ID) ON DELETE CASCADE ON UPDATE CASCADE,
Foreign Key (Show_ID) REFERENCES Shows(Show_ID) ON DELETE CASCADE ON UPDATE CASCADE,
Primary Key(Booking_ID));


CREATE Table Ticket(
Ticket_ID varchar(20),
Booking_ID varchar(10),
Class varchar(3) NOT NULL,
Price int NOT NULL,
Primary Key(Ticket_ID),
Foreign Key(Booking_ID) REFERENCES Booking(Booking_ID)ON DELETE CASCADE);
```

6. DML STATEMENTS:

```
Insert into users values
('1000', 'Sakeeb', 'Gadyal', 'sakeeb17@gmail.com', 20, '9682568456'),
('1001', 'Sahil', 'SK', 'sahil8@gmail.com', 20, '9632798647'),
('1002', 'Mayur', 'Patil', 'Mayur3451@gmail.com', 20, '9635472848'),
('1003', 'Virat', 'Kohli', 'virat18@gmail.com', 35, '8635262747'),
('1004', 'Prajwal', 'CA', 'caprajwal@gmail.com', 20, '6852796434'),
('1005', 'Anushaka', 'Sharma', 'Anushaka18@gmail.com', 33, '8624125186'),
('1006', 'Yuzi', 'Chahal', 'yuzi12@gmail.com', 29, '8643521789'),
('1007', 'AB', 'De villiers ', 'ab17@gmail.com', 38, '6842164678'),
('1008', 'Chandan', 'D', 'chandan45@gmail.com', 20, '8412536748'),
('1009', 'Sami', 'S', 'sami87@gmail.com', 20, '9721465368');


Insert into Theatre values
('TH1', 'INOX Movies', 3, 'Jayanagar, Bengaluru South'),
('TH2', 'PVR Cinemas', 2, 'JP nagar , Bengaluru South'),
('TH3', 'Cinepolis', 2, 'Yeswanthpur, Bengaluru North'),
('TH4', 'Inox Lido', 20, 'Bannerghattha, Bengaluru South'), -- Adjusted 'Area' value
('TH5', 'Cinepolis1', 3, 'Yelahankha, Bengaluru North');
Insert into Screen values
('TH11', 40, 60, 'TH1'),
('TH12', 40, 60, 'TH1'),
('TH13', 40, 60, 'TH1'),
('TH21', 36, 64, 'TH2'),
('TH22', 36, 64, 'TH2'),
('TH31', 50, 50, 'TH3'),
('TH32', 50, 50, 'TH3'),
('TH41', 40, 60, 'TH4'),
('TH42', 40, 60, 'TH4'),
('TH51', 50, 50, 'TH5'),
('TH52', 50, 50, 'TH5'),
('TH53', 50, 50, 'TH5');


Insert into Movie values
('001', 'Doctor Strange', 'English','Fantasy/Adventure', 'U/A'),
('002', 'KGF',  'kannada','Action','U/A'),
('003', 'Kantara','Kannada', 'Action/triller', 'U/A'),
('004', 'housefull', 'Hindi','Comedy', 'U/A'),
('005', 'Avengers: Infinity War', 'English','Fantasy/Adventure','U/A'),
('006', 'Jailer','Tamil','Action' ,'U/A');
 Insert into Shows values
('SHTH110001', '09:00:00', '2023-11-17', 40, 60, 400, 350, 'TH11', '001'),
('SHTH110002', '16:00:00', '2023-11-17', 38, 60, 400, 350, 'TH11', '002'),
('SHTH120001', '09:00:00', '2023-11-17', 40, 60, 400, 350, 'TH12', '003'),
('SHTH130001', '09:00:00', '2023-11-17', 40, 60, 400, 350, 'TH13', '004'),
('SHTH210001', '16:00:00', '2023-11-17', 36, 64, 415, 375, 'TH21', '003'),
('SHTH220001', '13:00:00', '2023-11-17', 36, 64, 415, 375, 'TH22', '002'),
('SHTH310001', '09:00:00', '2023-11-18', 50, 50, 480, 380, 'TH31', '002'),
('SHTH310002', '16:00:00', '2023-11-18', 50, 50, 480, 380, 'TH31', '004'),
('SHTH320001', '09:00:00', '2023-11-19', 50, 46, 480, 380, 'TH32', '005'),
('SHTH320002', '16:00:00', '2023-11-19', 50, 50, 480, 380, 'TH32', '006'),
('SHTH410001', '09:00:00', '2023-11-20', 40, 60, 415, 375, 'TH41', '001'),
('SHTH420001', '19:00:00', '2023-11-20', 40, 60, 415, 375, 'TH42', '004'),
('SHTH510001', '19:00:00', '2023-11-20', 50, 50, 480, 380, 'TH51', '002'),
('SHTH520001', '19:00:00', '2023-11-20', 50, 50, 480, 380, 'TH52', '003'),
('SHTH530001', '19:00:00', '2023-11-21', 50, 50, 480, 380, 'TH53', '005');


INSERT into Booking values('BOOK0001', 2, 800, '8249621092163126', 'Sakeeb Gadyal', 1000,'SHTH110002');
INSERT into Ticket values('TIDBOOK0001001', 'BOOK0001', 'GLD',  400);
INSERT into Ticket values('TIDBOOK0001002', 'BOOK0001', 'GLD',  400);

INSERT into Booking values('BOOK0002', 4, 1520, '9261738271340646', 'Virat Kohli', 1001, 'SHTH320001');
INSERT into Ticket values('TIDBOOK0002001', 'BOOK0002', 'SLV',  380);
INSERT into Ticket values('TIDBOOK0002002', 'BOOK0002', 'SLV',  380);
INSERT into Ticket values('TIDBOOK0002003', 'BOOK0002', 'SLV',  380);
INSERT into Ticket values('TIDBOOK0002004', 'BOOK0002', 'SLV',  380);
```

```
INSERT into Booking values('BOOK0003', 4, 1660, '9864821890538268', 'Mayur Patil', 1007, 'SHTH410001');
INSERT into Ticket values('TIDBOOK0003001', 'BOOK0003', 'GLD',  415);
INSERT into Ticket values('TIDBOOK0003002', 'BOOK0003', 'GLD',  415);
INSERT into Ticket values('TIDBOOK0003003', 'BOOK0003', 'GLD',  415);
INSERT into Ticket values('TIDBOOK0003004', 'BOOK0003', 'GLD',  415);

INSERT into Booking values('BOOK0004', 2, 960, '8261723854786968', 'Sahil SK', 1008, 'SHTH520001');
INSERT into Ticket values('TIDBOOK0004001', 'BOOK0004', 'GLD',  480);
INSERT into Ticket values('TIDBOOK0004002', 'BOOK0004', 'GLD',  480);

INSERT into Booking values('BOOK0005', 2, 960, '6826145790463578', 'Prajwal CA', 1009, 'SHTH530001');
INSERT into Ticket values('TIDBOOK0005001', 'BOOK0005', 'GLD',  480);
INSERT into Ticket values('TIDBOOK0005002', 'BOOK0005', 'GLD',  480);
```

6.1 SIMPLE QUERY:

```
def get_showtimes():
    c.execute('SELECT Show_ID, Show_Time, Show_Date FROM Shows',)
    data = c.fetchall()
    return data

def get_theatre_details():
    c.execute('SELECT * FROM Theatre')
    data = c.fetchall()
    return data

def book_movie(Booking_ID, User_ID, Show_ID, No_of_Tickets, Total_Cost):
    c.execute('INSERT INTO BOOKINGS(Booking_ID, User_ID, Show_ID, No_of_Tickets, Total_Cost) '
              'VALUES (%s, %s, %s, %s, %s)',
              (Booking_ID, User_ID, Show_ID, No_of_Tickets, Total_Cost))
    mydb.commit()
```

6.2 UPDATE OPERATION:

```
def edit_user_data(new_User_ID, new_First_Name, new_Last_Name, new_Email_ID, new_Age, new_Phone_Number,
                   User_ID, First_Name, Last_Name, Email_ID, Age, Phone_Number):
    c.execute("UPDATE USERS SET User_ID=%s, First_Name=%s, Last_Name=%s, Email_ID=%s, Age=%s, Phone_Number=%s "
              "WHERE User_ID=%s and First_Name=%s and Last_Name=%s and Email_ID=%s and Age=%s and Phone_Number=%s",
              (new_User_ID, new_First_Name, new_Last_Name, new_Email_ID, new_Age, new_Phone_Number,
              User_ID, First_Name, Last_Name, Email_ID, Age, Phone_Number))
    mydb.commit()
    #data = c.fetchall()
    #return data
```

6.3 DELETE OPERATION:

```
def delete_data(User_ID):
    c.execute('DELETE FROM USERS WHERE User_ID="{}"'.format(User_ID))
```

6.4 CORRELATED QUERY

```
def execute_correlated_query():
    st.subheader("Correlated MySQL Query with IN Operator: Users and Their Bookings")

    cursor = mydb.cursor()

    query = """
        SELECT u.User_ID, u.First_Name, u.Last_Name, u.Email_ID, b.Booking_ID, b.Total_Cost
        FROM Users u
        JOIN Booking b ON u.User_ID = b.User_ID
        WHERE u.User_ID IN (
            SELECT DISTINCT b.User_ID
            FROM Booking b
            WHERE b.Total_Cost > 100  # Condition for the IN operator, change as needed
        )
    """

    cursor.execute(query)
    results = cursor.fetchall()
```

6.5 NESTED QUERY

```
def execute_nested_mysql_query():
    st.subheader("Nested MySQL Query Results: User Bookings with Movie Details")
    cursor = mydb.cursor()
    query = """
        SELECT u.User_ID, u.First_Name, u.Last_Name, u.Email_ID, u.Age, u.Phone_Number,
               b.Booking_ID, b.No_of_Tickets, b.Total_Cost, b.Card_Number, b.Name_on_card,
               s.Show_ID, s.Show_Time, s.Show_Date, s.Seats_Remaining_Gold, s.Seats_Remaining_Silver,
               m.Movie_ID, m.Name AS Movie_Name, m.Language, m.Genre, m.Target_Audience
        FROM Users u
        JOIN Booking b ON u.User_ID = b.User_ID
        JOIN Shows s ON b.Show_ID = s.Show_ID
        JOIN Movie m ON s.Movie_ID = m.Movie_ID
    """

    cursor.execute(query)
    results = cursor.fetchall()
```

7.QUERIES:

```
mysql> show databases;
+----------------------------------+
| Database                         |
+----------------------------------+
| amitdb                           |
| cafe                             |
| company                          |
| db                               |
| fest_database                    |
| final_movie                      |
| information_schema               |
| movie_ticket                     |
| mysql                            |
| performance_schema               |
| pes2ug21cs266_university_fest_db |
| sakeeb                           |
| sys                              |
| university                       |
+----------------------------------+
14 rows in set (0.01 sec)
```

```
mysql> use movie_ticket;
Database changed
mysql> SELECT * FROM users;
+---------+------------+-------------+---------------------+------+--------------+
| User_ID | First_Name | Last_Name   | Email_ID            | Age  | Phone_Number |
+---------+------------+-------------+---------------------+------+--------------+
| 1000    | Sakeeb     | Gadyal      | sakeeb17@gmail.com  |   20 | 9682568456   |
| 1001    | Sahil      | SK          | sahil8@gmail.com    |   20 | 9632798647   |
| 1002    | Mayur      | Patil       | rahul1@gmail.com    |   20 | 9635472848   |
| 1003    | Virat      | Kohli       | virat18@gmail.com   |   35 | 8635262747   |
| 1004    | Prajwal    | CA          | caprajwal@gmail.com |   20 | 6852796434   |
| 1005    | Anushaka   | Sharma      | Anushaka18@gmail.com|   33 | 8624125186   |
| 1006    | Yuzi       | Chahal      | yuzi12@gmail.com    |   29 | 8643521789   |
| 1007    | AB         | De villiers | ab17@gmail.com      |   38 | 6842164678   |
| 1008    | Chandan    | D           | chandan45@gmail.com |   20 | 8412536748   |
| 1009    | Sami       | S           | sami87@gmail.com    |   20 | 9721465368   |
+---------+------------+-------------+---------------------+------+--------------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM screen;
+-----------+-----------------+-------------------+------------+
| Screen_ID | No_of_Seats_Gold | No_of_Seats_Silver | Theatre_ID |
+-----------+-----------------+-------------------+------------+
| TH11      |              40 |                60 | TH1        |
| TH12      |              40 |                60 | TH1        |
| TH13      |              40 |                60 | TH1        |
| TH21      |              36 |                64 | TH2        |
| TH22      |              36 |                64 | TH2        |
| TH31      |              50 |                50 | TH3        |
| TH32      |              50 |                50 | TH3        |
| TH41      |              40 |                60 | TH4        |
| TH42      |              40 |                60 | TH4        |
| TH51      |              50 |                50 | TH5        |
| TH52      |              50 |                50 | TH5        |
| TH53      |              50 |                50 | TH5        |
+-----------+-----------------+-------------------+------------+
12 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM theatre;
+------------+-----------------+---------------+-------------------------------+
| Theatre_ID | Name_of_Theatre | No_of_Screens | Area                          |
+------------+-----------------+---------------+-------------------------------+
| TH1        | INOX Movies     |             3 | Jayanagar, Bengaluru South    |
| TH2        | PVR Cinemas     |             2 | JP nagar , Bengaluru South    |
| TH3        | Cinepolis       |             2 | Yeswanthpur, Bengaluru North  |
| TH4        | Inox Lido       |            20 | Bannerghattha, Bengaluru South|
| TH5        | Cinepolis1      |             3 | Yelahankha, Bengaluru North   |
+------------+-----------------+---------------+-------------------------------+
5 rows in set (0.00 sec)
```

8. STORED PROCEDURES, FUCNTIONS:

```
DELIMITER $$

CREATE FUNCTION CalculateTotalUsers()
RETURNS INT
DETERMINISTIC
READS SQL DATA
BEGIN
  DECLARE totalUsers INT;
  SELECT COUNT(User_ID) INTO totalUsers FROM USERS;
  RETURN totalUsers;
END $$

DELIMITER ;
```

```
DELIMITER //
CREATE PROCEDURE CalculateTotalMoviesNow(OUT total_movies INT)
BEGIN
  SELECT COUNT(*) INTO total_movies FROM Movie;
END;
//
DELIMITER ;


DELIMITER $$

CREATE FUNCTION CalculateTotalBooking()
RETURNS INT
DETERMINISTIC
READS SQL DATA
BEGIN
  DECLARE totalbooks INT;
  SELECT COUNT(Booking_ID) INTO totalbooks FROM booking;
  RETURN totalbooks;
END $$
DELIMITER ;
-- to update the gold and silver seats left when booking a ticket

DELIMITER &&

CREATE PROCEDURE Booking_Seats(IN b_id varchar(20), IN cl varchar(10))
BEGIN
    declare S_ID varchar(20);

    Select Show_ID into S_ID from Booking where Booking_ID = b_id;


    IF cl = 'GLD' THEN
            UPDATE Shows SET Seats_Remaining_Gold = Seats_Remaining_Gold - 1 Where Show_ID = S_ID;
    ELSE
            UPDATE Shows SET Seats_Remaining_Silver = Seats_Remaining_Silver - 1 Where Show_ID = S_ID;
    END IF;
END &&
DELIMITER ;
```

8.1 TRIGGERS:

```
DELIMITER &&
CREATE TRIGGER Adding_Seats
    BEFORE INSERT
    ON Ticket
    FOR EACH ROW
    BEGIN
        CALL Booking_Seats(NEW.Booking_ID, NEW.Class);
    END &&
DELIMITER ;
```

9.LIST OF TABLES & DESCRIPTIONS:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_movie_ticket |
+------------------------+
| booking                |
| movie                  |
| screen                 |
| shows                  |
| theatre                |
| ticket                 |
| users                  |
+------------------------+
7 rows in set (0.05 sec)
```

```
mysql> DESC booking;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Booking_ID    | varchar(10) | NO   | PRI | NULL    |       |
| No_of_Tickets | int         | NO   |     | NULL    |       |
| Total_Cost    | int         | NO   |     | NULL    |       |
| Card_Number   | varchar(19) | YES  |     | NULL    |       |
| Name_on_card  | varchar(21) | YES  |     | NULL    |       |
| User_ID       | varchar(5)  | YES  | MUL | NULL    |       |
| Show_ID       | varchar(10) | YES  | MUL | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
7 rows in set (0.04 sec)
```

```
mysql> DESC movie;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Movie_ID        | varchar(5)  | NO   | PRI | NULL    |       |
| Name            | varchar(30) | NO   |     | NULL    |       |
| Language        | varchar(10) | YES  |     | NULL    |       |
| Genre           | varchar(20) | YES  |     | NULL    |       |
| Target_Audience | varchar(5)  | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> DESC screen;
+-------------------+------------+------+-----+---------+-------+
| Field             | Type       | Null | Key | Default | Extra |
+-------------------+------------+------+-----+---------+-------+
| Screen_ID         | varchar(5) | NO   | PRI | NULL    |       |
| No_of_Seats_Gold  | int        | NO   |     | NULL    |       |
| No_of_Seats_Silver | int       | NO   |     | NULL    |       |
| Theatre_ID        | varchar(5) | YES  | MUL | NULL    |       |
+-------------------+------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> DESC shows;
+-----------------------+-------------+------+-----+---------+-------+
| Field                 | Type        | Null | Key | Default | Extra |
+-----------------------+-------------+------+-----+---------+-------+
| Show_ID               | varchar(10) | NO   | PRI | NULL    |       |
| Show_Time             | time        | NO   |     | NULL    |       |
| Show_Date             | date        | NO   |     | NULL    |       |
| Seats_Remaining_Gold  | int         | NO   |     | NULL    |       |
| Seats_Remaining_Silver | int        | NO   |     | NULL    |       |
| Class_Cost_Gold       | int         | NO   |     | NULL    |       |
| Class_Cost_Silver     | int         | NO   |     | NULL    |       |
| Screen_ID             | varchar(5)  | NO   | MUL | NULL    |       |
| Movie_ID              | varchar(5)  | NO   | MUL | NULL    |       |
+-----------------------+-------------+------+-----+---------+-------+
9 rows in set (0.03 sec)
```

```
mysql> DESC theatre;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Theatre_ID      | varchar(5)  | NO   | PRI | NULL    |       |
| Name_of_Theatre | varchar(30) | NO   |     | NULL    |       |
| No_of_Screens   | int         | YES  |     | NULL    |       |
| Area            | varchar(30) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> DESC ticket;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Ticket_ID  | varchar(20) | NO   | PRI | NULL    |       |
| Booking_ID | varchar(10) | YES  | MUL | NULL    |       |
| Class      | varchar(3)  | NO   |     | NULL    |       |
| Price      | int         | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> DESC users;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| User_ID      | varchar(5)  | NO   | PRI | NULL    |       |
| First_Name   | varchar(15) | YES  |     | NULL    |       |
| Last_Name    | varchar(20) | YES  |     | NULL    |       |
| Email_ID     | varchar(30) | YES  |     | NULL    |       |
| Age          | int         | YES  |     | NULL    |       |
| Phone_Number | varchar(10) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

10.FRONT END DEVELOPEMNT:

Connecting to MySQL:
We have used the mysql-connector-python library to establish a connection to our MySQL database within your Streamlit app. Here's a basic example:

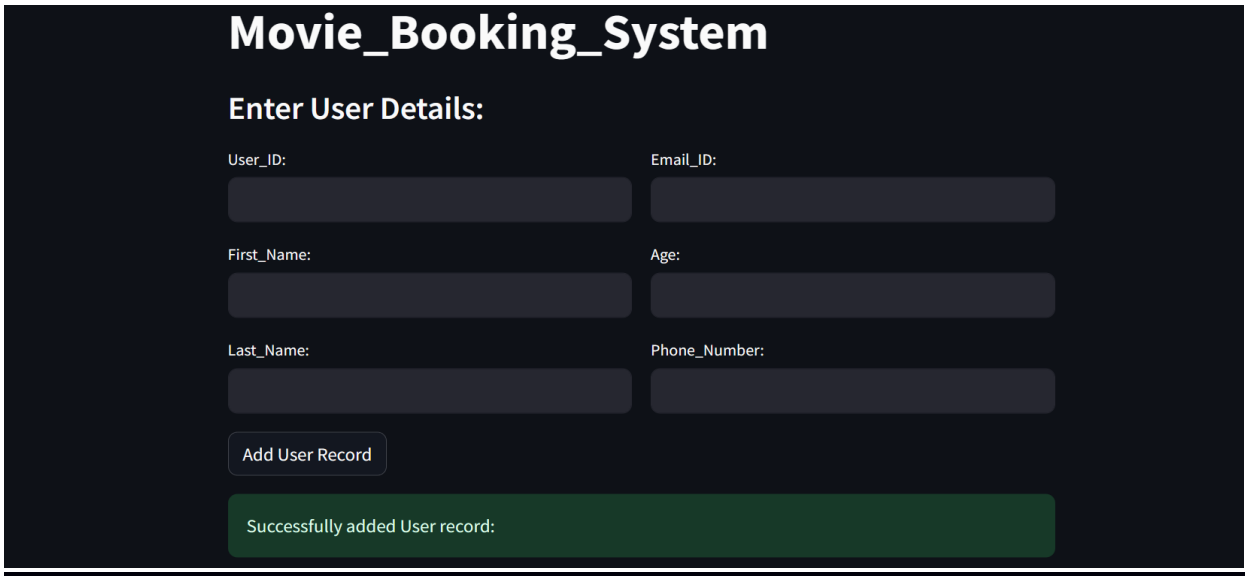
Python Code :

```python
import mysql.connector
mydb = mysql.connector.connect(host="localhost", user="root", password="sakeeb@17", database="movie_ticket")
c = mydb.cursor()
```

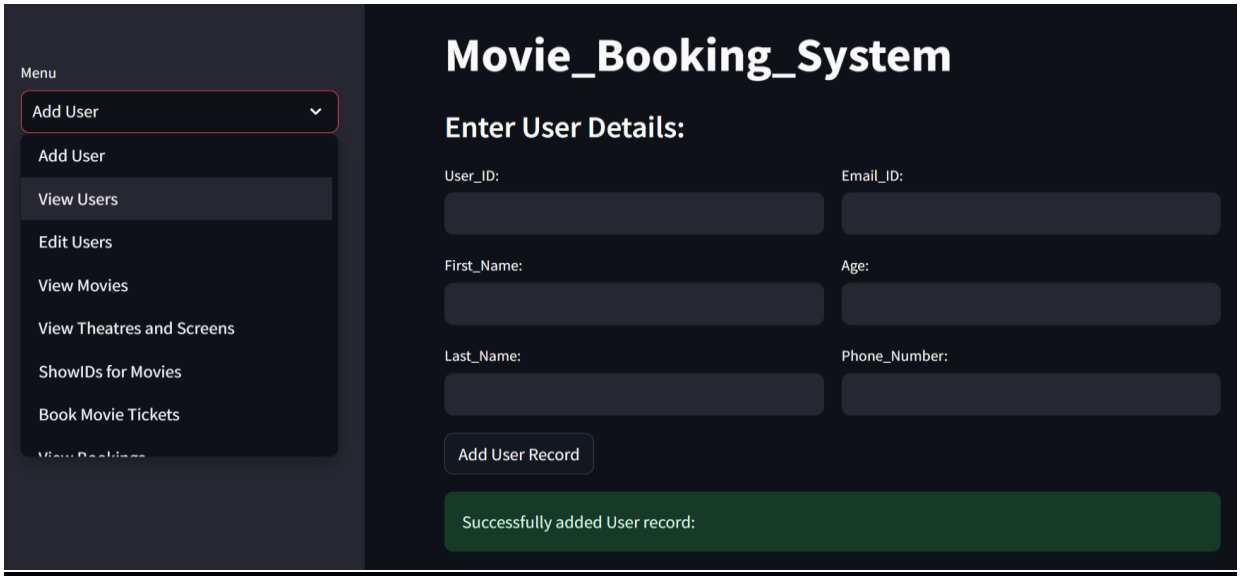
Interact with Database:
Performed database operations in Streamlit by executing SQL queries, fetching data, and displaying it in the app.

CRUD Operations:
We have Integrate Streamlit components like input fields and buttons to create forms for inserting, updating, or deleting data in the MySQL database.

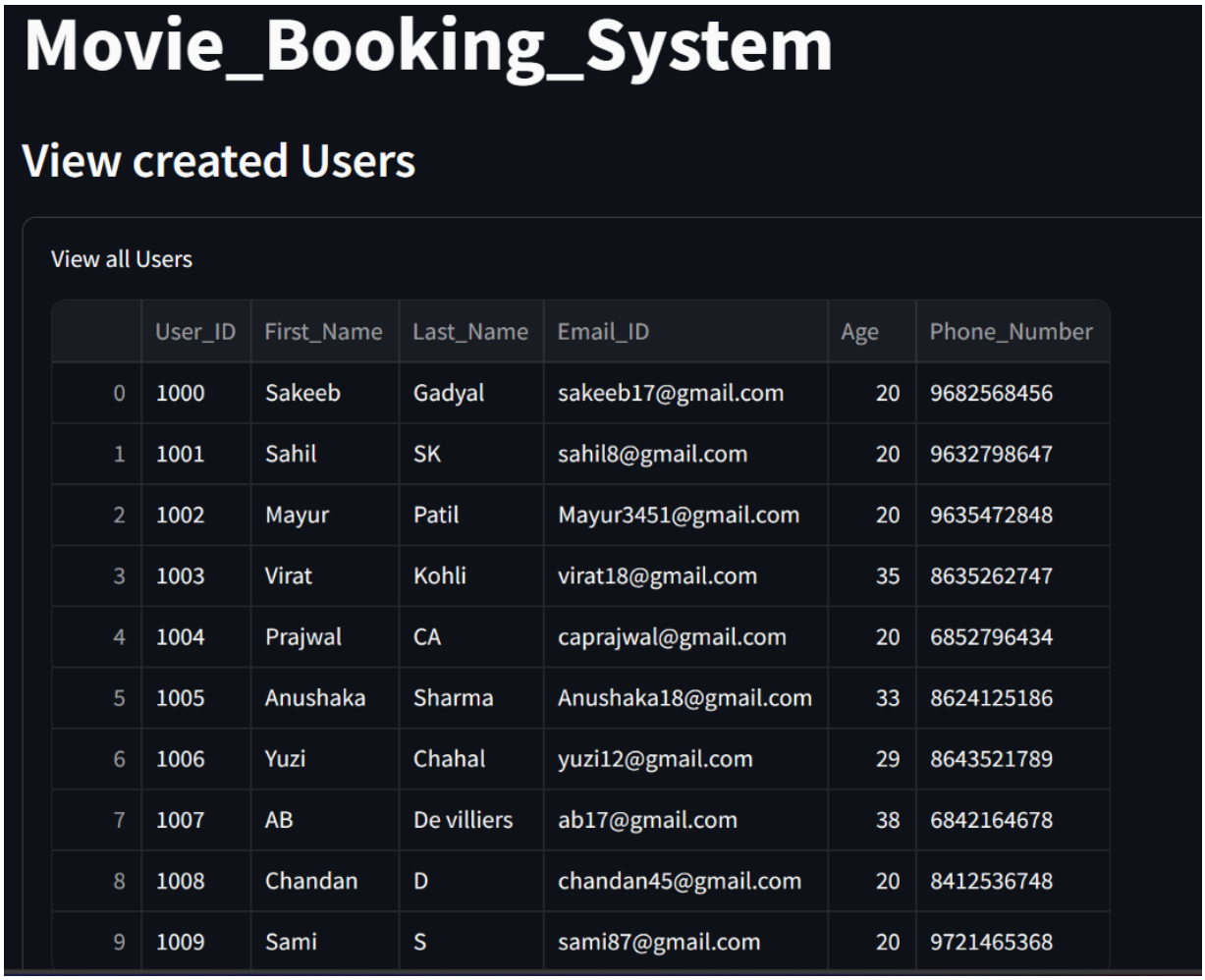
Home Page:



Menu Page:



Admin View:
View Users:

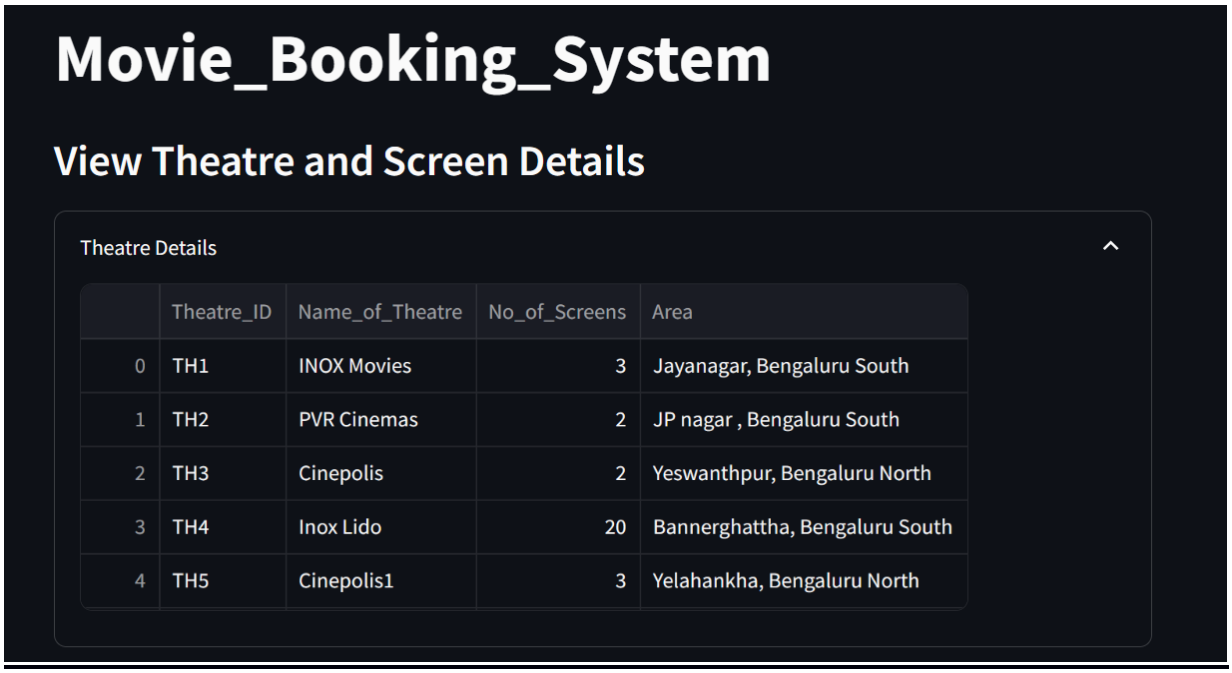


	User_ID	First_Name	Last_Name	Email_ID	Age	Phone_Number
0	1000	Sakeeb	Gadyal	sakeeb17@gmail.com	20	9682568456
1	1001	Sahil	SK	sahil8@gmail.com	20	9632798647
2	1002	Mayur	Patil	Mayur3451@gmail.com	20	9635472848
3	1003	Virat	Kohli	virat18@gmail.com	35	8635262747
4	1004	Prajwal	CA	caprajwal@gmail.com	20	6852796434
5	1005	Anushaka	Sharma	Anushaka18@gmail.com	33	8624125186
6	1006	Yuzi	Chahal	yuzi12@gmail.com	29	8643521789
7	1007	AB	De villiers	ab17@gmail.com	38	6842164678
8	1008	Chandan	D	chandan45@gmail.com	20	8412536748
9	1009	Sami	S	sami87@gmail.com	20	9721465368

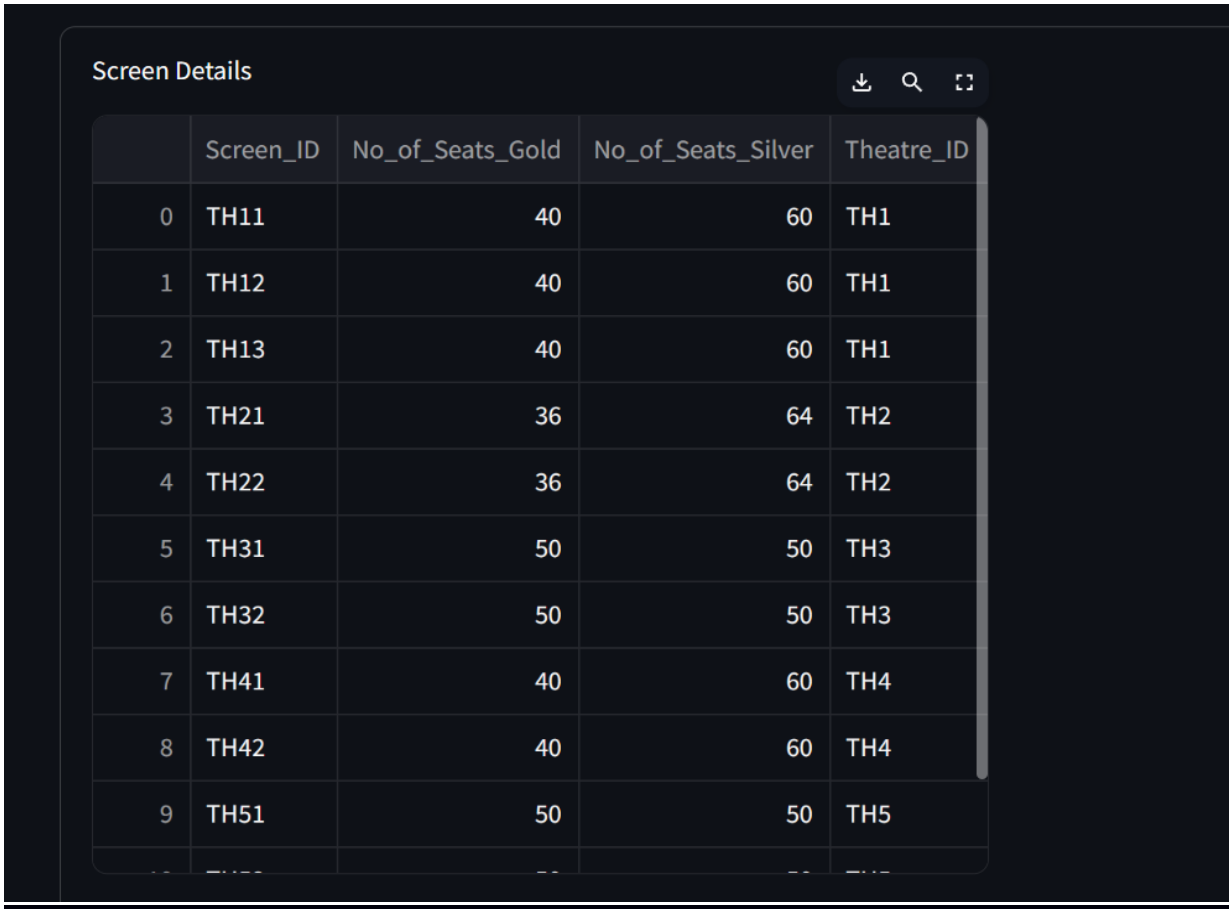
Movies Page:

	Movie_ID	Name	Language	Genre	Target_Audience
0	001	Doctor Strange	English	Fantasy/Adventure	U/A
1	002	KGF	kannada	Action	U/A
2	003	Kantara	Kannada	Action/triller	U/A
3	004	housefull	Hindi	Comedy	U/A
4	005	Avengers: Infinity War	English	Fantasy/Adventure	U/A
5	006	Jailer	Tamil	Action	U/A

Theatres & Screens Available:

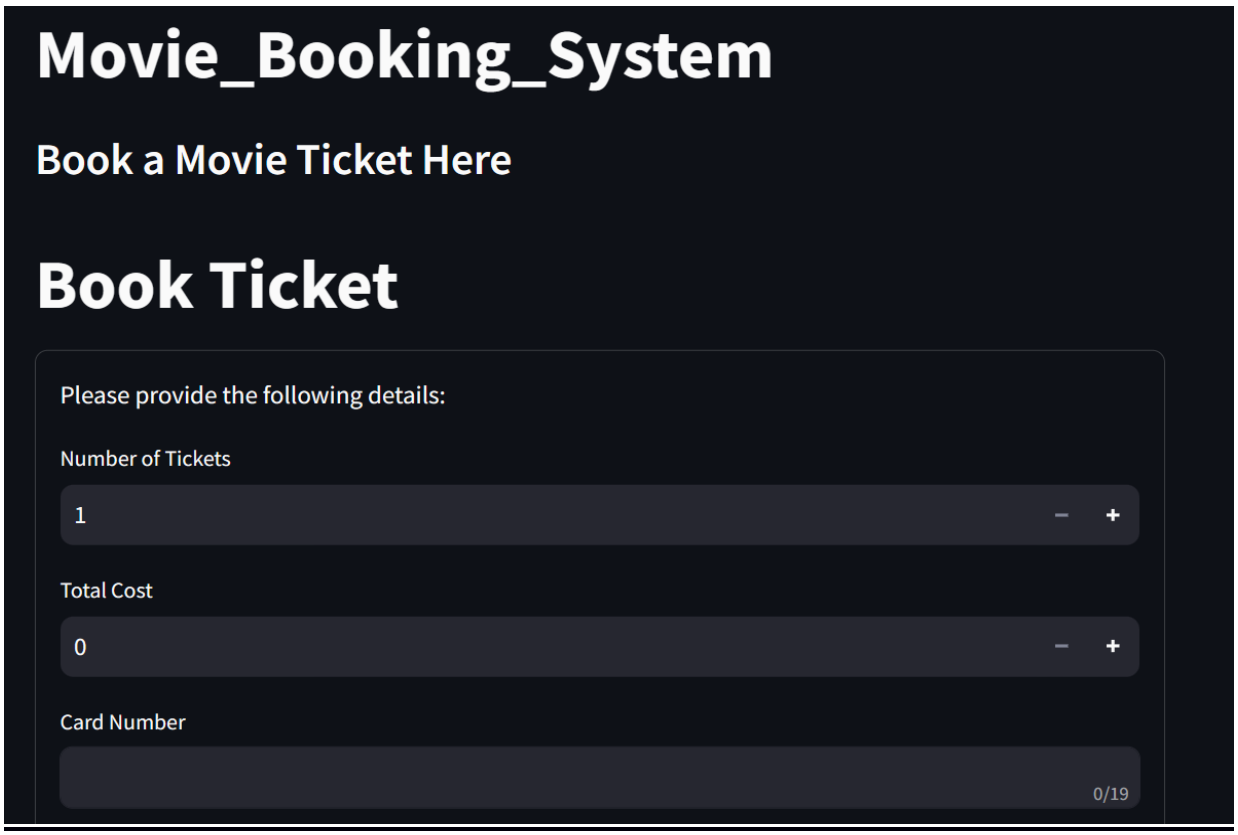


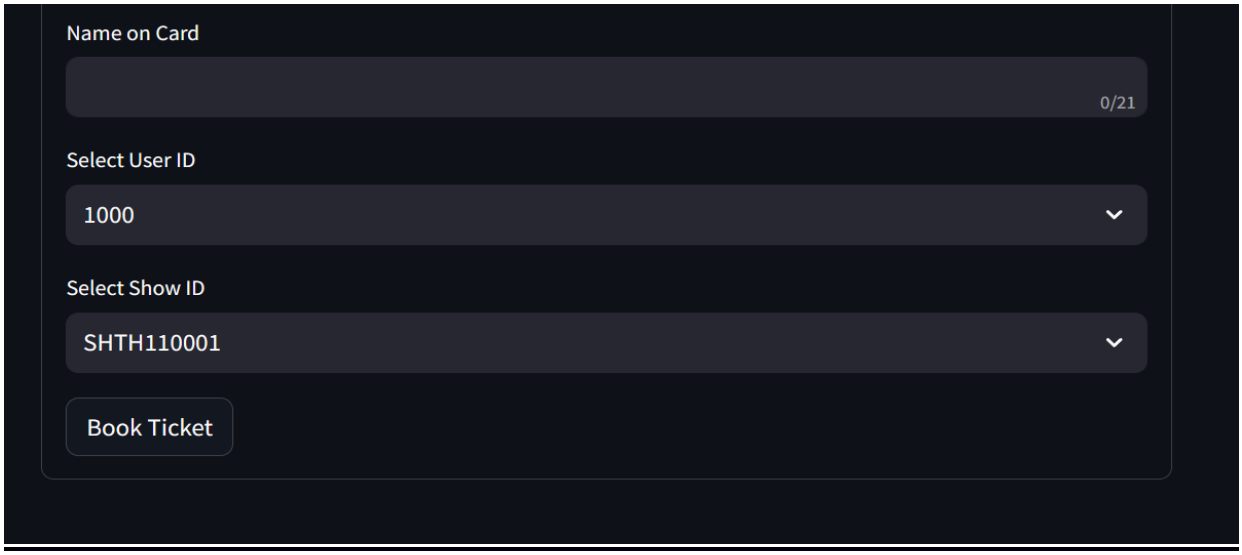
	Theatre_ID	Name_of_Theatre	No_of_Screens	Area
0	TH1	INOX Movies	3	Jayanagar, Bengaluru South
1	TH2	PVR Cinemas	2	JP nagar , Bengaluru South
2	TH3	Cinepolis	2	Yeswanthpur, Bengaluru North
3	TH4	Inox Lido	20	Bannerghattha, Bengaluru South
4	TH5	Cinepolis1	3	Yelahankha, Bengaluru North



	Screen_ID	No_of_Seats_Gold	No_of_Seats_Silver	Theatre_ID
0	TH11	40	60	TH1
1	TH12	40	60	TH1
2	TH13	40	60	TH1
3	TH21	36	64	TH2
4	TH22	36	64	TH2
5	TH31	50	50	TH3
6	TH32	50	50	TH3
7	TH41	40	60	TH4
8	TH42	40	60	TH4
9	TH51	50	50	TH5

Ticket Booking:





Show Details:

Show ID: SHTH120001, Movie Name: Kantara,Show Date: 2023-11-17,Show Time: 9:00:00,Screen ID: TH12

Show ID: SHTH210001, Movie Name: Kantara,Show Date: 2023-11-17,Show Time: 16:00:00,Screen ID: TH21

Show ID: SHTH520001, Movie Name: Kantara,Show Date: 2023-11-20,Show Time: 19:00:00,Screen ID: TH52

Show ID: SHTH130001, Movie Name: housefull,Show Date: 2023-11-17,Show Time: 9:00:00,Screen ID: TH13

Show ID: SHTH310002, Movie Name: housefull,Show Date: 2023-11-18,Show Time: 16:00:00,Screen ID: TH31

Show ID: SHTH420001, Movie Name: housefull,Show Date: 2023-11-20,Show Time: 19:00:00,Screen ID: TH42

Show ID: SHTH320001, Movie Name: Avengers: Infinity War,Show Date: 2023-11-19,Show Time: 9:00:00,Screen ID: TH32

Show ID: SHTH530001, Movie Name: Avengers: Infinity War,Show Date: 2023-11-21,Show Time: 19:00:00,Screen ID: TH53

Show ID: SHTH320002, Movie Name: Jailer,Show Date: 2023-11-19,Show Time: 16:00:00,Screen ID: TH32

Booked User Details:

Users with Bookings:

Booking ID: BOOK0001, Name: Sakeeb Gadyal, Ticket Id: TIDBOOK0001001

Booking ID: BOOK0001, Name: Sakeeb Gadyal, Ticket Id: TIDBOOK0001002

Booking ID: BOOK0002, Name: Virat Kohli, Ticket Id: TIDBOOK0002001

Booking ID: BOOK0002, Name: Virat Kohli, Ticket Id: TIDBOOK0002002

Booking ID: BOOK0003, Name: Mayur Patil, Ticket Id: TIDBOOK0003001

Booking ID: BOOK0003, Name: Mayur Patil, Ticket Id: TIDBOOK0003002

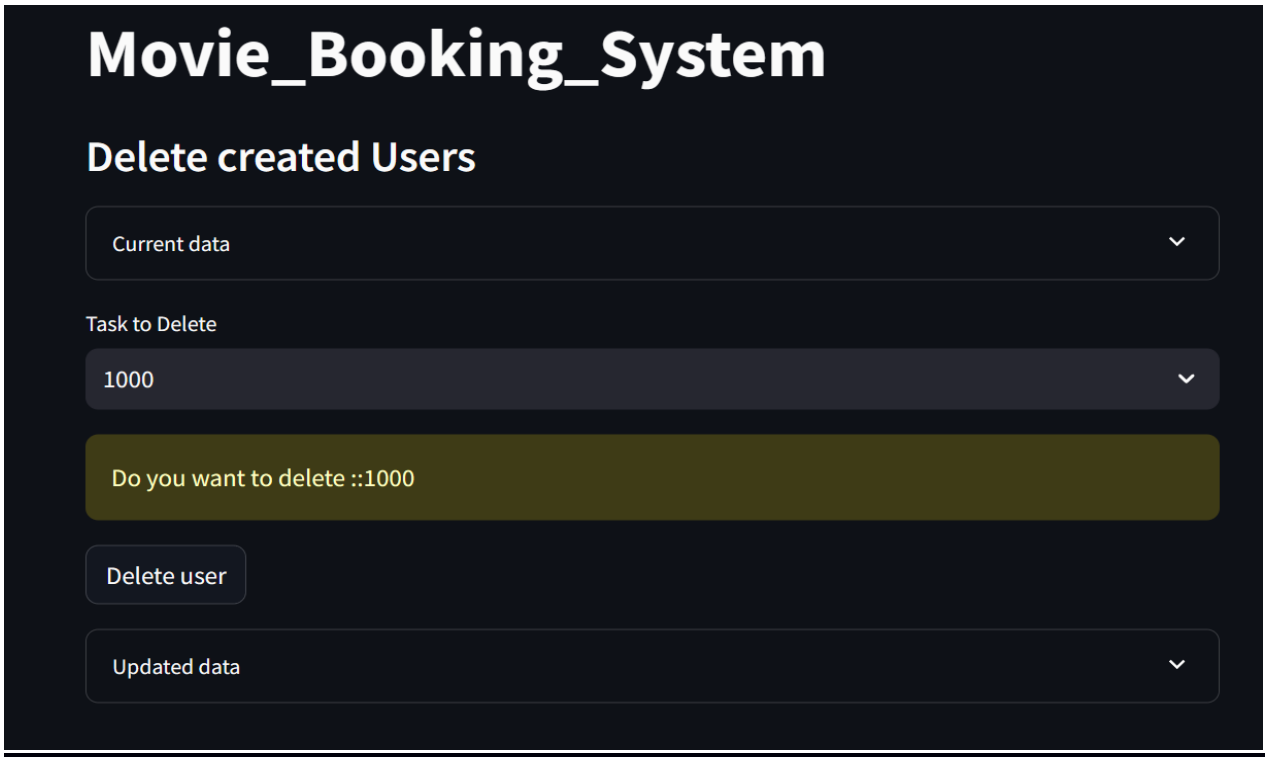
Booking ID: BOOK0004, Name: Sahil SK, Ticket Id: TIDBOOK0004001

Booking ID: BOOK0004, Name: Sahil SK, Ticket Id: TIDBOOK0004002

Booking ID: BOOK0005, Name: Prajwal CA, Ticket Id: TIDBOOK0005001

Booking ID: BOOK0005, Name: Prajwal CA, Ticket Id: TIDBOOK0005002

Delete Users:



11. CONCLUSION:

The Movie Ticket Booking System project offers an efficient and user-friendly solution for managing movie ticket reservations. By providing seamless interfaces and robust functionality, it significantly improves the overall ticket booking experience for both users and cinema administrators. The automation implemented in the system simplifies the reservation process, minimizing errors and enhancing customer satisfaction. In conclusion, this project plays a vital role in the modernization of movie ticket management, contributing to a hassle-free and enjoyable experience for cinema-goers.

12. REFERENCE:

1) https://www.w3schools.com/sql/

2) https://docs.streamlit.io/

Thank you!!!