Movie Ticket Booking System

Project Report

of

21CSC205P Database Management System (DBMS)

Submitted

by

T. YASHWANTH REDDY-(RA2311030010317)

LINGUTLA KOUSHIK - (RA2311030010283)

to Course Faculty

Name	Dr.M.Sivakumar Assistant Professor
Signature	

on

07-03-2025

Department of Networking and Communications
College of Engineering and Technology
SRM Institute of Science and Technology
Kattankulathur, Chengalpattu – 603203

Table of Contents

CHAPTER 1

Introduction

1.1 An Overview of the Project

The Movie Ticket Booking System is an online platform designed to streamline the process of booking movie tickets. It enables users to browse movie listings, select showtimes, choose preferred seats, and make payments, all through a single interface. By automating these tasks, the system eliminates the inefficiencies of traditional manual booking, offering a seamless and hassle-free experience for both customers and theatre administrators.

1.2 Purpose

The primary purpose of the Movie Ticket Booking System is to provide a convenient, efficient, and user-friendly platform for moviegoers and theatre management. It aims to reduce queues at ticket counters, eliminate errors in seat reservations, and enhance the overall movie-watching experience by ensuring real-time updates and confirmations. Additionally, it serves as a centralized database for theatre owners to manage movie schedules, seat availability, and customer transactions.

1.3 Significance

This system is significant in the modern digital age, where automation and convenience are paramount. It not only enhances customer satisfaction by providing a hassle-free booking experience but also improves business operations for theatres. By integrating secure payment methods, real-time seat allocation, and movie schedule management, the system increases operational efficiency and reduces the likelihood of errors, such as overbooking or duplicate reservations.

1.4 The Need for the System

The traditional ticket booking process involves standing in long queues, manual record-keeping, and a higher chance of human errors. This system addresses these issues by providing an automated, digital solution that allows users to book tickets anytime, anywhere. The need for such a system has grown due to the increasing use of online services and the demand for contactless transactions, particularly after the pandemic, where digital solutions became essential for public safety and convenience.

1.5 How It Helps Users or Organizations

For users, the system provides a hassle-free experience by allowing them to browse movies, check availability, and book tickets within minutes. They can also select their preferred seats and make secure online payments, reducing the risk of last-minute unavailability.

For theatre organizations, the system simplifies ticket management, optimizes seat allocation, and enhances revenue tracking. It helps prevent overbooking, reduces the workload on staff, and provides valuable insights into customer preferences and peak booking times. Additionally, the system supports promotional campaigns, loyalty programs, and targeted marketing, further boosting business growth. In summary, the Movie Ticket Booking System is a vital tool that benefits both users and theatre owners by offering a seamless, efficient, and modern solution to movie ticket reservations

CHAPTER 2

Objectives

The Movie Ticket Booking System is designed to improve the efficiency of movie ticket reservations by automating the booking process. The following objectives ensure smooth functioning for both users and theatre management:

1. Efficient Data Storage and Retrieval

- The system uses a structured database to store information about movies, theatres, customers, showtimes, seats, tickets, and payments.
- SQL queries enable fast data access, reducing delays in fetching available seats, movie details, and customer bookings.
- Proper indexing and normalization help avoid data redundancy and ensure data integrity.

2. User Authentication and Role-Based Access

- The system ensures secure access by implementing authentication mechanisms for different user roles (e.g., customers, administrators, and theatre managers).
- Customers can log in to book tickets, view showtimes, and make payments.
- Administrators and theatre managers have role-based access to manage movie schedules, seat availability, and financial records.
- Secure password storage and login methods (e.g., hashed passwords, OTP verification) help prevent unauthorized access.

3. Relationship Management Among Different Entities

- The system establishes clear relationships between entities such as customers, movies, theatres, shows, tickets, and payments.

- Customers are linked to their bookings, and each ticket is associated with a specific show, seat, and payment.
- Theatre management can track reservations, check occupancy, and ensure real-time seat availability updates.

CHAPTER 3

System Requirements

The Movie Ticket Booking System requires both hardware and software components to function efficiently. Below are the necessary requirements to ensure smooth performance and reliability.

3.1 Hardware Requirements

- Processor: Intel Core i3/i5/i7 or an equivalent AMD processor for optimal performance.
- RAM: A minimum of 4GB is required to handle database queries and web operations efficiently.
- Hard Disk: At least 10GB of free space to store application files, user data, and database records.
- Display: A standard resolution monitor (1366x768 or higher) for an optimal user experience.

3.2 Software Requirements

- Operating System: Compatible with Windows, Linux, or MacOS, ensuring cross-platform usability.
- Database Management System (DBMS): Supports MySQL, PostgreSQL, or Oracle for structured data storage and retrieval.
- Programming Language: The system can be developed using Python, Java, or PHP, providing flexibility in backend implementation.
- Web Server: If the system is web-based, an appropriate server like Apache or Tomcat is required to host the application.
- Frontend Technologies: HTML, CSS, JavaScript (for user interface and interaction).
- Frameworks (Optional): Django (Python), Spring Boot (Java), or Laravel (PHP) for enhanced backend development.

CHAPTER 4

System Design

4. Entity-Relationship (ER) Modelling

4.1 ER Modelling

ER modelling helps in designing the database structure by representing entities, attributes, and relationships. It ensures a well-organized database that avoids redundancy and maintains data integrity.

4.2 Steps in ER Modelling

1. Identify Entities – Define core objects like Movies, Theatres, Customers, Shows, Seats, Tickets, and Payments.
2. Define Relationships – Establish links between entities, such as a customer books a Ticket, or A Show belongs to a movie.
3. Specify Attributes – Add details like movie name, ticket price, seat type, and payment amount.
4. Draw the ER Diagram – Visually map out entities and their relationships.
5. Convert to Schema – Translate the diagram into database tables.

4.5 Merits and Demerits of ER Modelling

Merits:

- Clear structure – Helps in organizing complex data with relationships.
- Avoids redundancy – Ensures efficient data storage.
- Better integrity – Defines constraints like primary and foreign keys to maintain consistency.
- Scalability – Can be expanded as new requirements emerge.

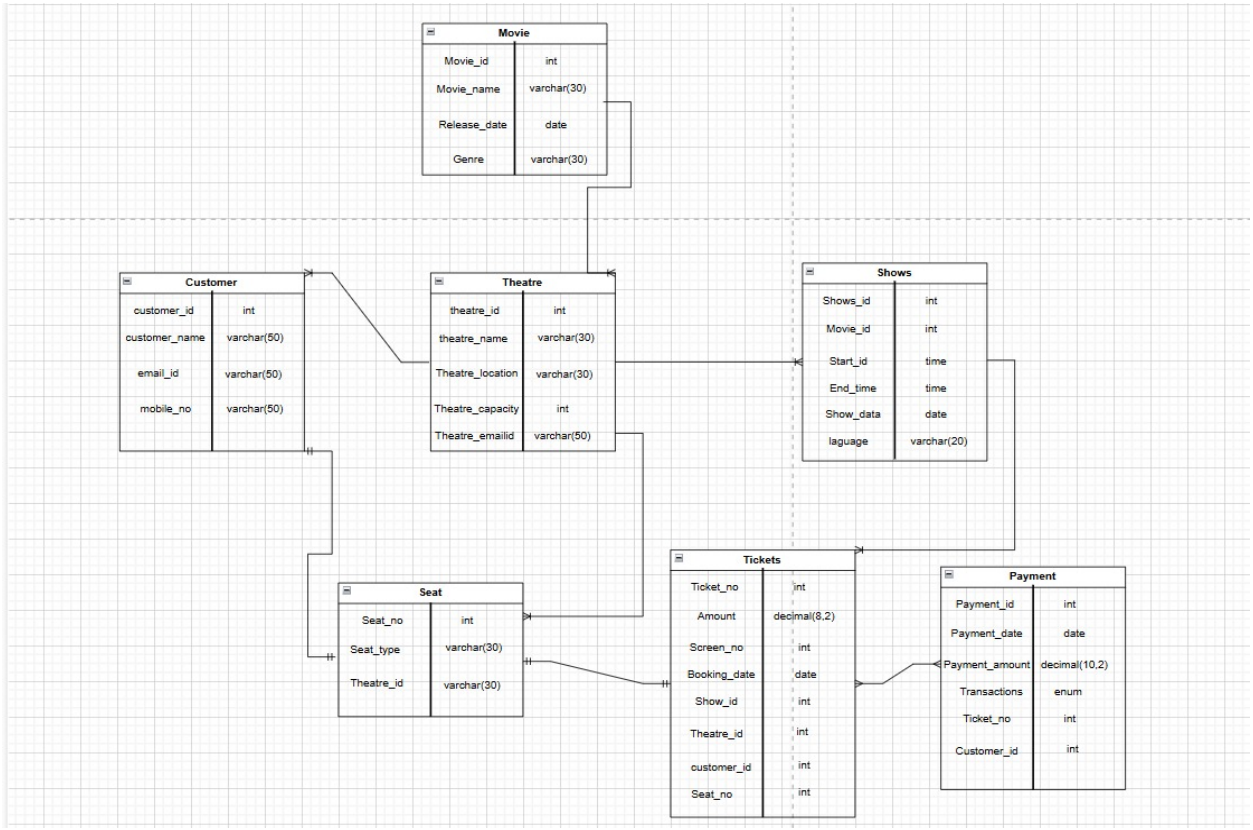
Demerits:

- Complexity – Large databases with multiple entities make ER models harder to manage.
- Performance Issues – Normalized databases may require more joins, slowing down queries.

- Not ideal for unstructured data – Works well for relational databases but not for NoSQL or hierarchical models.

ER Diagram:

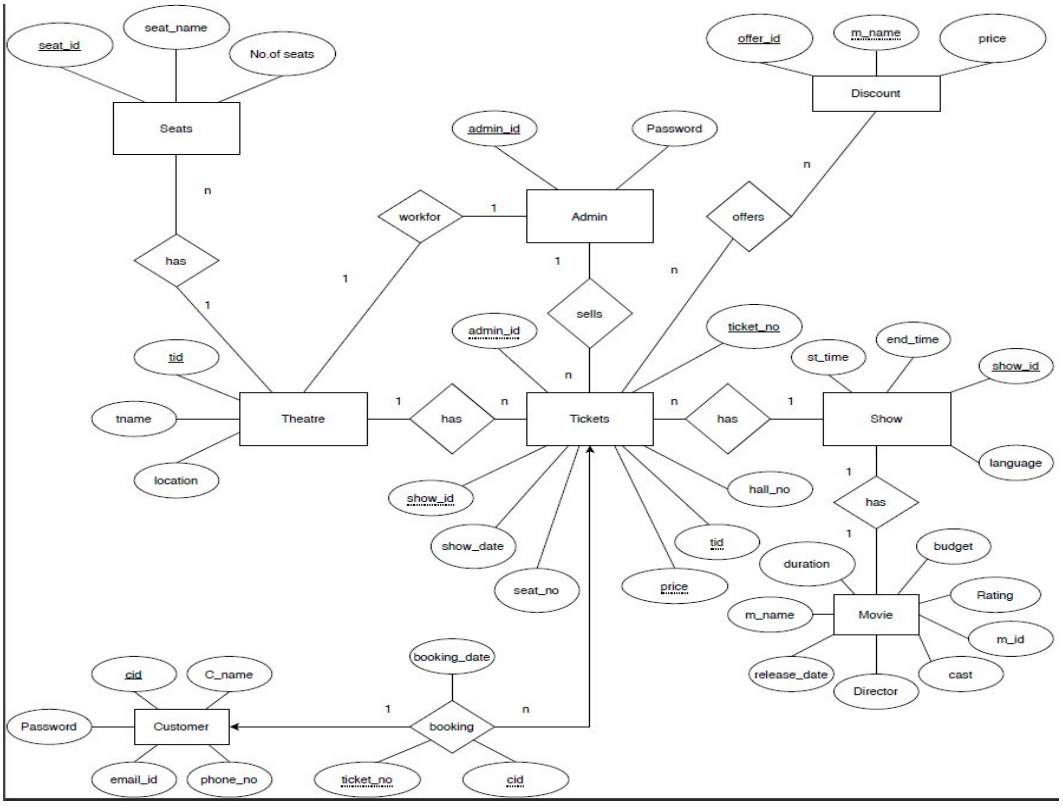
Physical ER Diagram



The physical ER diagram represents the actual database schema with table structures, primary keys (PK), foreign keys (FK), and attributes. It is a refined version of the logical ER model, focusing on database implementation.

- Tables and Relationships:
    - Customer (customer_id PK) → Connected to Tickets (customer_id FK)
    - Theatre (theatre_id PK) → Connected to Seats (theatre_id FK) and Shows (theatre_id FK)
    - Movie (movie_id PK) → Connected to Shows (movie_id FK)
    - Shows (show_id PK) → Connected to Tickets (show_id FK)
    - Tickets (ticket_no PK) → Connected to Payment (ticket_no FK), Seats (seat_no FK), and Customers (customer_id FK)
    - Payment (payment_id PK) → Connected to Tickets (ticket_no FK)

Logical ER Diagram:



The logical ER diagram represents the high-level structure of the Movie Ticket Booking System by defining the entities, their attributes, and relationships without focusing on database implementation

Relationships:

- Customer books Tickets (1: N)
- Theatre has Seats (1: N)
- Theatre hosts Shows (1: N)
- Show is for a Movie (1:1)
- Admin manages the system (1: N)
- Tickets belong to Customers (N:1)
- Discounts apply to Movies (1: N)

CHAPTER 5

IMPLEMENTATION

5.1 Deriving Logical Schema

The Logical Schema represents the structured design of the Movie Ticket Booking System's database. The system includes the following tables:

1. Theatre Table: Stores theatre details.
2. Seat Table: Defines seat types and their relation to theatres.
3. Customer Table: Manages customer information with unique emails.
4. Movie Table: Contains details about available movies.
5. Shows Table: Links movies with screening times and languages.
6. Ticket Table: Stores ticket booking details, customer, and theatre references.
7. Payment Table: Records payment transactions with various payment methods.

These tables are normalized and use primary and foreign keys to ensure integrity.

5.2 CRUD (Create, Read, Update, Delete) Operations

The system supports CRUD operations for managing movie bookings.

- Create (INSERT): Adds new data to the tables.

```
INSERT INTO CUSTOMER (CUSTOMER_ID, CUSTOMER_NAME, EMAIL_ID, MOBILE_NO)
VALUES (1, 'John Doe', 'john@example.com', '9876543210');
```

- Read (SELECT): Retrieves data from tables.

```
SELECT * FROM MOVIE WHERE GENRE = 'Action';
```

- Update (UPDATE): Modifies existing records.

```
UPDATE SEAT SET SEAT_TYPE = 'Deluxe' WHERE SEAT_NO = 10;
```

- Delete (DELETE): Removes records from tables.

```
DELETE FROM TICKET WHERE TICKET_NO = 105;
```

5.3 Authentication and Security Measures

The system ensures data security through:

1. User Authentication: Secure login with hashed passwords using SHA-256.
2. SQL Injection Prevention: Uses prepared statements to prevent malicious queries.
3. Data Encryption: Sensitive data (e.g., payment details) is encrypted.
4. Access Control: Different roles (admin, customer) for restricted access.
5. Session Management: Uses JWT tokens for secure user sessions.

CHAPTER 6

FEATURES AND FUNCTIONALITIES

Core Features of the System

- User Registration and Login: Secure authentication and unique user accounts.
- Movie and Show Management: Adding, updating, and displaying movie schedules.
- Seat Selection & Booking: Users can book specific seats.
- Payments Processing: Supports multiple payment methods (Cash, Card, UPI).
- Data Insertion, Deletion, and Modification: CRUD operations on the database.
- Query Optimization Techniques: Indexing and normalization for faster queries.

CHAPTER 7

RESULTS AND SCREENSHOTS

This chapter includes:

- Database Queries & Outputs: Execution of SQL queries with results.

1. Theatre Table Schema

(The following screenshot displays the schema of the Theatre table, which contains details about different theatres, including their name, location, capacity, and email ID)

```sql
CREATE TABLE Theatre (
    THEATRE_ID INT PRIMARY KEY,
    THEATRE_NAME VARCHAR(30) NOT NULL,
    THEATRE_LOCATION VARCHAR(30) NOT NULL,
    THEATRE_CAPACITY INT,
    THEATRE_EMAIL_ID VARCHAR(50) NOT NULL UNIQUE
);
```

Output:

```
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| THEATRE_ID      | int         | NO   | PRI | NULL    |       |
| THEATRE_NAME    | varchar(30) | NO   |     | NULL    |       |
| THEATRE_LOCATION| varchar(30) | NO   |     | NULL    |       |
| THEATRE_CAPACITY| int         | YES  |     | NULL    |       |
| THEATRE_EMAIL_ID| varchar(50) | NO   | UNI | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.06 sec)
```

2. **Seat Table Schema**

(The following screenshot shows the schema of the Seat table, which stores seat details such as seat number, type, and the theatre it belongs to.)

```
CREATE TABLE Seat (
    SEAT_NO INT PRIMARY KEY,
    SEAT_TYPE ENUM('Regular', 'Recliner', 'Deluxe', 'VIP') NOT NULL,
    THEATRE_ID INT,
    FOREIGN KEY (THEATRE_ID) REFERENCES Theatre(THEATRE_ID)
);
```

Output:

```
+------------+-----------------------------------------+------+-----+---------+-------+
| Field      | Type                                    | Null | Key | Default | Extra |
+------------+-----------------------------------------+------+-----+---------+-------+
| SEAT_NO    | int                                     | NO   | PRI | NULL    |       |
| SEAT_TYPE  | enum('Regular','Recliner','Deluxe','VIP') | NO   |     | NULL    |       |
| THEATRE_ID | int                                     | YES  | MUL | NULL    |       |
+------------+-----------------------------------------+------+-----+---------+-------+
```

3.Customer Table Schema

(The following screenshot displays the schema of the Customer table, which contains customer information, including name, email ID, and mobile number.)

```sql
CREATE TABLE Customer (
    CUSTOMER_ID INT PRIMARY KEY,
    CUSTOMER_NAME VARCHAR(50) NOT NULL,
    EMAIL_ID VARCHAR(50) NOT NULL UNIQUE,
    MOBILE_NO VARCHAR(10) NOT NULL
);
```

Output:

```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| CUSTOMER_ID   | int         | NO   | PRI | NULL    |       |
| customer__name | varchar(50) | NO   |     | NULL    |       |
| EMAIL_ID      | varchar(50) | NO   | UNI | NULL    |       |
| MOBILE_NO     | varchar(10) | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

Movie Table Schema

(The following screenshot shows the schema of the Movie table, which stores movie details such as movie ID, name, release date, and genre.)

```sql
CREATE TABLE Movie (
    MOVIE_ID INT PRIMARY KEY,
    MOVIE_NAME VARCHAR(50) NOT NULL,
    RELEASE_DATE DATE NOT NULL,
    GENRE VARCHAR(30) NOT NULL
);
```

Output:

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| MOVIE_ID     | int         | NO   | PRI | NULL    |       |
| MOVIE_NAME   | varchar(50) | NO   |     | NULL    |       |
| RELEASE_DATE | date        | NO   |     | NULL    |       |
| GENRE        | varchar(30) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Shows Table Schema

(The following screenshot displays the schema of the Shows table, which contains details about movie shows, including show timings, date, language, and associated movie.)

```sql
CREATE TABLE Shows (
    SHOW_ID INT PRIMARY KEY,
    MOVIE_ID INT,
    START_TIME TIME NOT NULL,
    END_TIME TIME NOT NULL,
    SHOW_DATE DATE NOT NULL,
    LANGUAGE VARCHAR(20) NOT NULL,
    FOREIGN KEY (MOVIE_ID) REFERENCES Movie(MOVIE_ID)
);
```

Output:

```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| SHOW_ID    | int         | NO   | PRI | NULL    |       |
| MOVIE_ID   | int         | YES  | MUL | NULL    |       |
| START_TIME | time        | NO   |     | NULL    |       |
| END_TIME   | time        | NO   |     | NULL    |       |
| SHOW_DATE  | date        | NO   |     | NULL    |       |
| LANGUAGE   | varchar(20) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

6. Ticket Table Schema

(The following screenshot shows the schema of the Ticket table, which contains booking details such as ticket number, amount, screen number, booking date, show ID, and theatre ID.)

```sql
CREATE TABLE Ticket (
    TICKET_NO INT PRIMARY KEY,
    AMOUNT DECIMAL(10,2) NOT NULL,
    SCREEN_NO INT NOT NULL,
    BOOKING_DATE DATE NOT NULL,
    SHOW_ID INT,
    THEATRE_ID INT,
    FOREIGN KEY (SHOW_ID) REFERENCES Shows(SHOW_ID),
    FOREIGN KEY (THEATRE_ID) REFERENCES Theatre(THEATRE_ID)
);
```

Output:

```
mysql> desc ticket;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| TICKET_NO    | int           | NO   | PRI | NULL    |       |
| AMOUNT       | decimal(10,2) | NO   |     | NULL    |       |
| SCREEN_NO    | int           | NO   |     | NULL    |       |
| BOOKING_DATE | date          | NO   |     | NULL    |       |
| SHOW_ID      | int           | YES  | MUL | NULL    |       |
| THEATRE_ID   | int           | YES  | MUL | NULL    |       |
| CUSTOMER_ID  | int           | YES  | MUL | NULL    |       |
| SEAT_NO      | int           | YES  | MUL | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

Conclusion

The Movie Ticket Booking System provides a user-friendly interface for booking movie tickets, ensuring data security, fast processing, and a seamless user experience. It successfully integrates database management with authentication and optimized queries.

Future Enhancements

1. **Integration with Cloud Storage:** Storing user data and booking history securely on the cloud.
2. **AI-Based Recommendations:** Personalized movie recommendations using AI.
3. **Performance Improvement:** Using indexing and caching for faster data retrieval

REFERENCES

- Database Management System by Korth & Silberschatz
- SQL Optimization Techniques – Oracle Documentation
- MySQL Documentation – Foreign Key Constraints
- Web Security Best Practices – OWASP Guide
- Online resources for CRUD operations and authentication methods