**Database Management System**

**Semester-III (Batch-2024)**

**Movie Ticket Booking System**

**Supervised By:**

**Sachin Garg**

**Submitted By:**

**Aarti Goyal, 2410990809 (G11)**

**Aditi Jindal, 2410990818 (G11)**

**Aman Thakur, 2410990827 (G11)**

****

**Department of Computer Science and Engineering**

**Chitkara University Institute of Engineering & Technology,**

**Chitkara University, Punjab**

**Movie Ticket Booking System**

**1. Introduction**

The movie industry is one of the most popular sources of entertainment, with millions of people booking tickets every day to watch their favourite films. Traditionally, movie tickets were purchased manually at cinema counters, which is time-consuming, inconvenient, and often leads to errors in booking and seat allocation.

In today’s digital era, a **Movie Ticket Booking System** is essential to simplify and automate the ticketing process. The proposed system serves as a centralized platform for managing movies, show timings, seat availability, bookings, and payments efficiently.

This system is designed to meet the operational needs of both customers and cinema management by ensuring quick reservations, accurate seat allocation, and secure payment handling. It incorporates modern database principles to maintain data integrity, consistency, security, and scalability.

**2. Problem Overview**

Movie ticket booking involves multiple interconnected entities such as **Customer, Movies, Shows, Theaters, Seats, Payments**, and **Bookings.** In the absence of a well-designed management system, the following challenges arise:

* Manual ticket booking leads to long queues and delays for customers.
* Errors in seat allocation and overbooking due to poor record-keeping.
* Difficulty in tracking available seats, show timings, and movie schedules.
* Lack of centralized access to booking information across different theaters
* Security issues related to payment handling and customer details.
* Limited ability to generate reports on sales, occupancy rates, and customer preferences.

A well-structured Movie Ticket Booking System eliminates these problems by providing a centralized, relational database with clearly defined entities, relationships, and constraints.

**3. Scope of the Project**

The scope of this project is to design a relational database that will:

* Store all details of customers, movies, theaters, shows, seats, bookings, and payments.
* Represent the complex relationships between these entities in a logical and optimized structure.
* Enable fast retrieval and updating of records through structured queries.
* Support future scalability by accommodating growth in the number of movies, theaters, and customers.
* Ensure data security and access control to protect sensitive customer and payment information.

1. **Objectives**
2. **Centralized Data Management**

Develop a relational database model that integrates all movie booking data—customers, movies, theaters, show timings, seats, and payments—into a unified system, eliminating duplication and redundancy.

1. **Efficient Data Retrieval**

Optimize the database design to allow fast and accurate retrieval of information such as seat availability, movie schedules, booking history, and customer details.

1. **Relationship Mapping**

Implement proper handling of one-to-many and many-to-many relationships, such as:

* A theatre hosting multiple movies.
* A movie having multiple show timings.
* A customer booking multiple tickets for different shows.

1. **Data Integrity and Consistency**

Enforce Primary Keys, Foreign Keys, and constraints to maintain accuracy and validity of booking records, seat allocations, and payments.

1. **Security and Access Control**

Protect sensitive customer and payment information using secure database practices and role-based access for administrators, theatre staff, and customers.

1. **Scalability and Maintainability**

Ensure the system can handle future growth in the number of theaters, movies, and users, and can be updated easily without disrupting ongoing bookings.

1. **Support for Reporting and Analytic**

Provide the ability to generate reports for:

* Movie-wise and theatre-wise ticket sales.
* Customer booking history and preferences.
* Show occupancy rates.
* Revenue tracking and payment summaries.

1. **Minimizing Redundancy**

Apply normalization techniques to reduce data duplication, optimize storage, and maintain consistency across bookings, customer records, and payment details.

**5. Significance of the Project**

The proposed Movie Ticket Booking System will significantly improve the movie ticketing process by:

* Reducing long queues and manual booking delays.
* Improving accuracy in seat allocation and booking records.
* Enhancing customer satisfaction through faster and convenient online booking.
* Strengthening security of customer and payment information.
* Enabling theatre management to make informed decisions through accurate and timely reports on sales and occupancy.

**6. Main Entities and Attributes**

**6.1 Customer**

* customer\_id (PK)
* name
* email (unique)
* phone
* password

**6.2 Movie**

* Movie \_ID (PK)
* Title
* Genre
* Language
* Duration
* Release \_Date
* Rating

**6.3 Theatre**

* Theatre \_ID (PK)
* Name
* Location
* Total \_screens

**6.4 Show**

* Show \_ID (PK)
* Movie \_ID (FK)
* Theatre \_ID (FK)
* Show \_Date
* Show \_Time

**6.5 Seat**

* Seat \_ID (PK)
* Theatre \_ID (FK)
* Seat \_Number
* Seat \_Type (Regular/VIP)
* Availability \_Status

**6.6 Booking**

* Booking \_ID (PK)
* Customer \_ID (FK)
* Show \_ID (FK)
* Booking \_Date
* Number\_ of\_ Seats
* Total \_Amount

**6.7 Payment**

* Payment \_ID (PK)
* Booking \_ID (FK)
* Payment \_Date
* Payment \_Method (Card/UPI/Net-Banking)
* Amount
* Payment \_Status

1. **Relationships**

**Customer – Booking**

One customer can make multiple bookings.

Each booking belongs to exactly one customer.

Cardinality: 1 : M (Customer → Booking)

Name: Makes

**Movie – Show**

One movie can have multiple shows.

Each show is linked to exactly one movie.

Cardinality: 1 : M (Movie → Show)

Name: Has

**Theatre – Show**

One theatre can host multiple shows.

Each show runs in exactly one theatre.

Cardinality: 1 : M (Theatre → Show)

Name: Hosts

**Show – Seat**

One show contains many seats.

Each seat belongs to exactly one show.

Cardinality: 1 : M (Show → Seat)

Name: Contains

**Show – Booking**

One show can be booked many times by different customers.

Each booking belongs to exactly one show.

Cardinality: 1 : M (Show → Booking)

Name: Is\_Booked\_In

**Seat – Booking**

One seat (for a given show) can be reserved by only one booking.

Each booking reserves exactly one seat.

Cardinality: 1 : 1 (Seat ↔ Booking)

Name: Reserves

**Booking – Payment**

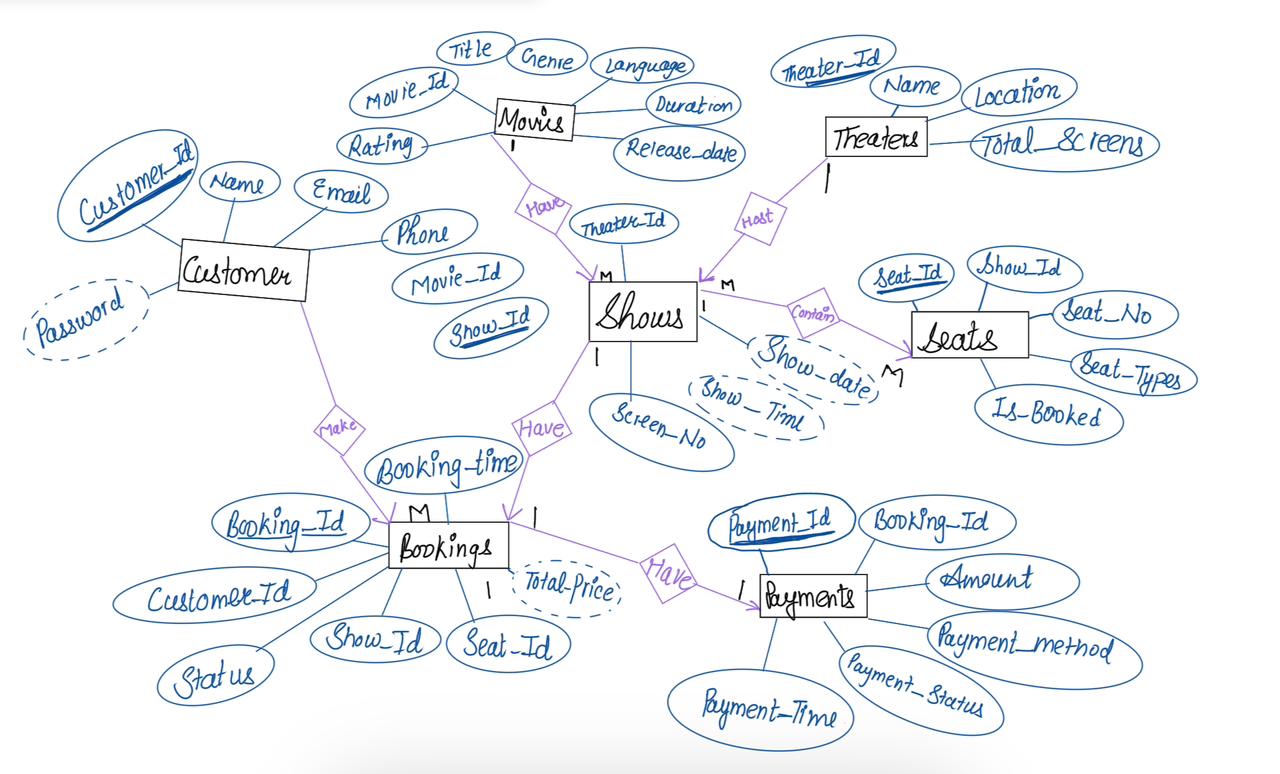
One booking generates exactly one payment.

Each payment is linked to exactly one booking.

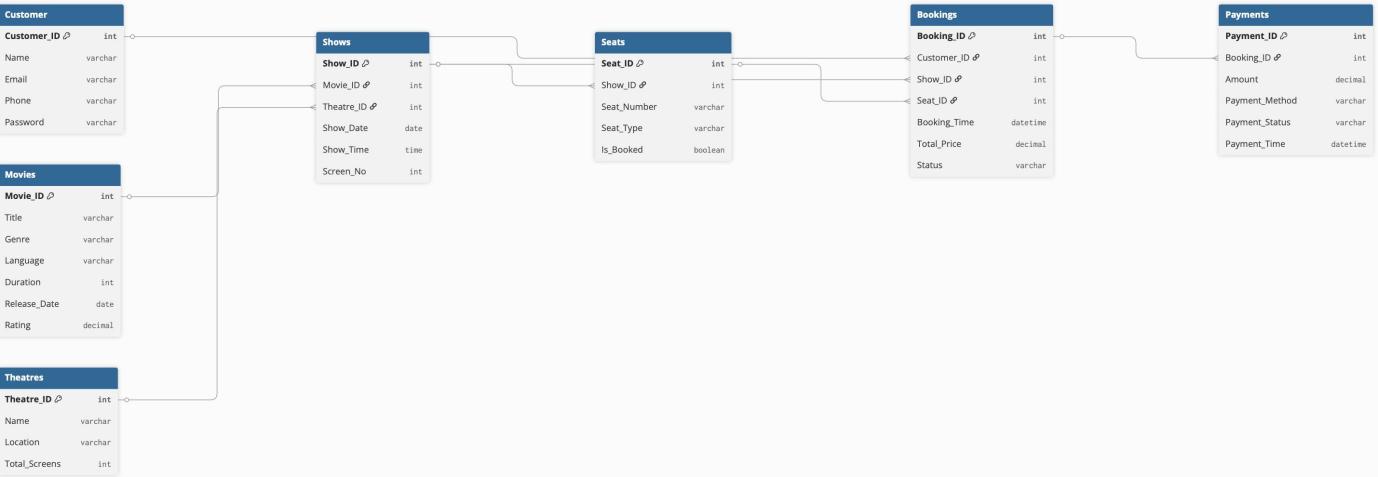
Cardinality: 1 : 1 (Booking ↔ Payment)

Name: Has

**ER DIAGRAM**



**ER MODEL**

****

**8. ER Model to Relational Schema**

Customer (customer\_id [PK], name, email, phone, password)

Movies (movie\_id [PK], title, genre, language, duration, release\_date, rating)

Theaters (theatre\_id [PK], name, location, total\_screens)

Shows (show\_id [PK], movie\_id [FK], theatre\_id [FK], show\_date, show\_time, screen\_no)

Seats (seat\_id [PK], show\_id [FK], seat\_number, seat\_type, is\_booked)

Bookings (booking\_id [PK], customer\_id [FK], show\_id [FK], seat\_id [FK], booking\_time, total\_price, status)

Payments (payment\_id [PK], booking\_id [FK], amount, payment\_method, payment\_status, payment\_time)

**9. Relational Algebra (RA) Queries — 25**

**Q1. Get names of customers who made bookings.**  
→ π\_name (Customer ⨝ Booking)

**Q2. Find all bookings made by Customer\_ID = 101.**  
→ σ\_CustomerID=101 (Booking)

**Q3. List all shows of movie “Inception”.**  
→ π\_Show\_ID, Show\_Date, Show\_Time (σ\_Title="Inception" (Movie ⨝ Show))

**Q4. Get details of movies released after 2020.**  
→ σ\_ReleaseDate > "2020-01-01" (Movie)

**Q5. Find all movies with rating greater than 8.**  
→ σ\_Rating>8(Movie)

**Q6. Get names of customers who have not made any booking.**  
→ π\_Name (Customer) − π\_Name (Customer ⨝ Booking)

**Q7. Find all theaters located in “Mumbai”.**  
→ σ\_Location="Mumbai" (Theatre)

**Q8. List all VIP seats available in Theatre\_ID = 5.**  
→ σ\_Seat\_Type="VIP" ∧ Availability\_Status="Available" ∧ Theater\_ID=5 (Seat)

**Q9. Get titles of movies currently showing in Theatre\_ID = 3.**  
→ π\_Title (Movie ⨝ (σ\_Theater\_ID=3 (Show)))

**Q10. Find customers who booked more than 3 tickets in a single booking.**  
→ σ\_Number\_of\_Seats > 3 (Booking ⨝ Customer)

**Q11. Get all movies that have never been booked.**  
→ π\_Movie\_ID (Movie) − π\_Movie\_ID (Show ⨝ Booking)

**Q12. List customers who booked shows on “2025-01-15”.**  
→ π\_Name (σ\_Booking\_Date="2025-01-15" (Customer ⨝ Booking))

**Q13.Find all movies of genre “Comedy”.**  
→ σ\_Genre="Comedy"(Movie)

**Q14. List movies of genre “Action” that are booked.**  
→ π\_Title (σ\_Genre="Action" (Movie ⨝ Show ⨝ Booking))

**Q15. Get movie titles and corresponding customer names for all bookings.**  
→ π\_Title, Name (Movie ⨝ Show ⨝ Booking ⨝ Customer)

**Q16. List all shows running in Theatre\_ID = 2.**→ σ\_Theater\_ID=2​(Show)

**Q17. List customers who booked seats but cancelled payment (Payment\_Status = "Failed").**  
→ π\_Name (σ\_Payment\_Status="Failed" (Customer ⨝ Booking ⨝ Payment))

**Q18. Get names of customers who booked but not paid yet.**  
→ π\_Name (Customer ⨝ Booking) − π\_Name (Customer ⨝ Booking ⨝ Payment)

**Q19.** . **Find all bookings made on ‘2025-01-10’.**  
→ σ\_Booking\_Date="2025−01−10"(Booking)

**Q20. Get movies booked between March and May 2025.**  
→ π\_Title (σ\_Booking\_Date ≥ "2025-03-01" ∧ Booking\_Date ≤ "2025-05-31" (Booking ⨝ Show ⨝ Movie))

**Q21. List all payments made via “UPI”.**  
→ σ\_PaymentMethod="UPI" (Payment)

**Q22. Find theaters where Customer\_ID = 205 made bookings.**  
→ π\_Name (σ\_Customer\_ID=205 (Booking ⨝ Show ⨝ Theatre))

**Q23. Find all customers who booked tickets for Movie\_ID = 10.**  
→ π\_Name(σMovie\_ID=10(Customer⨝Booking⨝Show))

**Q24. List all bookings made on “2025-02-10”.**  
→σ\_Booking\_Date="2025−02−10"(Booking)

**Q25. List all payments that failed (Payment\_Status = "Failed").**

→ σ\_Payment\_Status="Failed" (Payment)