# SE Experiment-3
## (Division:D - Batch:D)

**-Student's Detail-**
Dev Rai – UID (2024301022)
T.E. Computer Engineering

**Aim:**
To draw a Class Diagram for a Theatre Management System.

**Implementation:**

The design of the Class Diagram for the Theatre Management System was created by analyzing the domain of theatre operations and identifying the essential concepts involved. Key entities such as Theatre, Movie, Seat, Ticket, Customer, and Payment were recognized through an understanding of real-world processes in theatre management. These entities were then refined into classes with well-defined responsibilities. The actions and interactions typically performed in a theatre context helped determine the methods (operations) for each class. Finally, the relationships among these classes, including associations, multiplicities, composition, and generalization, were carefully structured to represent the overall functionality and organization of the system.

**Problem Description:**

The Theatre Management System is a web-based platform designed to automate and streamline operations for a movie theatre. The system must manage two primary user roles: Customers and Administrators. Customers need to be able to register, log in, browse movies, view showtimes, select seats from a graphical map, book tickets, and process payments securely. After a successful booking, they should receive a downloadable PDF ticket. Administrators require a secure dashboard to perform CRUD operations on movies and manage screening schedules in various auditoriums. They also need to be able to view system reports on metrics like occupancy and revenue to track business performance.

Following is a breakdown of various parts that were considered for designing the class diagram-

[1] Noun/Noun Phrases.

User, Customer, Administrator, Movie, Screening, Showtime, Auditorium (Screen), Seat, Ticket, Booking, Payment, Report, Profile, Movie Details (Title, Synopsis, Duration, Genre, Poster), Interactive Map, Booking ID, QR Code, Payment.

[2] Classes.

User, Customer, Administrator, Movie, Screening, Auditorium, Seat, Booking, Payment, Ticket

[3] Verb Phrases.

1. Customer registers, logs in, and manages their profile.
2. Customer browses and views movies.
3. Customer selects a screening, chooses seats, and books a ticket.
4. Customer processes payment.
5. Customer views booking history and prints an e-ticket.
6. Administrator manages (adds, updates, deletes) movies.
7. Administrator manages (schedules, cancels) screenings.
8. Administrator views and exports reports.
9. System generates a PDF ticket.
10. System validates credentials and seat availability.

[4] Relations.

1. (User Hierarchy)
   - Customer and Administrator specialize the User class.
   - The User class is specialized by both Customer and Administrator.

2. (Auditorium and Seat)
   - An Auditorium is composed of one or more Seats.

3. (Customer and Booking)
   - A Customer makes zero or more Bookings.
   - A Booking is made by exactly one Customer.

4. (Administrator and Movie)
   - An Administrator manages zero or more Movies.
   - A Movie is managed by exactly one Administrator.

5. (Administrator and Screening)
   - An Administrator manages zero or more Screenings.
   - A Screening is managed by exactly one Administrator.

6. (Screening and Movie)
   - A Screening shows exactly one Movie.
   - A Movie is shown by zero or more Screenings.

7. (Screening and Auditorium)
   - An Auditorium hosts zero or more Screenings.
   - A Screening is hosted by exactly one Auditorium.

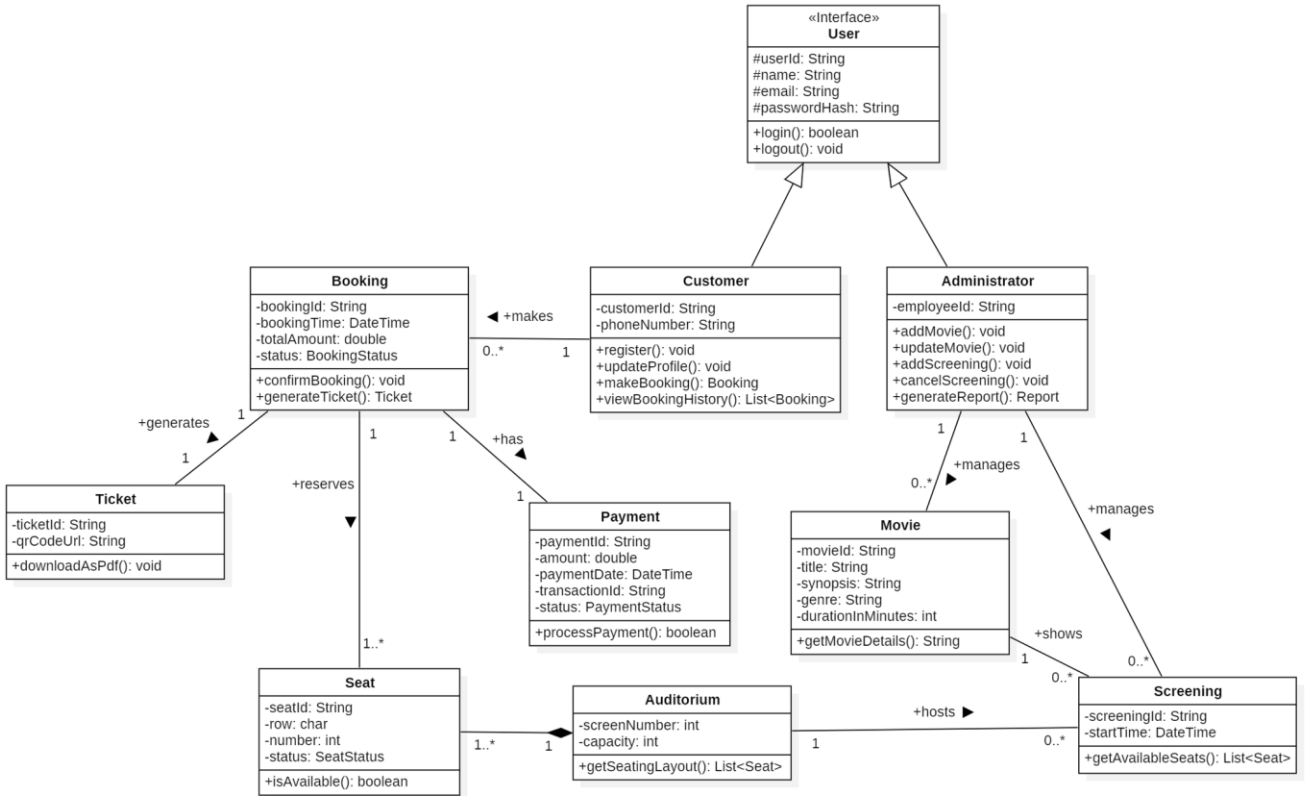
8. (Booking and Screening)
   - A Booking is made for exactly one Screening.
   - A Screening is booked by zero or more Bookings.

9. (Booking and Seat)
   - A Booking reserves one or more Seats.
   - A Seat is reserved by exactly one Booking.

10. (Booking and Payment)
    - A Booking requires exactly one Payment.
    - A Payment is required by exactly one Booking.

11. (Booking and Ticket)
    - A Booking generates exactly one Ticket.
    - A Ticket is generated by exactly one Booking.

[5] Class Diagram.




Following are the code snippets that were written corresponding to each class and interface demonstrated in the above diagram-

User:

```
public interface User {
    String userId;
    String name;
    String email;
    String passwordHash;

    public boolean login(String email, String password);
    public void logout();
}
```

CUSTOMER:

```
public class Customer implements User {
    private String phoneNumber;

    public void register(){}
    public void updateProfile(){}
    public Booking makeBooking(Screening screening, List<Seat> seats){}
    public List<Booking> viewBookingHistory(){}


    public boolean login(String email, String password){}
```

```
    public void logout(){}
}
```

ADMINISTRATOR:

```
public class Administrator implements User {
    private String employeeId;

    public void addMovie(Movie movie){}
    public void updateMovie(Movie movie){}
    public void addScreening(Screening screening){}
    public void cancelScreening(String screeningId){}
    public Report generateReport(String reportType){}

    public boolean login(String email, String password){}

    public void logout(){}
}
```

MOVIE:

```
public class Movie {
    private String movieId;
    private String title;
    private String synopsis;
    private int durationInMinutes;
    private String genre;

    public String getMovieDetails(){}
}
```

SCREENING:

```
public class Screening {
    private String screeningId;
    private DateTime startTime;

    public List<Seat> getAvailableSeats(){}
}
```

AUDITORIUM:

```
public class Auditorium {
    private int screenNumber;
    private int capacity;

    public List<Seat> getSeatingLayout(){}
}
```

SEAT:

```
public class Seat {
    private String seatId;
    private char row;
    private int number;
    private SeatStatus status;

    public boolean isAvailable(){}
}
```

BOOKING:

```
public class Booking {
    private String bookingId;
    private DateTime bookingTime;
    private double totalAmount;
    private BookingStatus status;

    public void confirmBooking(){}
    public Ticket generateTicket(){}
}
```

PAYMENT:

```
public class Payment {
    private String paymentId;
    private double amount;
    private DateTime paymentDate;
    private String transactionId;
    private PaymentStatus status;

    public boolean processPayment(){}

}
```

TICKET:

```
public class Ticket {
    private String ticketId;
    private String qrCodeUrl;

    public void downloadAsPdf(){}
}
```

**Conclusion:**

Through this experiment, I understood how to design a class diagram for the Theatre Management System by analyzing its requirements. I identified the main classes like Customer, Booking, and Movie, defined their attributes and operations, and mapped their relationships through generalization, composition, and associations. This gave me a clear blueprint of the system's structure and a solid guide for its development.