

MOVIE TICKET RESERVATION SYSTEM
CS23333 – Object Oriented Programming using Java

Submitted by

ANUSHAA MAAI T M - 231001015

HEMAPRABHA S - 231001063

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE
BONAFIDE CERTIFICATE

Certified that this project titled **“MOVIE TICKET RESERVATION SYSTEM”** is the bonafide work of **“ANUSHAA MAAI T M (231001015), HEMAPRABHA S (2310010063)”** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.P.Valarmathie

HEAD OF THE DEPARTMENT

Department of Information Technology
Rajalakshmi Engineering College

SIGNATURE

Mrs.Usha S

COURSE INCHARGE

Department of Information Technology
Rajalakshmi Engineering College

Submitted to Project Viva – Voce Examination held on

INTERNAL EXAMINAR

EXTERNAL EXAMINAR

ABSTRACT

The Movie Ticket Booking System is an online platform designed to simplify the process of purchasing movie tickets. The system allows users to browse and select movies, view showtimes, and choose their preferred cinema locations. The Home Page serves as the introductory point where users can explore a variety of movies currently playing, view movie details such as genres and ratings, and easily navigate to the booking page for further actions. This page offers an intuitive design, making it easy for users to quickly find the movie they are interested in and proceed with the booking process. The Booking Page enables users to select their desired showtime and choose from available seats through an interactive seating chart. Here, users can review the movie, showtime, and seating arrangements before confirming their ticket choices. This page ensures that customers can make informed decisions, offering them a clear overview of available seats and the option to adjust the number of tickets or seating preferences. Once the user finalizes the ticket selection, they are guided to the Payment Page, where they can securely enter payment details and complete the transaction. The overall design of the system prioritizes ease of use, security, and customer satisfaction, providing a seamless and efficient movie ticket booking experience from start to finish.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	iii
	LIST OF TABLES	iv
1	INTRODUCTION	1
	1.1 PROBLEM STATEMENT	1
	1.2 OBJECTIVE OF THE PROJECT	2
	1.3 ORGANIZATION OF THE REPORT	2
2	SYSTEM DESIGN	3
	2.1 OVERVIEW	3
	2.2 LIST OF MODULES	3
	2.3 UML MODELING	4
	2.3.1 USE CASE DIAGRAM OVERVIEW	5
	2.3.2 ENTITY-RELATIONSHIP DIAGRAM	6
	2.3.3 DATA FLOW DIAGRAM	6
	2.4 SYSTEM SPECIFICATION	7
	2.4.1 FUNCTIONAL REQUIREMENTS	9
3	DESIGN	10
	3.1 DESIGN	10
	3.2 DATABASE DESIGN	14
	3.1 IMPLEMENTATION (CODE)	15
4	CONCLUSION	21
	REFERENCES	22

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NUMBER
2.1	USE CASE DIAGRAM OF MOVIE TICKET RESERVATION DIAGRAM	5
2.2	ENTITY RELATIONSHIP DIAGRAM OF MOVIE RESERVATION SYSTEM	6
2.3	DATA FLOW DIAGRAM	6
3.1	HOME PAGE FOR AH CINEMAS	10
3.2	SELECTING MOVIE PAGE	10
3.3	LOGIN PAGE	11
3.4	PAYMENT PAGE	11
3.5	RESERVATION SUCCESSFUL PAGE	12
3.6	MOVIE ADDING PAGE	12
3.7	MOVIE LIST PAGE	13
3.8	BOOKINGS LIST PAGE	13

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NUMBER
3.2.1	BOOKINGS TABLE	14
3.2.2	MOVIES TABLE	14

CHAPTER 1

INTRODUCTION

1.1.PROBLEM STATEMENT

In today's fast-paced digital world, movie theaters continue to rely on outdated, manual systems for ticket reservations, leading to several operational challenges and a subpar customer experience. Moviegoers often face difficulties when booking tickets, including lack of real-time seat availability, limited booking options, and inefficient manual processes. Additionally, existing systems may not provide the flexibility required for modern consumer demands, such as dynamic pricing, personalized discounts, or diverse payment options. This not only results in customer dissatisfaction but also operational inefficiencies for theaters.

Key issues include:

1. **Inefficient Manual Booking:** Moviegoers have to rely on traditional methods such as phone calls or in-person visits to reserve tickets, resulting in longer wait times and increased administrative overhead for theaters.
2. **Limited Payment Methods:** Current reservation systems typically offer only basic payment options, restricting users from using modern, convenient payment methods like digital wallets, mobile payments, or credit card integrations.
3. **Lack of Personalization and Pricing Flexibility:** Movie theaters struggle to offer dynamic pricing models or personalized promotions based on customer preferences or loyalty, missing opportunities for upselling and enhancing the customer experience.

Objective: The aim of this movie reservation system project is to develop an automated, scalable platform that streamlines the entire booking process. This platform will enable users to browse showtimes, book seats in real time, choose from various payment methods, access personalized offers, and receive efficient customer support. The goal is to enhance user convenience, increase operational efficiency for theatres, and provide a seamless, enjoyable movie-going experience for customers.

1.2.OBJECTIVE OF THIS PROJECT

The primary objective of the Movie Ticket Booking System is to provide a seamless, user-friendly platform that enables users to efficiently browse movies, select showtimes, and book tickets online. The system aims to offer a well-designed Home Page where users can easily explore available movies, view trailers, and check showtimes. The Booking Page should allow for easy seat selection, with real-time availability updates, ensuring a smooth and efficient ticketing process. The Payment Page must support secure, multiple payment options, enabling users to complete transactions with confidence. Finally, the View Booking Page will offer users a clear overview of their booking details, with options to modify or cancel tickets as needed. The overall goal is to enhance user satisfaction by streamlining the entire booking process, offering flexibility, security, and convenience at each step.

1.3.ORGANIZATION OF THE REPORT

CHAPTER 1 – INTRODUCTION

CHAPTER 2 – SYSTEM DESIGN

CHAPTER 3 – IMPLEMENTATION

CHAPTER 4 – CONCLUSION

CHAPTER 2

SYSTEM DESIGN

2.1. OVERVIEW

The Movie Ticket Booking System is designed with a modular architecture that ensures scalability, usability, and efficient performance. The system is divided into four main components: the Home Page, Booking Page, Payment Page, and View Booking Page. The Home Page acts as the entry point, providing users with an intuitive interface to browse available movies, check showtimes, and access detailed movie information. It connects to a backend database that stores movie data, including showtimes, cinema locations, and movie details, using SQL or NoSQL technologies. The Booking Page allows users to select showtimes, choose seats using an interactive seating chart, and review booking details. The seating chart is dynamically updated in real-time to reflect available seats, which enhances user experience by providing accurate information during the selection process.

The Payment Page integrates a secure payment gateway, allowing users to make transactions using multiple payment methods such as credit/debit cards or digital wallets. This page uses encryption protocols (e.g., SSL/TLS) to protect users' financial information. Once payment is processed, the system updates the backend database and confirms the booking. The View Booking Page displays a summary of the user's booking details, including movie name, showtime, and seat number, and provides options to modify or cancel the booking if necessary. The entire system is designed to ensure high availability and responsiveness, utilizing cloud services and scalable backend infrastructure to handle varying traffic loads, especially during peak times. This architecture supports a smooth, secure, and efficient user experience across all stages of the booking process.

2.2. LIST OF MODULES

1. User Management Module
2. Product Management Module
3. Transaction Management Module
4. Cross-Origin Resource Sharing(CORS) Handling Module
5. Error Handling and Response Module
6. Database Connectivity Module

2.3. UML MODELING

2.3.1. USE CASE DIAGRAM OVERVIEW:

The use case diagram for the Movie Reservation System visualizes the interactions between users (customers and administrators) and the system, outlining the functional requirements and key activities associated with booking ,managing and viewing movie tickets. The primary actors in this diagram are:

Actors:

1. User

- Can book tickets.
- Can view available movies.

2. Admin

- Can add new movies.
- Can view available movies.
- Can view all bookings.

Use Cases:

1. Book Ticket (User)

- **Select Movie:** Choose a movie from the list of available movies.
- **Enter User Details:** Provide name, phone number, and other required information.
- **Payment Successful:** Display a confirmation message once the payment is successful.

2. View Movies (User and Admin)

- Display the list of movies available in the system.

3. Add Movie (Admin)

- Admin can add a new movie with details such as title, available seats, and price per ticket.

4. View Bookings (Admin)

- Display a list of all bookings made by users.

Relationships:

1. Includes

- Book Ticket includes sub-processes like Select Movie, Enter User Details, Make Payment, and Payment Successful.

2. Actors to Use Cases

- User interacts with Book Ticket and View Movies.
- Admin interacts with Add Movie, View Bookings, and View Movies.

Overview:

The Movie Reservation System captures the key elements of the Movie Reservation System's use case diagram. It involves customer interactions for booking tickets, adding movies, selecting movies and viewing bookings.

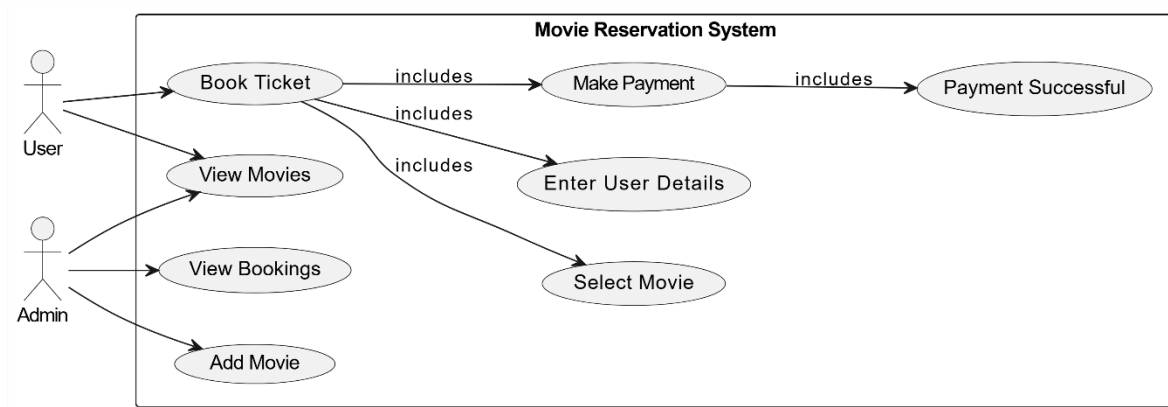


Figure 2.1. Use Case Diagram of Movie ticket Reservation System

2.3.2. ENTITY RELATIONSHIP DIAGRAM:

The **Entity Relationship Diagram (ERD)** for a **Movie Reservation System** provides a structured representation of the system's data entities, their relationships, and the rules governing how data interacts. The diagram helps to visualize how information about movies, users, reservations, and payments is organized and stored in the system.

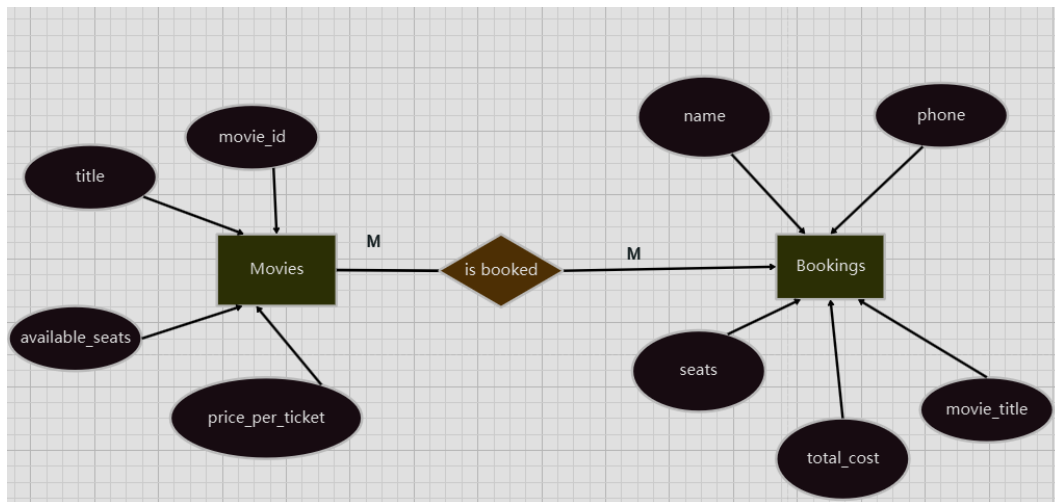


Figure 2.3.3 Entity Relationship diagram of Movie reservation System

2.3.3. DATA FLOW DIAGRAM:

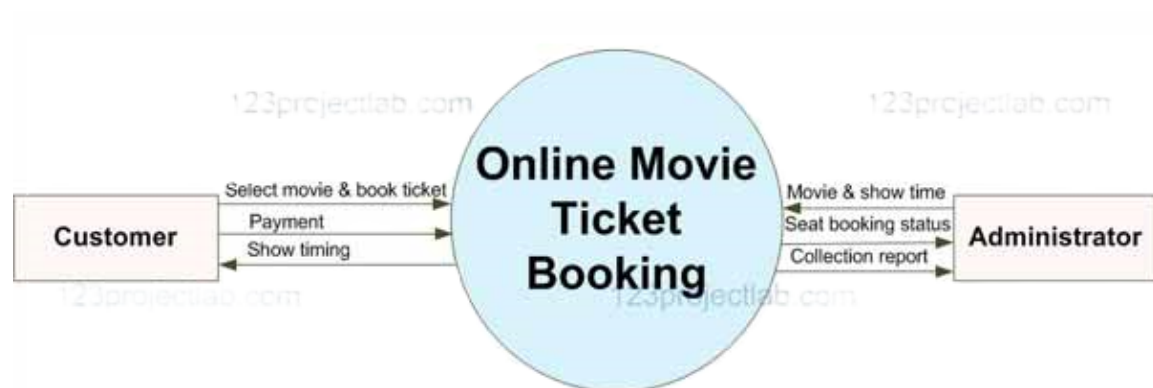


Figure 2.3 Data Flow Diagram

2.4. SYSTEM SPECIFICATION

Frontend:

1. Technology Used:

- **Java Swing** (or JavaFX): For creating the graphical user interface (GUI).

2. Key Features:

- **Login Page:** Allows users to log in with credentials.
- **Home Page:**
 - Displays two buttons: "Book Now" and "View Movies."
- **Movie Selection Page:**
 - Displays a list of movies retrieved from the database with details like title, price, and available seats.
- **Booking Page:**
 - Fields to input user details (name, phone, and number of seats).
 - A payment button to confirm the booking.
- **Admin Features:**
 - "Add Movie" option for adding new movies.
 - "View Bookings" option to check all user bookings.

3. Libraries/Tools:

- **TextField:** For input fields (e.g., name, phone, seats).
- **Button:** For actions (e.g., "Book Now," "Submit").
- **Table:** For displaying movies and bookings.

Backend:

1. Technology Used:

- **Java:** Core programming language.
- **JDBC (Java Database Connectivity):** To connect the Java application with MySQL.

2.Key Functionalities:

- **Database Connection:**

- Establishes a connection with the MySQL database using DriverManager.

Development Tools:

Integrated Development Environment (IDE)

- **Eclipse IDE:**

- Preferred IDE for Java development.
- Use the **Eclipse IDE for Enterprise Java Developers** for better features like Java EE tools.

Database Management System

- **MySQL Workbench:**

- For designing, managing, and querying the MySQL database.
- Provides a graphical interface for database modeling and query execution.
- Connect to the MySQL server locally or remotely.

2.4.1. FUNCTIONAL REQUIREMENTS

1. User Management:

- Users should be able to view a list of available movies with details such as title, price, and available seats.
- Users can select a movie and enter their booking details (name, phone number, and number of seats).
- Users should be able to make payments, and the system must confirm the booking upon successful payment.
- Users should be able to view the complete list of movies stored in the database.

2. Admin Functionalities

- Admins can add new movies with details such as title, price per ticket, and the number of available seats.
- Admins should be able to update movie details, such as the price per ticket or available seats.
- Admins should be able to view a list of all bookings made by users, including details like user name, phone number, movie title, and seats booked.

3. Payment Processing

- Calculate the total cost based on the number of seats and the ticket price.
- Simulate a payment gateway for processing payments (e.g., a confirmation message for simplicity).

CHAPTER 3

3.1 DESIGN

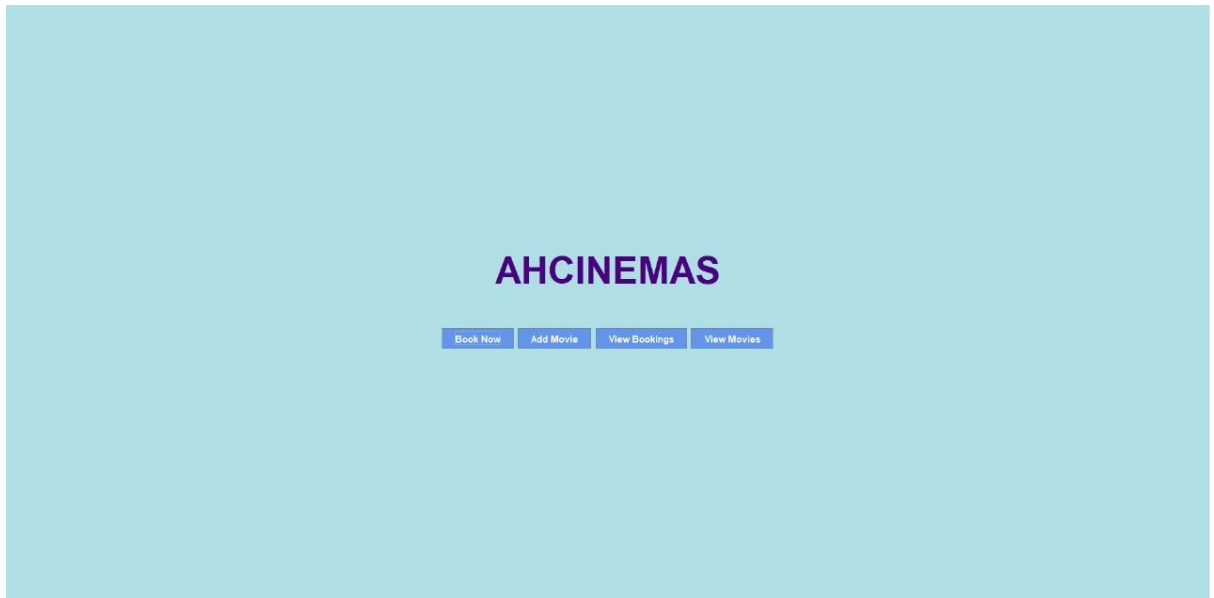


Figure 3.1. Home Page for AHcinemas

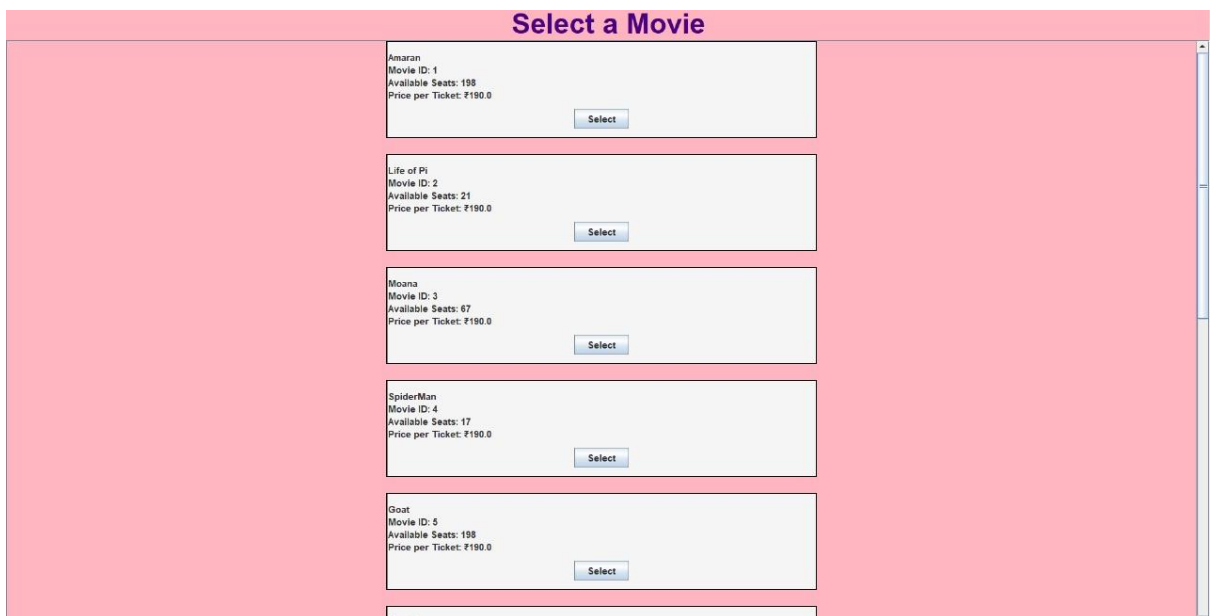
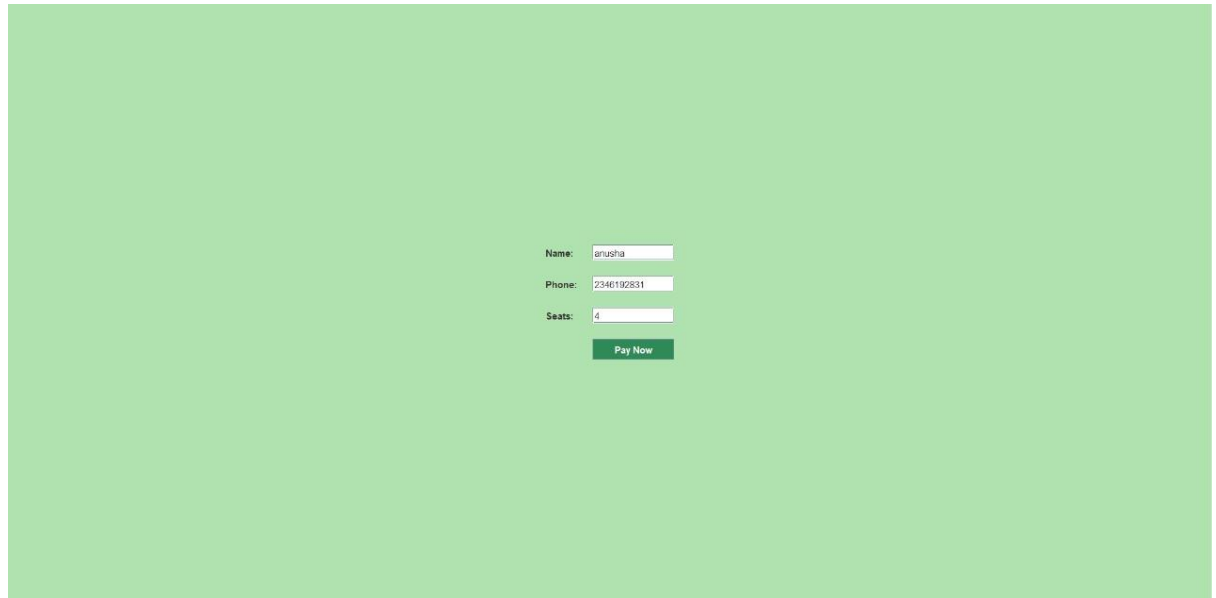


Figure 3.2. Selecting Movie Page

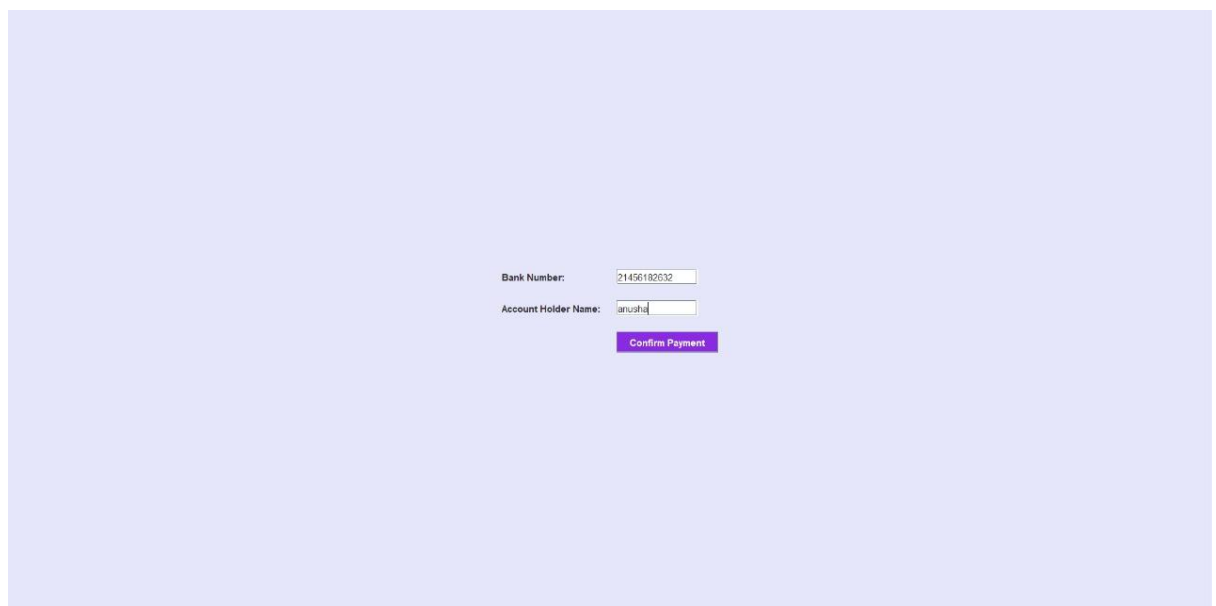
A screenshot of a login page with a light green background. In the center, there is a form with three input fields: 'Name' containing 'anusha', 'Phone' containing '2346192831', and 'Seats' containing '4'. Below these fields is a green button labeled 'Pay Now'.

Name:

Phone:

Seats:

Figure 3.3. Login Page

A screenshot of a payment page with a light blue background. In the center, there is a form with two input fields: 'Bank Number' containing '21456182632' and 'Account Holder Name' containing 'anusha'. Below these fields is a purple button labeled 'Confirm Payment'.

Bank Number:

Account Holder Name:

Figure 3.4. Payment Page

Reservation Successful!

Theater: AHCinemas
Movie: Sita Raman
Seats: 2
Total Cost: ₹380

Figure 3.5. Reservation Successful Page

Movie ID:

Title:

Available Seats:

Price per Ticket:

Figure 3.6. Movie Adding Page

All Movies in AHCINEMAS	
Amaran	
Movie ID: 1	
Available Seats: 198	
Price per Ticket: ₹190.0	
Life of PI	
Movie ID: 2	
Available Seats: 21	
Price per Ticket: ₹190.0	
Moana	
Movie ID: 3	
Available Seats: 67	
Price per Ticket: ₹190.0	
SpiderMan	
Movie ID: 4	
Available Seats: 17	
Price per Ticket: ₹190.0	
Goat	
Movie ID: 5	
Available Seats: 198	
Price per Ticket: ₹190.0	
Beast	
Movie ID: 18	
Available Seats: 96	
Price per Ticket: ₹190.0	
Brother	
Movie ID: 19	
Available Seats: 198	
Price per Ticket: ₹200.0	
Thor	
Movie ID: 21	
Available Seats: 200	
Price per Ticket: ₹120.0	
Titanic	
Movie ID: 25	
Available Seats: 200	
Price per Ticket: ₹190.0	
Asuran	
Movie ID: 28	
Available Seats: 200	
Price per Ticket: ₹190.0	
Sita Raman	
Movie ID: 30	
Available Seats: 200	
Price per Ticket: ₹190.0	

Figure 3.7. Movies List Page

Your Bookings	
Name: q	
Phone: 1	
Movie ID: Movie A	
Seats: 1	
Name: aaa	
Phone: 112	
Movie ID: Beast	
Seats: 1	
Name: a	
Phone: 1	
Movie ID: null	
Seats: 1	
Name: aa	
Phone: 11	
Movie ID: null	
Seats: 1	
Name: a	
Phone: 5	
Movie ID: null	
Seats: 1	
Name: anu	
Phone: 1232	
Movie ID: null	
Seats: 1	
Name: aa	
Phone: 1	
Movie ID: null	
Seats: 1	
Name: ss	
Phone: 12345	
Movie ID: null	
Seats: 1	
Name: aa	
Phone: 12344	
Movie ID: null	
Seats: 1	
Name: kai	
Phone: 765	
Movie ID: null	
Seats: 3	
Name: aa	
Phone: 11	
Movie ID: Dhoom	
Seats: 1	
Name: aa	
Phone: 111	
Movie ID: Amaran	

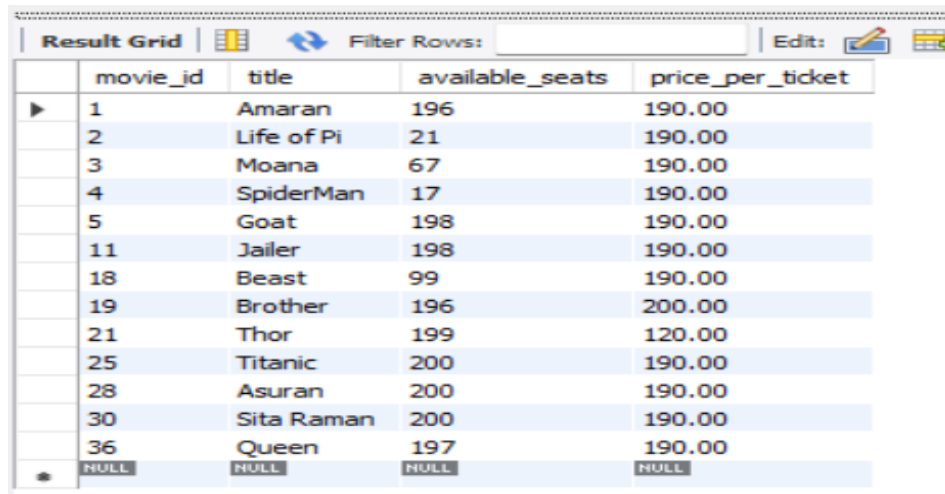
Figure 3.8. Bookings List Page

3.2 DATABASE DESIGN

1. Movies Table:

Contains details of movies available for booking.

- movie_id serves as the primary key.
- Other attributes include title, price_per_ticket, and available_seats.



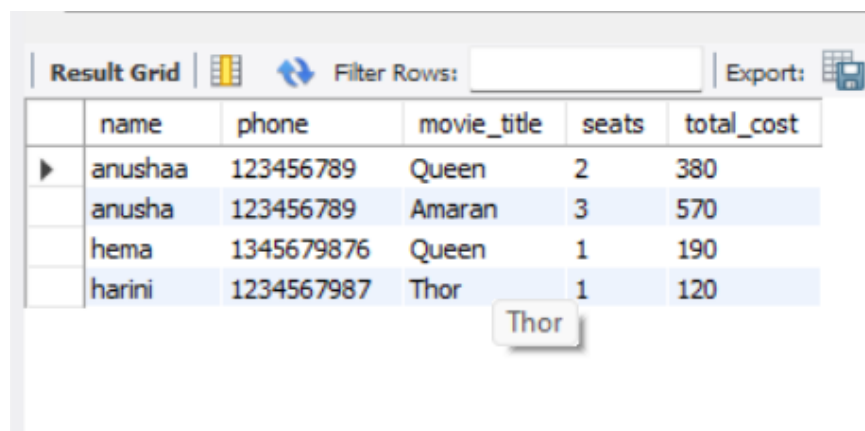
	movie_id	title	available_seats	price_per_ticket
▶	1	Amaran	196	190.00
	2	Life of Pi	21	190.00
	3	Moana	67	190.00
	4	SpiderMan	17	190.00
	5	Goat	198	190.00
	11	Jailer	198	190.00
	18	Beast	99	190.00
	19	Brother	196	200.00
	21	Thor	199	120.00
	25	Titanic	200	190.00
	28	Asuran	200	190.00
	30	Sita Raman	200	190.00
	36	Queen	197	190.00
•	NULL	NULL	NULL	NULL

Fig 3.2.1 Movies table

2. Bookings Table

Stores user booking details for movies.

- Attributes include name, phone, seats, total_cost, and movie_title.
- movie_title refers to the title column in the Movies Table, establishing a relationship.



	name	phone	movie_title	seats	total_cost
▶	anushaa	123456789	Queen	2	380
	anusha	123456789	Amaran	3	570
	hema	1345679876	Queen	1	190
	harini	1234567987	Thor	1	120

Fig 3.2.2 Bookings table

3.3 IMPLEMENTATION (CODE)

1.Home page

```
private JPanel createAHCinemasPanel() {

    JPanel ahcinemasPanel = new JPanel(new GridBagLayout());

    ahcinemasPanel.setBackground(PASTEL_BLUE);

    JLabel ahcinemasLabel = new JLabel("AHCINEMAS", JLabel.CENTER);
    ahcinemasLabel.setFont(new Font("Arial", Font.BOLD, 48));
    ahcinemasLabel.setForeground(new Color(75, 0, 130));

    JButton bookButton = new JButton("Book Now");
    bookButton.setBackground(new Color(100, 149, 237));
    bookButton.setForeground(Color.WHITE);
    bookButton.addActionListener(e -> cardLayout.show(mainPanel, "MovieListPanel"));

    JButton addMovieButton = new JButton("Add Movie");
    addMovieButton.setBackground(new Color(100, 149, 237));
    addMovieButton.setForeground(Color.WHITE);
    addMovieButton.addActionListener(e -> cardLayout.show(mainPanel, "AddMoviePanel"));

    JButton viewBookingsButton = new JButton("View Bookings");
    viewBookingsButton.setBackground(new Color(100, 149, 237));
    viewBookingsButton.setForeground(Color.WHITE);
    viewBookingsButton.addActionListener(e -> {
        populateBookings();
        cardLayout.show(mainPanel, "ViewBookingsPanel");});

    JButton viewMoviesButton = new JButton("View Movies");
    viewMoviesButton.setBackground(new Color(100, 149, 237));
    viewMoviesButton.setForeground(Color.WHITE);
    viewMoviesButton.addActionListener(e -> {
        populateAllMovies();
        cardLayout.show(mainPanel, "ViewAllMoviesPanel");
```

```
});
```

Setting database Connection:

```
private void setupDatabaseConnection() {  
    try {  
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/javamovie", "root",  
            "Anu2006@31");  
    } catch (SQLException e) {  
        e.printStackTrace();  
        JOptionPane.showMessageDialog(this, "Database Connection Failed!", "Error",  
            JOptionPane.ERROR_MESSAGE);  
    }  
}
```

View Bookings Page:

```
private void populateBookings() {  
    try (PreparedStatement ps = con.prepareStatement("SELECT * FROM bookings")) {  
        ResultSet rs = ps.executeQuery();  
        JPanel bookingsContainer = (JPanel) ((JScrollPane) ((JPanel)  
            mainPanel.getComponent(6)).getComponent(1)).getViewport().getView();  
        bookingsContainer.removeAll();  
        while (rs.next()) {  
            String bookingDetails = "<html>Name: " + rs.getString("name") + "<br>" + "Phone: " +  
                rs.getString("phone") + "<br>" + "Movie ID: " + rs.getString("movie_title") + "<br>" + "Seats: " +  
                rs.getInt("seats") + "</html>";  
            JLabel bookingLabel = new JLabel(bookingDetails);  
            bookingsContainer.add(bookingLabel);  
        }  
        bookingsContainer.revalidate()  
        bookingsContainer.repaint();  
    } catch (SQLException e) {  
        JOptionPane.showMessageDialog(this, "Error Loading Bookings: " + e.getMessage(), "Error",  
            JOptionPane.ERROR_MESSAGE);  
        e.printStackTrace();  
    }  
}
```

```
}
```

View Movies Page:

```
private void populateAllMovies() {  
    try (PreparedStatement ps = con.prepareStatement("SELECT * FROM movies")) {  
        ResultSet rs = ps.executeQuery();  
  
        JPanel moviesContainer = (JPanel) ((JScrollPane) ((JPanel)  
            mainPanel.getComponent(7)).getComponent(1)).getViewport().getView();  
        moviesContainer.removeAll();  
  
        while (rs.next()) {  
  
            String movieDetails = "<html><b>" + rs.getString("title") + "</b><br>" + "Movie ID: " +  
                rs.getString("movie_id") + "<br>" + "Available Seats: " + rs.getInt("available_seats") + "<br>" + "Price  
            per Ticket: ₹" + rs.getDouble("price_per_ticket") + "</html>";  
  
            JLabel movieLabel = new JLabel(movieDetails);  
  
            moviesContainer.add(movieLabel);  
        }  
  
        moviesContainer.revalidate();  
        moviesContainer.repaint();  
    } catch (SQLException e) {  
  
        JOptionPane.showMessageDialog(this, "Error Loading Movies: " + e.getMessage(), "Error",  
            JOptionPane.ERROR_MESSAGE);  
  
        e.printStackTrace();  
    }  
}
```

Adding a Movie Page:

```
private void addMovie() {  
    String movieId = movieIdField.getText();  
    String title = movieTitleField.getText();  
    int availableSeats;  
    double pricePerTicket;  
    try {  
        availableSeats = Integer.parseInt(availableSeatsField.getText());  
        pricePerTicket = Double.parseDouble(pricePerTicketField.getText());  
    } catch (NumberFormatException e) {
```

```

JOptionPane.showMessageDialog(this, "Invalid input for seats or price!", "Error",
JOptionPane.ERROR_MESSAGE);

return;

}

try (PreparedStatement ps = con.prepareStatement("INSERT INTO movies VALUES (?, ?, ?, ?)") {
ps.setString(1, movieId);
ps.setString(2, title);
ps.setInt(3, availableSeats);
ps.setDouble(4, pricePerTicket);
ps.executeUpdate();

JOptionPane.showMessageDialog(this, "Movie added successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);

movieIdField.setText("");
movieTitleField.setText("");
availableSeatsField.setText("");
pricePerTicketField.setText("");
} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Error Adding Movie: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);

e.printStackTrace();

}}

```

Reservation Successful Page:

```

private void makeReservation() {
try {
if (selectedMovie == null || selectedMovie.isEmpty()) {
JOptionPane.showMessageDialog(this, "Please select a movie before proceeding.");
return;
}

int numSeats = Integer.parseInt(numSeatsTextField.getText());

String priceQuery = "SELECT price_per_ticket, available_seats FROM movies WHERE title = ?";
PreparedStatement pstmt = con.prepareStatement(priceQuery);
pstmt.setString(1, selectedMovie);

ResultSet rs = pstmt.executeQuery();

```



```

if (rs.next()) {
    int pricePerTicket = rs.getInt("price_per_ticket");
    int availableSeats = rs.getInt("available_seats");
    if (numSeats > availableSeats) {
        JOptionPane.showMessageDialog(this, "Not enough seats available for this movie.");
        return;
    }
    int totalCost = numSeats * pricePerTicket;

    String updateSeatsQuery = "UPDATE movies SET available_seats = available_seats - ? WHERE title = ?";

    pstmt = con.prepareStatement(updateSeatsQuery);
    pstmt.setInt(1, numSeats);
    pstmt.setString(2, selectedMovie);
    pstmt.executeUpdate();

    String bookingQuery = "INSERT INTO bookings (name, phone, movie_title, seats, total_cost) VALUES (?, ?, ?, ?, ?)";

    pstmt = con.prepareStatement(bookingQuery);
    pstmt.setString(1, nameTextField.getText());
    pstmt.setString(2, phoneTextField.getText());
    pstmt.setString(3, selectedMovie);
    pstmt.setInt(4, numSeats);
    pstmt.setInt(5, totalCost);
    pstmt.executeUpdate();
    resultTextArea.setText(
        "Reservation Successful!\n\n" +
        "Theater: AHCinemas\n" +
        "Movie: " + selectedMovie + "\n" +
        "Seats: " + numSeats + "\n" +
        "Total Cost: ₹" + totalCost
    );
    cardLayout.show(mainPanel, "ResultPanel");
} else {
    JOptionPane.showMessageDialog(this, "Movie not found in the database.");
}

```

```
}  
} catch (SQLException e) {  
JOptionPane.showMessageDialog(this, "Failed to make reservation.\nError: " + e.getMessage());  
e.printStackTrace();  
} catch (NumberFormatException e) {  
JOptionPane.showMessageDialog(this, "Please enter a valid number of seats.");  
}  
}
```

CHAPTER 4

CONCLUSION

The Movie Reservation System effectively combines a user-friendly front-end interface built with Java and a robust backend powered by MySQL to handle movie bookings efficiently. The Movies Table stores information about the available movies, including titles, ticket prices, and seat availability, while the Bookings Table tracks customer booking details such as name, phone number, number of seats, and total cost. The system employs a clear one-to-many relationship between the two tables, ensuring data integrity and eliminating redundancy. Key functionalities include adding new movies, displaying available movies, booking tickets, and updating seat availability in real-time. The design is scalable, allowing for future enhancements like payment gateway integration, user authentication, and reporting features, making it a comprehensive solution for movie ticket reservations. This system demonstrates effective use of relational database principles and programming logic, providing a seamless booking experience for users and efficient management for administrators.

REFERENCES

- [1] <https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/>
- [2] <https://stackoverflow.com/questions/40685370/simple-cinema-booking-system>
- [3] <https://github.com/Movie-Ticket-Booking-Group/Movie-Ticket-Booking-Project>
- [4] <https://www.youtube.com/watch?v=ThTgN90PA44>
- [5] <https://www.oracle.com/technical-resources/articles/java/webapps.html>
- [6] <https://www.geeksforgeeks.org/java-database-connectivity-with-mysql/>
- [7] <https://www.javatpoint.com/example-to-connect-to-the-mysql-database>