RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

MOVIE TICKET BOOKING MANAGEMENT SYSTEM

MINI PROJECT REPORT

SUBMITTED BY:

MANISHAA G - 230701176

MADHUMITHA P - 230701169

CS23332 DATABASE MANAGEMENT SYSTEM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE
THANDALAM,CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

Certified that this project report " **MOVIE TICKET BOOKING SYSTEM**" is the bonafide work of **"MANISHAA G (230701176),MADHUMITHA P (230701169)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Dr.R.Sabitha
Professor and II year Academic Head
Science and Engineering,
Rajalakshmi EngineeringCollege
(Autonomous),Chennai-602105

SIGNATURE

Mrs.K.Maheshmeena

Assistant Professor(SG)
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous), Thandalam, Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT:

Online Movie Booking System is a web-based application designed to provide users with a seamless and efficient way to book movie tickets. The system enables customers to browse through available movies, view showtimes, select seats, and purchase tickets from the comfort of their homes or on-the-go. With a user-friendly interface, the platform allows moviegoers to easily check cinema listings, filter movies by genre, rating, or language, and make secure payments through integrated payment gateways. The system also supports features such as real-time seat availability, ticket cancellation, and booking history, ensuring a hassle-free experience. Additionally, cinema administrators can manage movie schedules, cinema locations, and ticket sales efficiently through a backend dashboard. This system enhances convenience, reduces long queues at the box office, and offers greater flexibility to both customers and cinema operators. The Online Movie Booking System aims to improve the overall movie-watching experience by providing an accessible, reliable, and time-saving solution for movie ticket bookings.

TABLE OF CONTENT

1. INTRODUCTION

1.1 General

This report tells about the **Movie Ticket Booking System**, a digital solution designed to streamline the process of purchasing movie tickets. It focuses on how the system allows users to easily browse available movies, check showtimes, select seats, and complete payments—all online, without the need to visit a cinema in person. The system is equipped with user-friendly features such as real-time seat availability, ticket cancellations, and viewing booking history. It also provides an admin interface for cinema operators to manage schedules, track sales, and update movie listings. By offering these functionalities, the Movie Ticket Booking System improves the customer experience by reducing wait times and eliminating the need for manual ticketing. Additionally, it helps cinema operators optimize operations, manage ticket sales efficiently, and gain insights into audience preferences. Ultimately, this system offers a convenient, accessible, and efficient way for customers to book movie tickets while enhancing overall cinema management.

1.2 Objectives

The main objectives of the Movie Ticket Booking System are as follows:

1. Simplify the Booking Process: To provide a user-friendly platform that allows customers to easily browse available movies, view showtimes, select seats, and complete ticket purchases online, thus eliminating the need for in-person visits to the cinema box office.

2. Enhance Convenience for Users: To enable moviegoers to book tickets anytime and from anywhere, offering a hassle-free experience without the need to wait in long queues or adhere to box office hours.

3. Provide Real-Time Information: To ensure that users have access to real-time information regarding movie availability, showtimes, and seat occupancy, allowing them to make informed decisions when booking tickets.

4. Offer Secure Payment Options: To integrate reliable and secure payment gateways that ensure smooth, safe transactions for users when purchasing tickets online.

5. Streamline Cinema Operations: To help cinema administrators efficiently manage movie schedules, seating arrangements, ticket pricing, and sales tracking, improving operational efficiency and reducing manual tasks.

1.3 Scope

The scope of this Movie ticket booking System

1. User Functionality

The system enables users to browse available movies, view detailed information, and select showtimes. Users can choose from available seats in real-time and book tickets securely online. A payment gateway allows for seamless transactions, supporting various payment methods. Booking history allows users to track past tickets and manage future bookings. Notifications keep users informed about booking confirmations, reminders, and changes to showtimes**.**

2. Admin Functionality

Admins can manage movie listings, update schedules, and set ticket prices. The system allows admins to monitor seat availability and adjust cinema seating arrangements as needed. Admins have access to sales reports, enabling them to track revenue and optimize movie schedules. The platform also provides tools for user account management, ensuring smooth operation of the system. Additionally, admins can run promotional campaigns and offer discounts to increase ticket sales.

3. System Features and Integration

The system supports real-time seat availability, ensuring users can only book available seats. It is designed to work across both web and mobile platforms, enhancing accessibility. The platform integrates with third-party payment gateways for secure online transactions. The system will be scalable to support multiple cinema locations and various movie genres. Future extensions may include loyalty programs and advanced reporting tools to further enhance the user experience.

2. SYSTEM OVERVIEW

2.1 System Architecture

1. Frontend (Java Swing)

The **Client Layer** is responsible for user interactions, where the user interacts with the system through a graphical user interface (GUI) built using **Java Swing**. This layer handles user actions such as movie browsing, showtime selection, seat reservation, and payment processing. It communicates with the backend using **JDBC** to fetch movie details, seat availability, and booking status, and to send booking data for processing.

2. Backend (MySQL Database)

The **Backend Layer** processes business logic and handles all core operations. When the user selects a movie, seat, or makes a booking, the backend verifies availability, handles the booking process, and interacts with the database. It also integrates with third-party payment gateways (such as PayPal or Stripe) for secure payment transactions. The backend communicates with the frontend through **JDBC** or **APIs**, processing data and sending back responses like booking confirmations, seat updates, or payment success messages.

3. Database:

The **Data Layer** (or Database Layer) is the foundation where all persistent data is stored. This includes movie listings, showtimes, user profiles, and booking records. The **MySQL** database stores all this data and ensures data integrity and consistency. The backend queries and updates the database as users interact with the system, such as retrieving available movies or updating seat availability after a successful booking.

Together, these layers form a robust architecture where the frontend interacts with users, the backend processes logic and business rules, and the database stores and retrieves essential data. This architecture ensures efficient, secure, and scalable operations, allowing users to seamlessly browse movies, book tickets, and manage their bookings while cinema administrators can control showtimes, movies, and track sales data.

2.2 Modules Overview

The Movie Ticket Booking System is composed of several key modules, each serving a specific function within the system. The primary modules include:

The **User Registration and Authentication Module** allows users to register and log into the system. It validates user credentials and handles user sessions to provide secure access to the system. Users can also reset their passwords if necessary.

The **Movie and Show Time Management Module** manages movie listings, showtimes, and related information. It enables users to view available movies, along with their details such as title, genre, cast, and description. The system allows filtering and sorting by genres, ratings, and release dates. Admins can add, update, or remove movie details and showtimes.

The **Seat Reservation and Availability Module** allows users to view and select available seats for a particular show. It ensures that only available seats are displayed in real-time and prevents double booking by locking selected seats temporarily during the booking process. Once a booking is completed, the seat status is updated accordingly.

The **Booking and Ticket Management Module** handles the process of booking tickets. Users can select their desired seats, enter their payment information, and confirm the booking. Once the booking is confirmed, the system generates a digital ticket or booking confirmation. Users can also view their booking history and cancel or modify bookings if allowed by the system.

The **Payment Processing Module** ensures secure online payments. It integrates with third-party payment gateways like PayPal or Stripe to handle financial transactions. The module validates payment information, processes transactions securely, and confirms successful payments before completing the ticket booking process.

The **Admin Panel Module** allows cinema administrators to manage and control the system. Admins can add new movies, set showtimes, and update seat availability. The

panel also provides tools for viewing booking data, managing user accounts, and generating reports on sales and movie performance.

The **Notification and Alert Module** sends notifications to users regarding booking confirmations, reminders for upcoming showtimes, and any changes to their bookings. The system can notify users via email or SMS about booking status, seat availability, and any relevant updates.

The **Reporting and Analytics Module** generates detailed reports on booking statistics, revenue, and user behavior. Admins can access sales reports, view popular movies, track peak booking times, and analyze overall system performance. These insights help improve business operations and marketing strategies.

The **Search and Filter Module** enables users to search for movies by title, actor, genre, or other criteria. It also provides filters to narrow down results by language, rating, release date, and more, making it easier for users to find movies they’re interested in.

3. SURVEY OF TECHNOLOGIES

3.1 Software and Tools Used

Java Swing

- **Purpose**: Used for building the graphical user interface (GUI) of the system. Java Swing provides components like buttons, labels, and tables to design a desktop-based application for users to interact with.
- **Role**: Enables the frontend of the system, allowing users to browse movies, select seats, and complete bookings.

MySQL Database

- **Purpose**: A relational database management system used to store all the data related to the movie booking system, such as movies, showtimes, bookings, and user information.
- **Role**: Stores persistent data and handles queries related to movie listings, seat availability, user accounts, and transactions.

JDBC (Java Database Connectivity)

- **Purpose**: A Java API that allows the Java application (frontend) to interact with the MySQL database.
- **Role**: Facilitates communication between the frontend (Java Swing) and backend (MySQL) to retrieve, update, and manage data.

3.2 Programming Languages

The **Movie Ticket Booking System** employs a combination of programming languages and technologies, each serving a specific function within the application to provide a seamless experience for users and efficient operation for administrators:

- **SQL (MySQL)**: SQL is used for managing the relational database that stores movie details, showtimes, user accounts, and booking records. MySQL enables efficient querying and updating of data, ensuring real-time seat availability, accurate user records, and transaction management. It allows the system to

perform complex queries to retrieve movie information, manage bookings, and handle payments.

- **Java**: Java powers the backend logic and handles the system's core functionalities, including user authentication, seat reservation, booking processing, and payment integration. Known for its scalability and reliability, Java processes tasks such as checking seat availability, confirming ticket bookings, and generating booking details. It also ensures the smooth operation of backend features, including data validation, updates to the database, and interaction with payment gateways.
- **Java Swing**: Java Swing is used to design and implement the graphical user interface (GUI) of the desktop application. It allows users to interact with the system through a user-friendly interface where they can browse movies, select seats, and complete ticket purchases. Swing components provide a responsive layout for users to easily navigate the system and manage their bookings.

4. REQUIREMENTS AND ANALYSIS

4.1 Functional Requirements

The system must allow users to browse movies, select showtimes, choose seats, and book tickets.

Users should be able to register, log in, and make secure payments for ticket bookings.

4.2Non-Functional Requirements

The system should ensure high availability with minimal downtime, providing a responsive user experience.

It must be scalable to handle increasing user traffic and support multiple simultaneous bookings without performance degradation.

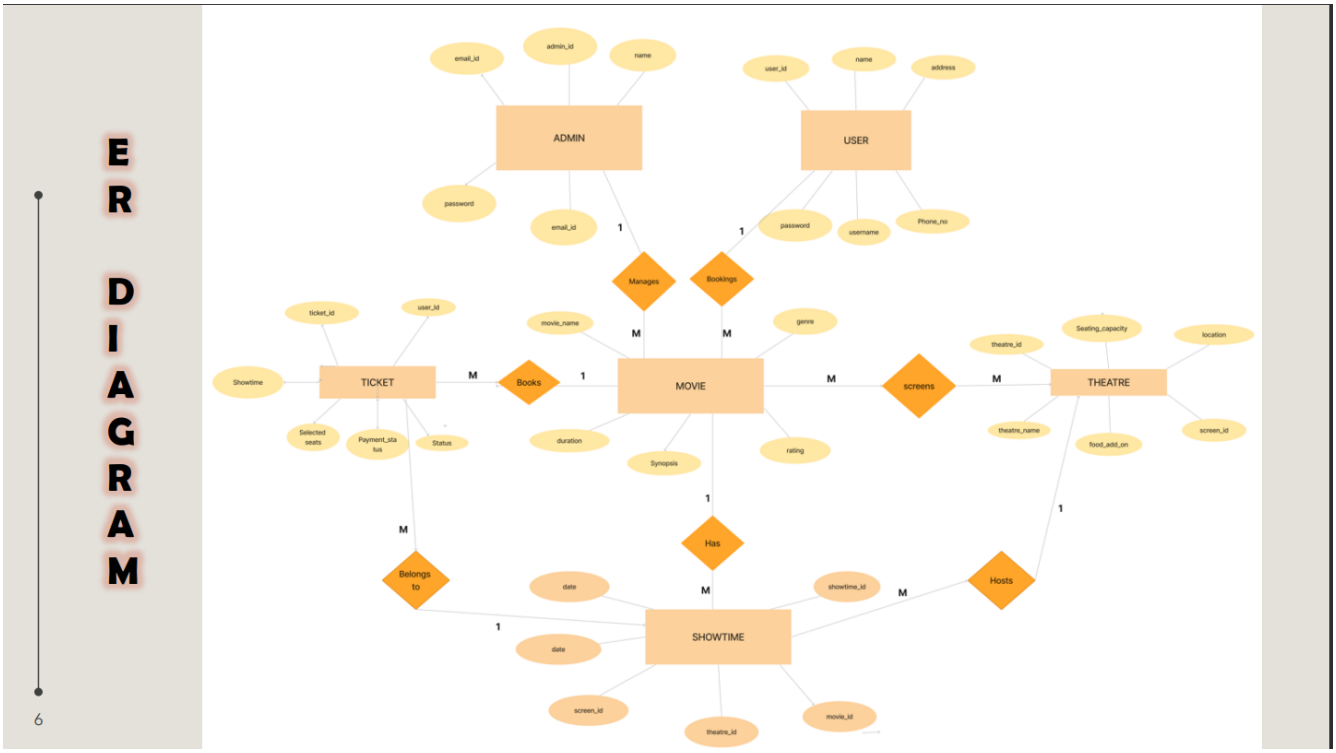
4.3 Hardware and Software Requirements

Hardware: Standard PC or server with internet access,Processor, Ram ,Storage

Software: Web browser, Java Swing, MySQL Database,JDBC

4.5 ER Diagram

An Entity-Relationship (ER) diagram maps out the database structure, showing tables such as Admin,User, Movie,Ticket, Theatre and Showtime.



5. IMPLEMENTATION

5.1 Program Code: (sample code)

```
public class MovieTicketBookingSystem {

    private static DatabaseOperation db = new DatabaseOperation();

    private static int loggedInUserID = -1; // User session management


    public static void main(String[] args) {

        showMainMenu();

    }

//inserting

String sql = "INSERT INTO movies (movie_ID, title, genre, Duration, Synopsis, rating) VALUES (?, ?, ?, ?, ?, ?)";

            Object[] values = {movie_id,title, genre, duration, synopsis, rating};

            int rowsAffected = db.executeUpdate(sql, values);


            if (rowsAffected > 0) {

                JOptionPane.showMessageDialog(addMovieFrame, "Movie added successfully!");

            } else {

                JOptionPane.showMessageDialog(addMovieFrame, "Failed to add the movie.");

            }

            addMovieFrame.dispose();

            showAdminDashboard();
```

```
    });


String sql = "INSERT INTO theatre (theatre_id,theatre_name, seating_capacity, location, screen_id,food_add_on) VALUES (?,?,?, ?, ?, ?)";

        Object[] values = {theatreID, name, capacity, location, screenID,foodaddon};

        int rowsAffected = db.executeUpdate(sql, values);



        if (rowsAffected > 0) {

            JOptionPane.showMessageDialog(addTheaterFrame, "Theater added successfully!");

        } else {

            JOptionPane.showMessageDialog(addTheaterFrame, "Failed to add the theater.");

        }

        addTheaterFrame.dispose();

        showAdminDashboard();

    });

String sql = "INSERT INTO showtimes (showtime_id, movie_ID, theatre_id, screen_id, day, date) VALUES (?,?, ?, ?, ?, ?)";

        Object[] values = {showtimeID,movieID, theaterID, screenID, day, date};

        int rowsAffected = db.executeUpdate(sql, values);



        if (rowsAffected > 0) {

            JOptionPane.showMessageDialog(addShowtimeFrame, "Showtime added successfully!");

        } else {

            JOptionPane.showMessageDialog(addShowtimeFrame, "Failed to add the
```

```
showtime.");
            }
            addShowtimeFrame.dispose();
            showAdminDashboard();
        });


int movieID = (int) movie.get("movie_ID");
        List<Map<String, Object>> showtimes = db.getRecords("SELECT * FROM showtimes WHERE movie_ID = " + movieID);


String sql = "SELECT UserID, Password FROM users WHERE Username = ?";
        Map<String, Object> user = db.validatePass(sql, username);


        String sql = "INSERT INTO tickets (UserID, showtime_id, selected_seats, payment_status, availability_status) VALUES (?, ?, ?, 'Paid', 'Confirmed')";
        Object[] values = {loggedInUserID, showtimeID, selectedSeats};
//database operation
package movieticketbooking;
import java.sql.*;
import java.util.*;
import javax.swing.*;

public class DatabaseOperation {
    static final String DB_URL = "jdbc:mysql://localhost/moviedb";
    static final String USER = "root";
    static final String PASS = "GsJm$2408";
```

```
    public Connection connectToDatabase() {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Database connection failed: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
        return conn;
    }

    public int executeUpdate(String sql, Object[] values) {
        int rowsAffected = 0;
        try (Connection conn = connectToDatabase();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            for (int i = 0; i < values.length; i++) {
                ps.setObject(i + 1, values[i]);
            }
            rowsAffected = ps.executeUpdate();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "SQL Update Failed: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
        return rowsAffected;
    }
```

```
    public List<Map<String, Object>> getRecords(String sql) {
        List<Map<String, Object>> records = new ArrayList<>();
        try (Connection conn = connectToDatabase();
             PreparedStatement pstmt = conn.prepareStatement(sql);
             ResultSet rs = pstmt.executeQuery()) {
            ResultSetMetaData rsmd = rs.getMetaData();
            int columnCount = rsmd.getColumnCount();
            while (rs.next()) {
                Map<String, Object> row = new HashMap<>();
                for (int i = 1; i <= columnCount; i++) {
                    row.put(rsmd.getColumnName(i), rs.getObject(i));
                }
                records.add(row);
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "SQL Query Failed: " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
        }
        return records;
    }

    public int getSeatingCapacity(String sql, int parameter) {
        int seatingCapacity = 0;
        try (Connection conn = connectToDatabase();
```

```
            PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, parameter);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                seatingCapacity = rs.getInt("SeatingCapacity");
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error fetching seating capacity: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
        return seatingCapacity;
    }

    public ArrayList<Integer> getBookedSeats(int showtimeID) {
        String sql = "SELECT SelectedSeats FROM bookings WHERE ShowtimeID = ?";
        ArrayList<Integer> bookedSeats = new ArrayList<>();
        try (Connection conn = connectToDatabase();
            PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, showtimeID);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                bookedSeats.add(rs.getInt("SelectedSeats"));
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error fetching booked seats: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
```

```
        }
        return bookedSeats;
    }

    public int removeBooking(int bookingID) {
        String sql = "DELETE FROM bookings WHERE BookingID = ?";
        int rowsAffected = 0;
        try (Connection conn = connectToDatabase();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, bookingID);
            rowsAffected = ps.executeUpdate();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error removing booking: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
        return rowsAffected;
    }

    public Map<String, Object> validatePass(String sql, Object... params) {
        Map<String, Object> result = new HashMap<>();
        try (Connection conn = connectToDatabase();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            for (int i = 0; i < params.length; i++) {
                ps.setObject(i + 1, params[i]);
            }
```

```
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            ResultSetMetaData metaData = rs.getMetaData();
            int columnCount = metaData.getColumnCount();
            for (int i = 1; i <= columnCount; i++) {
                result.put(metaData.getColumnName(i), rs.getObject(i));
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return result.isEmpty() ? null : result; // Return null if no record found
}


public int fetchUserID(String sql, String username) {
    int userID = 0;
    try (Connection conn = connectToDatabase();
         PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, username);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            userID = rs.getInt("UserID");
        }
    } catch (SQLException e) {
```

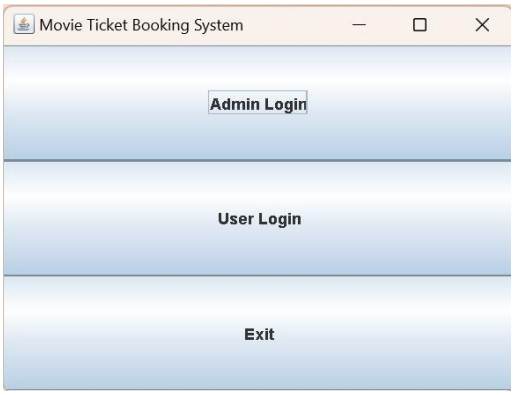
```
            JOptionPane.showMessageDialog(null, "Error fetching UserID: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);

        }

        return userID;

    }

}
```

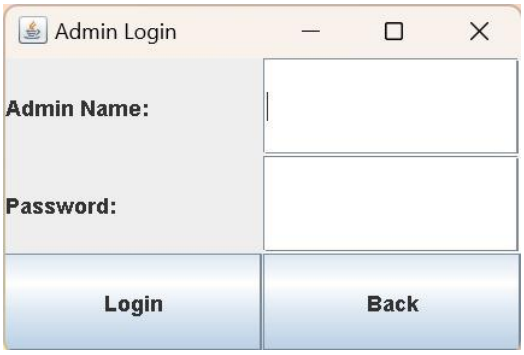
6. RESULTS AND DISCUSSION

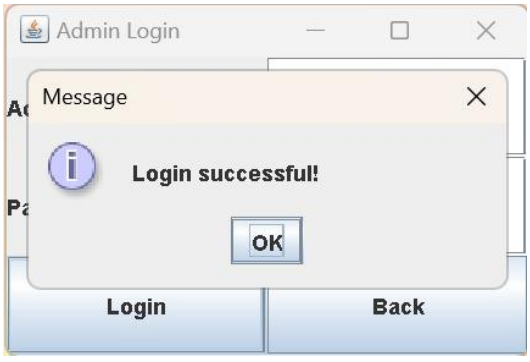
6.1 Summary of Features

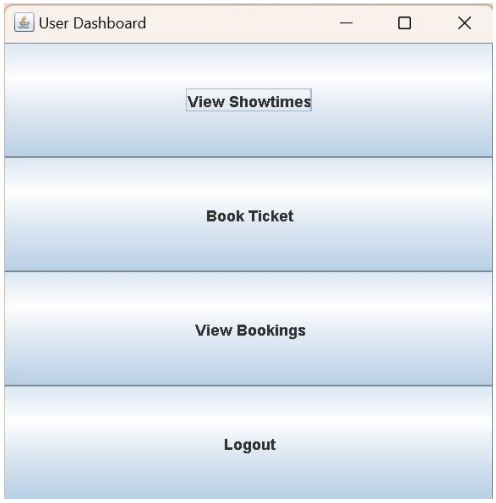
The **Movie Ticket Booking System** offers a range of features designed to provide users with an easy, convenient, and secure way to book movie tickets. Users can register and log in to the system, allowing them to manage their accounts and access personalized services. The system provides a comprehensive movie browsing experience, where users can view available movies, check showtimes, and select seats for their desired show. Once users select their seats, they can proceed with booking the tickets and make secure payments through integrated payment gateways like PayPal or Stripe. Additionally, users can view their booking history and access details of past bookings. The system also includes an admin panel where administrators can manage movie listings, update showtimes, and oversee user bookings. Real-time seat availability ensures that users can only book available seats, and the search and filter functionality makes it easy for users to find specific movies based on criteria like genre, name, and showtime. Finally, users receive email notifications confirming their bookings, which include ticket details and payment receipts. These features combine to deliver a seamless, user-friendly experience for both customers and administrators.

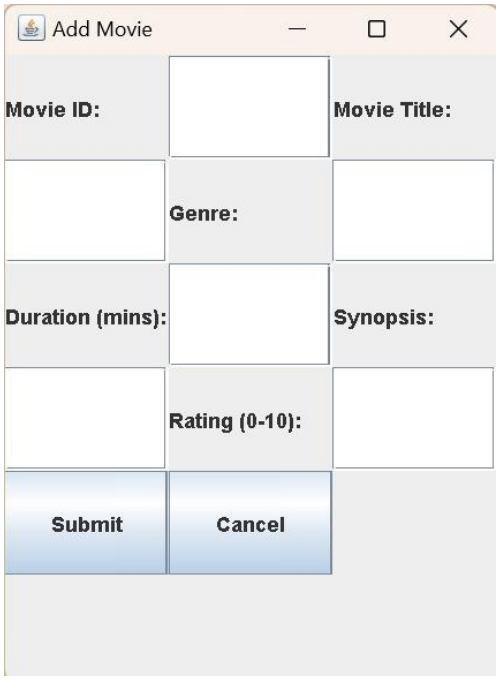
6.2 OUTPUT:

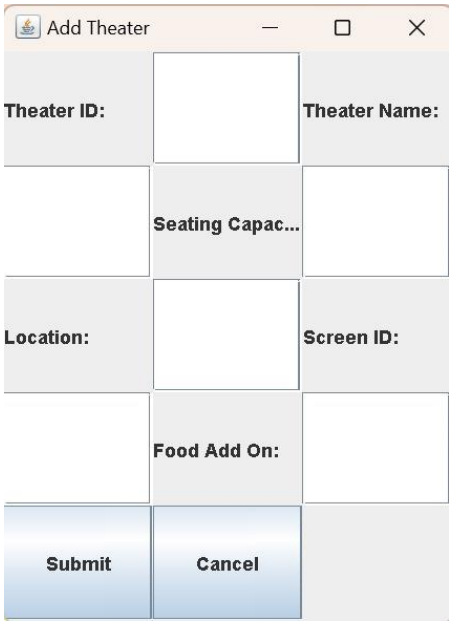

Movie Ticket Booking System
Admin Login
User Login
Exit


Admin Login
Admin Name:
Password:
Login
Back


Admin Login
Message
Login successful!
OK
Login
Back

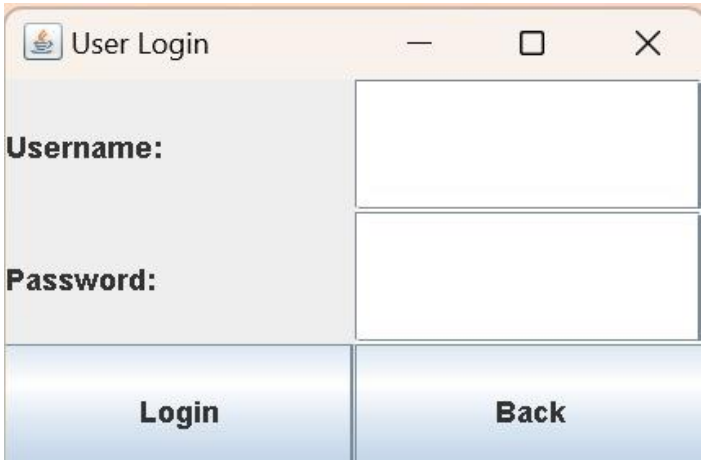

User Dashboard
View Showtimes
Book Ticket
View Bookings
Logout

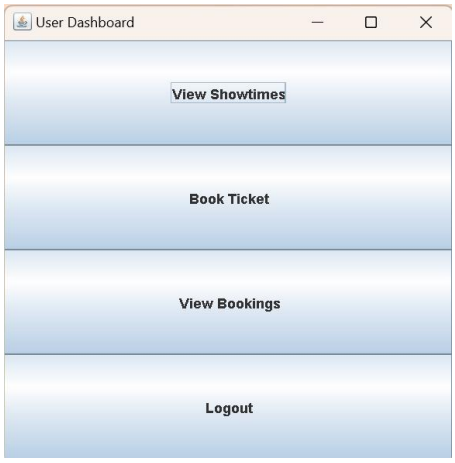

Add Movie
Movie ID:
Movie Title:
Genre:
Duration (mins):
Synopsis:
Rating (0-10):
Submit
Cancel


Add Theater
Theater ID:
Theater Name:
Seating Capac...
Location:
Screen ID:
Food Add On:
Submit
Cancel

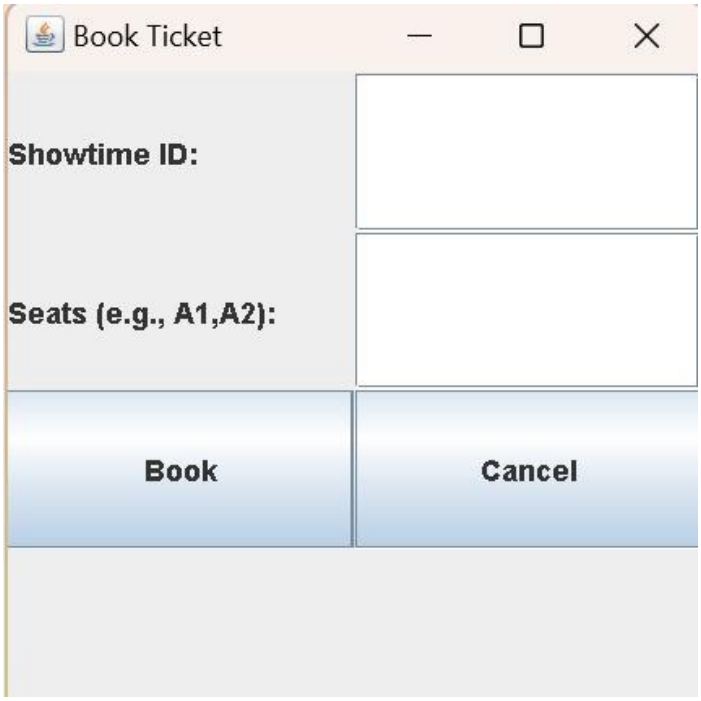

Add Showtime
Showtime ID:
Movie ID:
Theater ID:
Screen ID:
Day:
Date (YYYY-M...
Submit
Cancel

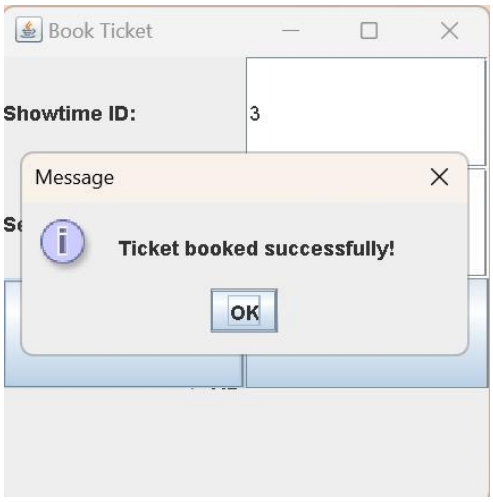

Movies and Showtimes
Movie ID: 1, Title: Inception, Genre: Science Fiction, Duration: 2.5, Rating: 8.8
Showtime ID: 1, Theater ID: 1, Screen ID: 1, Day: Friday, Date: 2024-11-15T09:00
Movie ID: 2, Title: The Shawshank Redemption, Genre: Drama, Duration: 2.4, Rating: 9.3
Showtime ID: 2, Theater ID: 2, Screen ID: 2, Day: Saturday, Date: 2024-11-16T09:00
Movie ID: 3, Title: The Dark Knight, Genre: Action, Duration: 2.5, Rating: 9.0
Showtime ID: 3, Theater ID: 3, Screen ID: 3, Day: Sunday, Date: 2024-11-17T09:00
Showtime ID: 6, Theater ID: 2, Screen ID: 2, Day: friday, Date: 2024-12-04T00:00
Movie ID: 4, Title: The Godfather, Genre: Crime, Duration: 2.9, Rating: 9.2
Showtime ID: 4, Theater ID: 4, Screen ID: 4, Day: Monday, Date: 2024-11-18T09:00
Movie ID: 6, Title: amaran, Genre: patriotic, Duration: 90.0, Rating: 8.7
Back

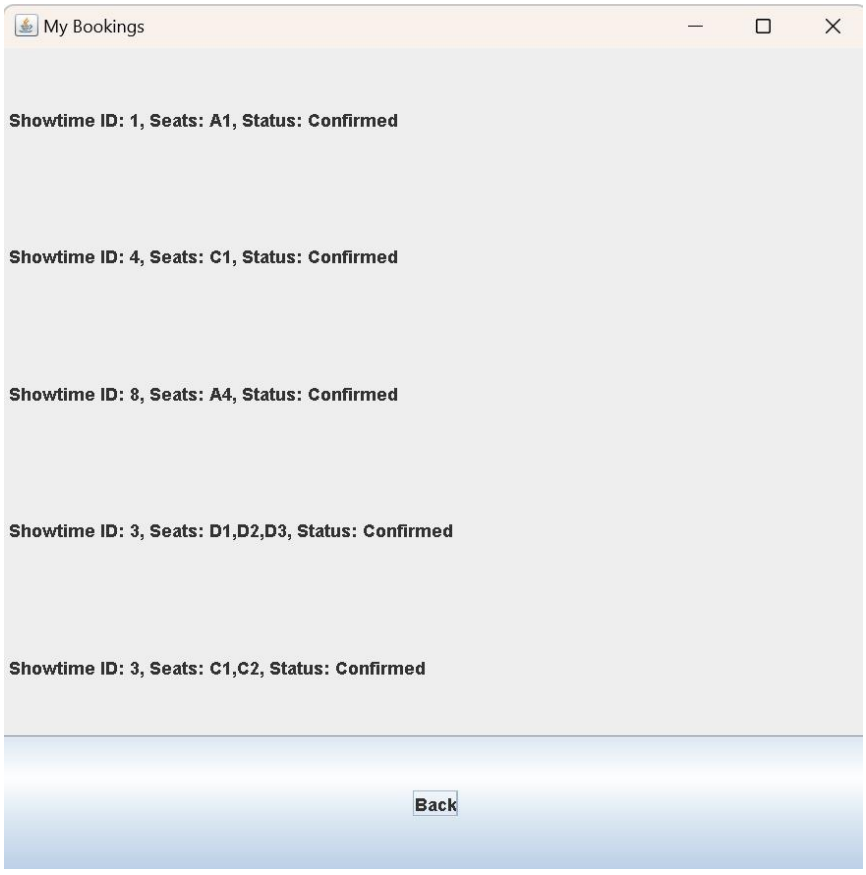

User Login
Username:
Password:
Login
Back


User Dashboard
View Showtimes
Book Ticket
View Bookings
Logout


Available Showtimes
Showtime ID: 1, Movie ID: 1, Theater ID: 1, Screen ID: 1, Day: Friday, Date: 2024-11-15T09:00
Showtime ID: 2, Movie ID: 2, Theater ID: 2, Screen ID: 2, Day: Saturday, Date: 2024-11-16T09:00
Showtime ID: 3, Movie ID: 3, Theater ID: 3, Screen ID: 3, Day: Sunday, Date: 2024-11-17T09:00
Showtime ID: 4, Movie ID: 4, Theater ID: 4, Screen ID: 4, Day: Monday, Date: 2024-11-18T09:00
Showtime ID: 6, Movie ID: 3, Theater ID: 2, Screen ID: 2, Day: friday, Date: 2024-12-04T00:00
Back


Book Ticket
Showtime ID:
Seats (e.g., A1,A2):
Book
Cancel


Book Ticket
Showtime ID:
3
Message
Ticket booked successfully!
OK


My Bookings
Showtime ID: 1, Seats: A1, Status: Confirmed
Showtime ID: 4, Seats: C1, Status: Confirmed
Showtime ID: 8, Seats: A4, Status: Confirmed
Showtime ID: 3, Seats: D1,D2,D3, Status: Confirmed
Showtime ID: 3, Seats: C1,C2, Status: Confirmed
Back

7. CONCLUSION

The **Movie Ticket Booking System** provides an efficient and user-friendly platform for booking movie tickets, offering seamless access to movie details, showtimes, and seat availability. Designed to enhance the user experience, it allows customers to browse movies, reserve seats, and complete bookings with ease. The system integrates secure payment gateways, ensuring a smooth and safe transaction process.

At the core of the system is an intuitive interface that allows users to quickly find and select their desired movies. The admin panel enables administrators to manage movie listings, showtimes, and user bookings, ensuring smooth operations. Real-time seat availability and booking confirmations further enhance the user experience by preventing overbookings and ensuring accurate information.

Developed using Java Swing for the frontend and MySQL for the backend, the system guarantees reliability and performance. Its modular structure allows for scalability, making it suitable for handling increasing traffic and expanding features such as advanced search filters, loyalty programs, and mobile integrations in the future. Overall, the Movie Ticket Booking System is a comprehensive and versatile tool, designed to meet the needs of both users and administrators while providing a smooth, enjoyable movie booking experience.