MOVIE-TICKET BOOKING

A MINI-PROJECT REPORT

Submitted by

S SAI ARAVIND 230701275

R SARVESH 230701295

In partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project “MOVIE-TICKET BOOKING” is the

bonafide work of “S SAI ARAVIND (230701275), R SARVESH(230701295)” who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Mr. G SARAVANA GOKUL Assistant Professor (SS), Computer Science and Engineering, Rajalakshmi Engineering College (Autonomous), Thandalam, Chennai-602105	Ms. V. JANANEE Assistant Professor (SG), Computer Science and Engineering, Rajalakshmi Engineering College (Autonomous), Thandalam, Chennai-602105

This mini project report is submitted for the viva voce examination to be held on ______

TABLE OF CONTENTS

ABSTRACT

The Movie Ticket Booking System is a modern digital solution designed to streamline the process of reserving, purchasing, and managing movie tickets. This system aims to enhance user convenience, reduce manual effort, and provide a seamless experience for both customers and theatre administrators.

The system allows users to browse movie schedules, select preferred showtimes, and reserve seats in realtime via a web or mobile interface. Features include interactive seat selection, secure payment integration, and notifications for booking confirmations or upcoming shows. Advanced functionalities, such as filters for movie genres, user reviews, loyalty rewards, and personalized recommendations, add value to the customer experience.

For administrators, the system offers tools to manage movie schedules, seat availability, pricing, and promotions efficiently. Analytics dashboards provide insights into ticket sales, audience preferences, and occupancy trends, enabling data-driven decisionmaking.

By leveraging cutting-edge technologies, such as cloud computing and APIs, the Movie Ticket Booking System

ensures high availability, scalability, and secure transactions. It bridges the gap between theatres and moviegoers, transforming the traditional ticketing process into a hassle-free digital experience.

This solution not only improves operational efficiency for cinemas but also fosters customer satisfaction, contributing to higher engagement and revenue growth in the entertainment industry.

INTRODUCTION

In a rapidly evolving digital age, the integration of technology into various aspects of our lives has become almost inevitable. One area where technology can make a profound impact is the electoral process. The Movie Ticket Booking System is a Java-based desktop application designed to automate and simplify the process of booking movie tickets. This system integrates a user-friendly graphical interface built with Java Swing and a robust backend for data storage and retrieval using JDBC (Java Database Connectivity) with a relational database like MySQL or PostgreSQL.

Key Features

1. User-Friendly Interface:
   The application leverages Java Swing to create an intuitive and visually appealing interface for users to browse movies, select showtimes, and book tickets seamlessly.
2. Database-Driven Functionality:

JDBC ensures efficient interaction between the application and the database, enabling real-time updates for seat availability, booking records, and user profiles.

3. Dynamic Seat Selection:
   Users can view seat layouts, check seat availability, and make reservations with interactive seat selection features.
4. Authentication and Role Management: The system supports user registration and login functionalities, with distinct roles for customers and administrators. Administrators can manage movie schedules, add new movies, and monitor ticket sales.
5. Real-Time Updates:

   Seat availability and booking status are updated in realtime to prevent overbooking or inconsistencies. Technical Overview

- Frontend: Developed using Java Swing to provide a platform-independent graphical interface.
- Backend: JDBC is utilized to connect the application to the database, enabling CRUD operations (Create, Read, Update, Delete) on tickets, movies, and user data.
- Database: A relational database stores all relevant data, such as user details, movie information, showtimes, and booking records.

SCOPE OF THE PROJECT

The Movie Ticket Booking System using Java Swing and JDBC serves as a comprehensive solution for automating the ticketing process for movie theatres. The scope of this project encompasses the development, deployment, and potential future enhancements that can make it a scalable, efficient, and user-friendly system. Below is a breakdown of the project's scope:

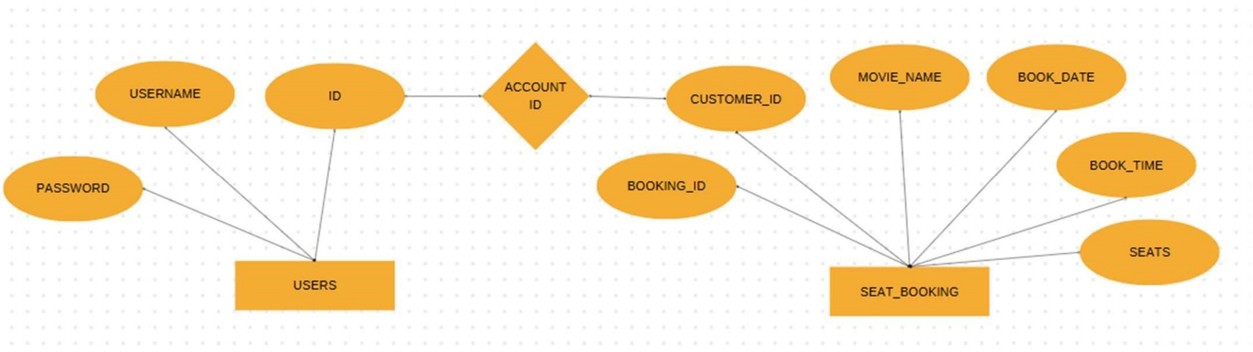
---

1. Functional Scope

- User Registration and Login:
  - Allows users to create accounts, log in, and manage personal profiles.
  - Provides role-based authentication (e.g., Customer, Administrator).
- Movie Browsing and Information:
  - Users can browse movies currently showing, their schedules, and other details (e.g., genre, duration, director).
  - Search functionality for movie titles or genres.
- Showtime and Seat Selection:
  - Users can view available showtimes for each movie and select preferred shows.
  - Interactive seat map that displays available, booked, and selected seats.
- Booking and Payment:
  - Allows customers to book movie tickets and confirm bookings.

    - Basic payment processing (or tracking) for ticket purchases.
- Booking History and Notifications:
    - Users can view past bookings and upcoming shows.
    - Booking confirmation and reminder notifications.
- Admin Functionality:
    - Admin users can manage movie schedules, showtimes, and seat availability.
    - Ability to add new movies, update details, and delete outdated listings.
- Reports and Analytics:
    - Admins can view booking statistics, occupancy rates, and other key metrics to analyze performance and trends.

ER DIAGRAMS



PROGRAM:-

```
import javax.swing.*; import java.awt.*;

import java.awt.event.ActionEvent; import

java.awt.event.ActionListener; import

java.sql.*; import java.util.ArrayList; import

java.text.SimpleDateFormat; import

javax.swing.event.DocumentEvent; import

javax.swing.event.DocumentListener; import

java.util.Calendar; import

java.time.LocalTime; import java.util.List;

public class Moviebooking extends JFrame{

public static JFrame frame;     public static

JPanel panel;

   private static Connection connect() {

       try {

           return
DriverManager.getConnection("jdbc:mysql://localhost:3306/db","root","Sai@
30.");

       } catch (SQLException e) {

e.printStackTrace();            return

null;

       }

   }

   private static void initializeDatabase() {

try{
```

```
            Connection conn = connect();
            Statement stmt = conn.createStatement();
            String sql = "CREATE TABLE IF NOT EXISTS users (" +
                    "id INTEGER PRIMARY KEY AUTO_INCREMENT, " +
                    "username varchar(50) NOT NULL UNIQUE, " +
                    "password varchar(50) NOT NULL)";
stmt.execute(sql);
            String sql1="CREATE TABLE IF (" +
            "    booking_id INT AUTO_INCREMENT PRIMARY KEY," +
               "    movie_nameVARCHAR(100)," +
           "    book_dateVarchar(50)," +
        "    book_timevarchar(50)," +
           "    seats VARCHAR(50)," +
               "    customer_id INT)"
;stmt.execute(sql1);
conn.close();
        } catch (SQLException e) {
e.printStackTrace();
        }
    }

public  static void main(String[] args) {
initializeDatabase();
SwingUtilities.invokeLater(() ->createAndShowLogin());
```

```
    }
private  staticJButton[] seatButtons;
 private static JPanelseatPanel;
private  static void MovieSeatBooking(int
id,Stringmovie_name,Stringbook_date,Stringbook_time,JFrameframe,JPa
nel panel) {          frame.remove(panel);
 frame.revalidate();
frame.repaint();
JPanel panel2=new JPanel();

panel2.setLayout(new BoxLayout(panel2, BoxLayout.Y_AXIS));
panel2.setBackground(Color.cyan);

 seatPanel = new JPanel();

 seatPanel.setLayout(new GridLayout(5, 5, 10, 10));  // 5x5 grid for seats
seatPanel.setBackground(Color.cyan);

        // Initialize seat buttons
seatButtons = new JButton[25];
  for (int i = 0; i<seatButtons.length; i++) {
seatButtons[i] = new JButton("Seat " + (i + 1));
seatButtons[i].setBackground(Color.GREEN);  // Default color for
available seats
seatPanel.add(seatButtons[i]);
```

```
            // Add action listener for seat selection              int
seatIndex = i;  // Final variable for use in lambda
seatButtons[i].addActionListener(e -> {
JButtonclickedButton = (JButton) e.getSource();
 if                  (clickedButton.getBackground()==Color.GREEN)                  {
clickedButton.setBackground(Color.YELLOW);       //    Mark    as    selected
}    else    if    (clickedButton.getBackground()    ==    Color.YELLOW)    {
clickedButton.setBackground(Color.GREEN);  // Deselect
                }
            });
        }

        // Fetch booked seats from the database
        List<Integer>bookedSeats =
fetchBookedSeats(movie_name,book_date,book_time);
        for (int seatIndex :bookedSeats) {
seatButtons[seatIndex - 1].setBackground(Color.RED);  // Mark as booked
seatButtons[seatIndex - 1].setEnabled(false);  // Disable button
        }

        // Add confirm booking button
JButtonconfirmButton = new JButton("Confirm Booking");
confirmButton.setAlignmentX(Component.CENTER_ALIGNMENT);
confirmButton.addActionListener(e ->
confirmBooking(movie_name,book_date,book_time,id,panel,panel2,frame));
```

```
        panel2.add(seatPanel);
panel2.add(confirmButton);
frame.add(panel2);
frame.setVisible(true);
    }


    // Static method to fetch booked seats from the database
    private static List<Integer>fetchBookedSeats(String
movie_name,Stringbook_date,Stringbook_time) {
        List<Integer>bookedSeats = new ArrayList<>();


        String query = "SELECT seats FROM seat_bookings WHERE movie_name
=
? AND book_date= ? AND book_time= ?";
        try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/db","root","Sai@
30.");
PreparedStatementpstmt = conn.prepareStatement(query)) {


pstmt.setString(1, movie_name);
pstmt.setString(2, book_date);
 pstmt.setString(3, book_time);


ResultSetrs = pstmt.executeQuery();
 while (rs.next()) {
```

```
String[] seats = rs.getString("seats").split(",");  // Parse booked seats
for (String seat : seats) {
bookedSeats.add(Integer.parseInt(seat.replaceAll("\\D", "")));  // Extract seat
number
            }
         }
conn.close();
      } catch (SQLException e) {
e.printStackTrace();
      }
      return bookedSeats;
   }


   // Static method to confirm the booking
   private static void confirmBooking(String movie_name,String
book_date,Stringbook_time,intid,JPanel panel1,JPanel panel2,JFrame frame) {
      List<String>selectedSeats = new ArrayList<>();
      for (int i = 0; i<seatButtons.length; i++) {          if
(seatButtons[i].getBackground() == Color.YELLOW) {  // Selected seats
selectedSeats.add("Seat " + (i + 1));
         }
      }


      if (!selectedSeats.isEmpty()) {
         String bookedSeatsStr = String.join(",", selectedSeats); // Convert list of
seats to a comma-separated string
```

```
        String currentDate = book_date;
        String currentTime = book_time;


        // MySQL connection and insert query
            String query = "INSERT INTO seat_bookings (movie_name,
customer_id, seats, book_date, book_time) VALUES (?, ?, ?, ?, ?)";
            try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/db","root","Sai@
30.");
PreparedStatementpreparedStatement                         =
conn.prepareStatement(query))                              {
preparedStatement.setString(1,movie_name);
preparedStatement.setInt(2,id);
preparedStatement.setString(3,bookedSeatsStr);
preparedStatement.setString(4,currentDate);
preparedStatement.setString(5, currentTime);

                int rowsAffected = preparedStatement.executeUpdate();
                if (rowsAffected> 0) {
JOptionPane.showMessageDialog(null, "Booking successful!");
frame.remove(panel2);
 frame.revalidate();
  frame.repaint();
frame.add(panel1);
                }
               } catch (SQLException e) {
```

```
JOptionPane.showMessageDialog(null, "Error booking seats: " +
e.getMessage());

e.printStackTrace();

      }

     // Add code here to update the database with new bookings.

       // Add code here to update the database with new bookings.

      } else {

JOptionPane.showMessageDialog(null, "No seats selected!");

      }

   }


   private static void createAndShowLogin() {

      // Create the main frame

JFrame frame = new JFrame("Movie Ticket booking");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

frame.setExtendedState(JFrame.MAXIMIZED_BOTH);

frame.setSize(400, 350);

 frame.getContentPane().setBackground(Color.red);


frame.setLayout(new GridBagLayout()); // Center content in the frame


      // Create a panel to hold the components

JPanel panel = new JPanel();

 panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20)); // Add
padding
panel.setBackground(Color.cyan);        // Add components to the panel
JLabeltitleLabel = new JLabel("Login");
 titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
panel.add(titleLabel);   panel.add(Box.createRigidArea(new
Dimension(0, 20))); // Spacer

JLabelusernameLabel = new JLabel("Username:");
usernameLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
panel.add(usernameLabel);

JTextFieldusernameField = new JTextField(15);
panel.add(usernameField);   panel.add(Box.createRigidArea(new
Dimension(0, 10))); // Spacer

JLabelpasswordLabel = new JLabel("Password:");
passwordLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
panel.add(passwordLabel);

JPasswordFieldpasswordField = new JPasswordField(15);
panel.add(passwordField);
  panel.add(Box.createRigidArea(new Dimension(0, 20))); // Spacer
```

```
        // Create a panel for buttons
 JPanelbuttonPanel = new JPanel();
   buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 0));


JButtonloginButton = new JButton("Login");
JButtonsignUpButton = new JButton("Sign Up");
loginButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
loginUser(usernameField.getText(), new
String(passwordField.getPassword()),frame,panel);
                    }
        });
signUpButton.addActionListener(e ->createAndShowSignUp());
buttonPanel.add(loginButton);
buttonPanel.add(signUpButton);


        // Add button panel to the main panel
panel.add(buttonPanel);


        // Add the panel to the frame
frame.add(panel);


        // Center the frame on screen and make it visible
frame.setLocationRelativeTo(null);
  frame.setVisible(true);
```

```
    }


    private static void createAndShowSignUp() {
JFrame frame = new JFrame("Sign Up Page");
frame.setSize(400, 200);


JPanel panel = new JPanel();
 panel.setLayout(new GridLayout(3, 2, 5, 5));


JTextFieldusernameField = new JTextField();
JPasswordFieldpasswordField = new JPasswordField();
 panel.add(new JLabel("Username:"));
 panel.add(usernameField);
 panel.add(new JLabel("Password:"));
  panel.add(passwordField);


JButtonsubmitButton = new JButton("Submit");
submitButton.addActionListener(e ->signUpUser(usernameField.getText(), new
String(passwordField.getPassword())));


panel.add(submitButton);


frame.add(panel);
frame.setLocationRelativeTo(null);
frame.setVisible(true);
```

```
    }


    private static void loginUser(String username, String
password,JFrameframe,JPanel panel) {


        String query = "SELECT * FROM users WHERE username = ? AND
password
= ?";

        try (Connection conn = connect(); PreparedStatementpstmt =
conn.prepareStatement(query)) {             pstmt.setString(1,
username);             pstmt.setString(2, password);
ResultSetrs = pstmt.executeQuery();
            if(rs.next()){
int id=rs.getInt("id");
frame.remove(panel);
frame.revalidate();
frame.repaint();
JPanel panel1=new JPanel(new FlowLayout());
        // Create a panel for the search section
JPanelsearchPanel = new JPanel();
panel1.setBackground(Color.cyan);
  searchPanel.setLayout(new BoxLayout(searchPanel, BoxLayout.Y_AXIS));
searchPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
        // Create and add components to the search panel
JLabelmovieLabel = new JLabel("Movie Name:");
JTextFieldmovieField = new JTextField(20);
```

```
        // Create a JComboBox to hold the dates
JComboBox<String>dateComboBox = new JComboBox<>();

        // Get today's date and the next 4 dates
        Calendar calendar = Calendar.getInstance();
SimpleDateFormatdateFormat = new SimpleDateFormat("MMM dd, yyyy");

        // Add the next 5 dates to the combo box
        for (int i = 0; i< 5; i++) {
            String formattedDate = dateFormat.format(calendar.getTime());
dateComboBox.addItem(formattedDate);
calendar.add(Calendar.DAY_OF_MONTH, 1);  // Move to the next day
        }
JLabeltimeLabel = new JLabel("Time:");
JLabelldate = new JLabel("Date:");
       // Define the time slots in AM/PM format
        // Define the available time slots
String[] timeSlots = {"9:00 AM", "1:00 PM", "5:00 PM", "10:00 PM"};

        // Get current time
LocalTimecurrentTime = LocalTime.now();

        // List to store valid time slots
        List<String>validTimeSlots = new ArrayList<>();
        // Convert time slots to LocalTime and compare with current time
for (String timeSlot :timeSlots) {
            // Parse the time slots and convert them to LocalTime objects
LocalTime time = parseTimeSlot(timeSlot);

            // Only add the time to the list if it's after the current time
if (time.isAfter(currentTime)) {
validTimeSlots.add(timeSlot);
            }
        }

        // Create a JComboBox with the valid time slots
JComboBox<String>timeField = new
JComboBox<>(validTimeSlots.toArray(new String[0]));



JButtonsearchButton = new JButton("Search");


searchPanel.add(movieLabel);
 searchPanel.add(movieField);
searchPanel.add(Box.createRigidArea(new Dimension(0, 20)));
searchPanel.add(ldate);
```

```
 searchPanel.add(dateComboBox);
searchPanel.add(Box.createRigidArea(new Dimension(0, 20)));
searchPanel.add(timeLabel);
searchPanel.add(timeField);
searchPanel.add(Box.createRigidArea(new Dimension(0, 20)));
searchPanel.add(searchButton);
searchPanel.add(Box.createRigidArea(new Dimension(0, 20)));
        // Add an area for suggestions (using a JList for example)
DefaultListModel<String>suggestionsModel = new DefaultListModel<>();
JList<String>suggestionsList = new JList<>(suggestionsModel);
JScrollPanesuggestionsScrollPane = new JScrollPane(suggestionsList);


        // Sample movie suggestions
String[] sampleMovies = {"Inception", "The Matrix", "Titanic", "Avatar"};
for (String movie : sampleMovies) {
suggestionsModel.addElement(movie);
        }


        // Add listener for movie field to show suggestions
movieField.getDocument().addDocumentListener(new DocumentListener() {
            @Override
  public void insertUpdate(DocumentEvent e) {
updateSuggestions();
            }
            @Override
public void removeUpdate(DocumentEvent e) {
updateSuggestions();
            }

            @Override
 public void changedUpdate(DocumentEvent e) {
updateSuggestions();
            }

            private void updateSuggestions() {
                String input =
movieField.getText().toLowerCase();
suggestionsModel.clear();
for (String movie : sampleMovies) {                    if
(movie.toLowerCase().contains(input)) {
suggestionsModel.addElement(movie);
                    }
                }
            }
        });
```

}

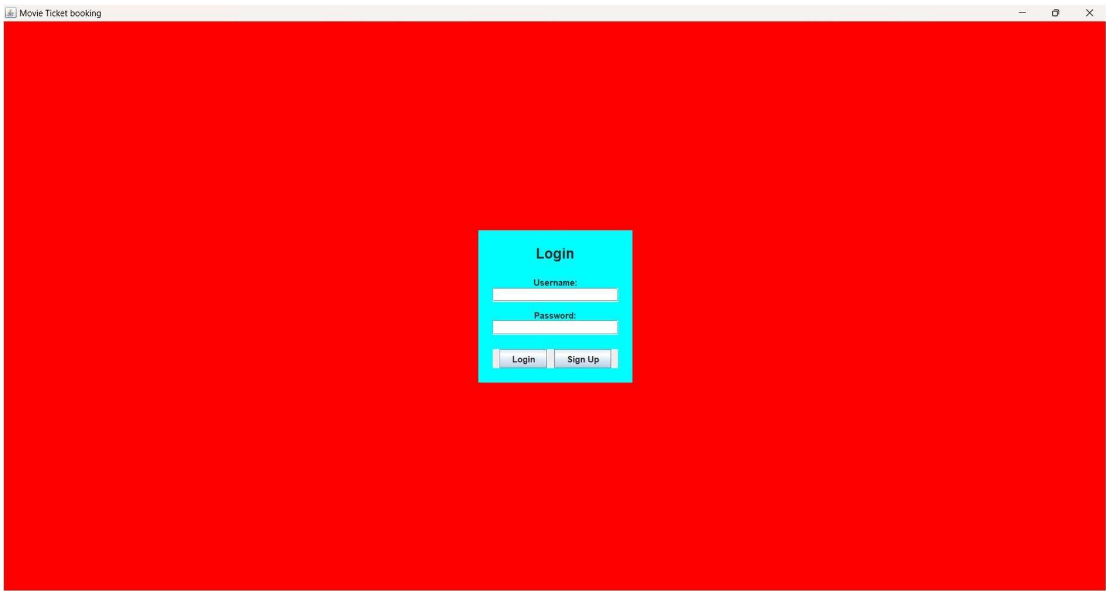
```
        // Add action to search button
    searchButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
                String movieName = movieField.getText();
                String date = (String)dateComboBox.getSelectedItem();
                String time = (String)timeField.getSelectedItem();

                if (movieName.isEmpty() || date.isEmpty() || time.isEmpty()) {
    JOptionPane.showMessageDialog(frame, "Please fill out all fields.",
    "Input Error", JOptionPane.ERROR_MESSAGE);
                } else {
    MovieSeatBooking(id,movieName,date,time,frame,panel1);
                }
            }
        });
    searchPanel.setBackground(Color.cyan);        //
    Add components to the frame
    panel1.add(searchPanel);
    panel1.add(suggestionsScrollPane);
    frame.add(panel1);

        // Make the frame visible
    frame.setVisible(true);
    }
    else{
```

```
JOptionPane.showMessageDialog(null,"Invalid username or
password","Error",JOptionPane.ERROR_MESSAGE);

          }

   }

   catch (SQLException e) {

e.printStackTrace();

       }

   }


   private static void signUpUser(String username, String password) {
String query = "INSERT INTO users(username, password) VALUES(?, ?)";

       try (Connection conn = connect(); PreparedStatementpstmt =

conn.prepareStatement(query)) {            pstmt.setString(1,

username);

 pstmt.setString(2, password);

 pstmt.executeUpdate();

JOptionPane.showMessageDialog(null, "Sign Up Successful!");

       } catch (SQLException e) {

  if (e.getErrorCode() == 19) {  // SQLite constraint violation for UNIQUE

JOptionPane.showMessageDialog(null, "Username already taken", "Error",
JOptionPane.ERROR_MESSAGE);

          } else {

e.printStackTrace();

          }

       }

   }
```

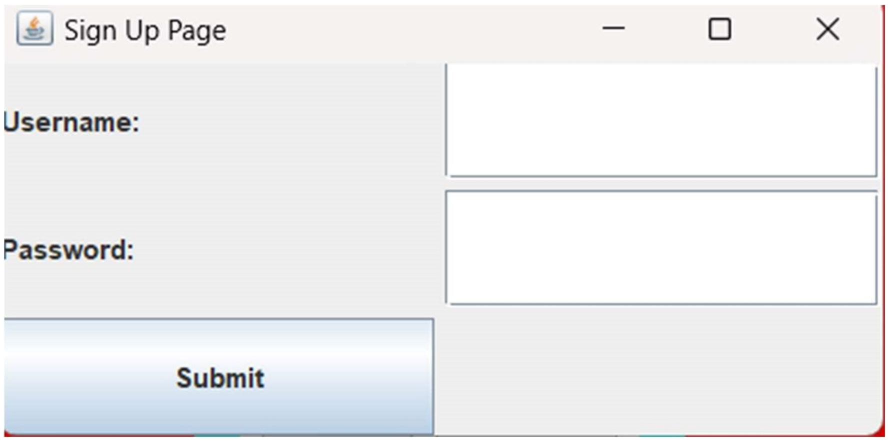
```
    private static LocalTimeparseTimeSlot(String timeSlot) {
switch (timeSlot) {              case "9:00 AM":
                return LocalTime.of(9, 0);
case "1:00 PM":
                return LocalTime.of(13, 0);
case "5:00 PM":
                return LocalTime.of(17, 0);
case "10:00 PM":

                return LocalTime.of(22, 0);
default:
                throw new IllegalArgumentException("Invalid time slot");
        }
}
}
```

SNAPSHOT

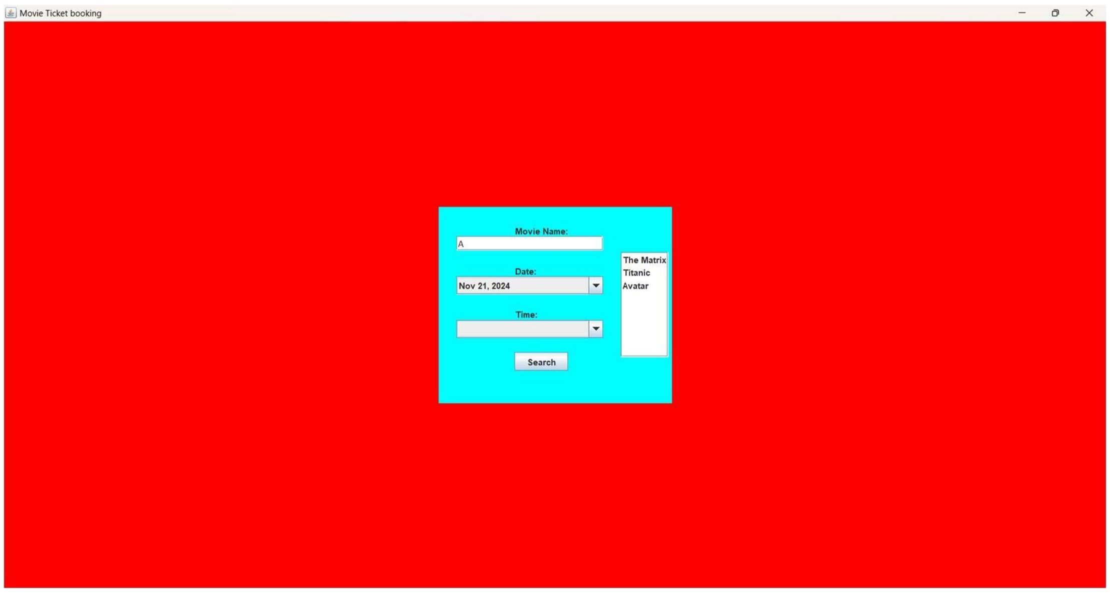
1)LOGIN PAGE



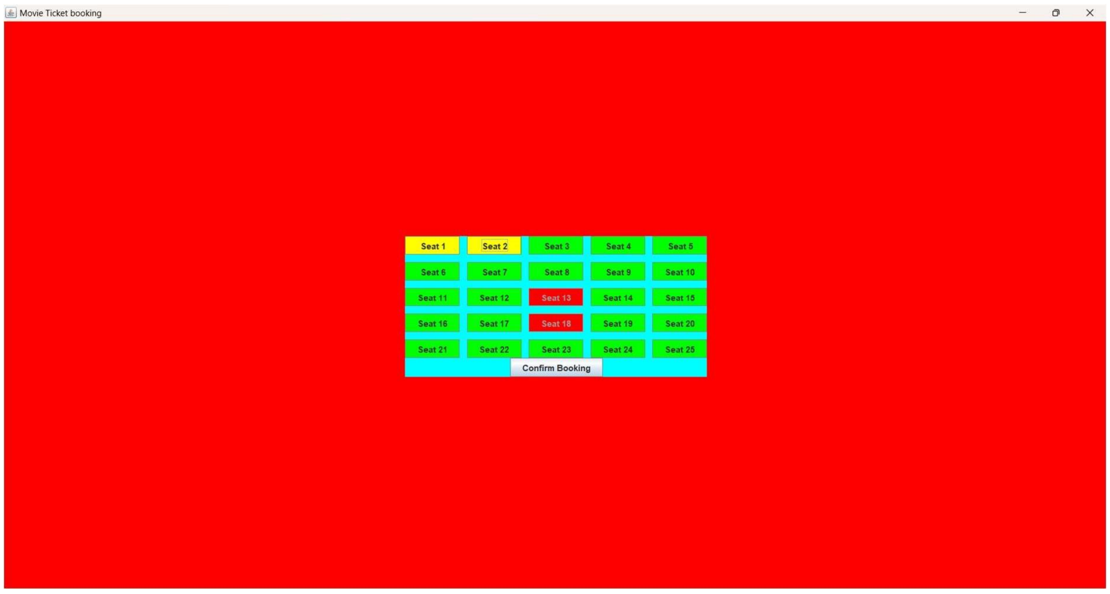
2)SIGN UP PAGE



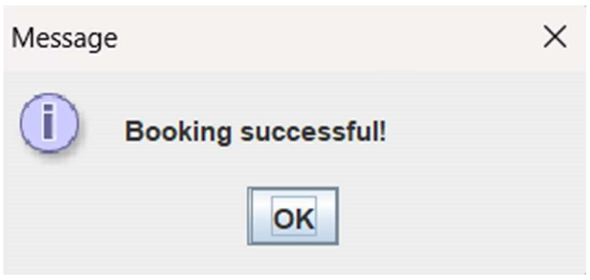
3)MAIN PAGE



4)SEAT BOOKING PAGE



5)BOOKING CONFIRMATION


Message

Booking successful!

CONCLUSION

The Movie Ticket Booking System built using Java Swing and JDBC provides an efficient, user-friendly platform for managing movie ticket reservations. By integrating a visually appealing GUI with robust backend functionalities, the system simplifies the ticket booking process for both customers and theatre administrators.

For users, the system offers seamless navigation through movie schedules, easy seat selection, and secure booking with real-time updates. Admin users are empowered with the ability to manage movie schedules, track bookings, and generate reports, contributing to smooth operations and informed decision-making.

Future Scope and Enhancements ● Multi-User and

Multi-Theatre Support:

- ○ Expansion to support multiple theatres or branches with separate movie schedules and bookings.
- ● Integration with External APIs:

    - Integrate with external services such as movie databases (e.g., IMDb) for up-to-date movie information.
- Mobile and Web Application:
    - Porting the system to web or mobile platforms using JavaFX, Spring Boot, or Android to extend reach and usability.
- Advanced Features:
    - AI-driven recommendations based on user preferences and past bookings.
    - Dynamic pricing based on seat location, movie popularity, or time.

REFERENCE

- www.google.com
- https://chat.openai.com/
- https://chatuml.com/