MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

IA-3 OOP-II [MCA 4221]

MOVIE TICKET BOOKING SYSTEM

MINI PROJECT

SUBMITTED BY: MOHAMMED WAHEED

REG NO: 240970144

CLASS: I MCA B

SUBMITTED TO: MR S SHAMEEM SS

DEPARTMENT OF DATA SCIENCE AND COMPUTER APPLICATIONS, MIT MANIPAL

Overview:

The Movie Ticket Booking System is a GUI-based desktop application developed in Java using Swing. It facilitates users in booking movie tickets seamlessly through a structured flow involving login, movie selection, seat reservation, and payment. The application interacts with a MySQL database to store and retrieve data related to users, movies, bookings, and payments, ensuring persistence and real-time validation of booking information.

Problem Statement:

The manual process of movie ticket booking at theaters is time-consuming, error-prone, and lacks real-time seat availability checks. This project aims to automate the ticket booking process by developing a user-friendly software solution that allows users to log in, browse available movies, select showtimes, reserve seats, and complete payments, all in a single digital interface. The system should ensure data consistency, seat availability, and provide clear user feedback.

Requirements:

1. Login Page:

- The application should launch with a login window.
- Users must authenticate with valid credentials stored in the database.
- Upon successful login, users are redirected to the movie selection page.

2. Movie Selection and Seat Booking:

- Display a list of movies with posters, descriptions, showtimes, and theaters.
- Allow users to select a movie and showtime.
- Present a graphical seat layout indicating:
    - **Green** for available seats
    - **Red** for already booked seats
    - **Yellow** for currently selected seats
- Prevent users from selecting already booked seats.
- Prompt users to select at least one seat before proceeding.

3. Payment and Confirmation:

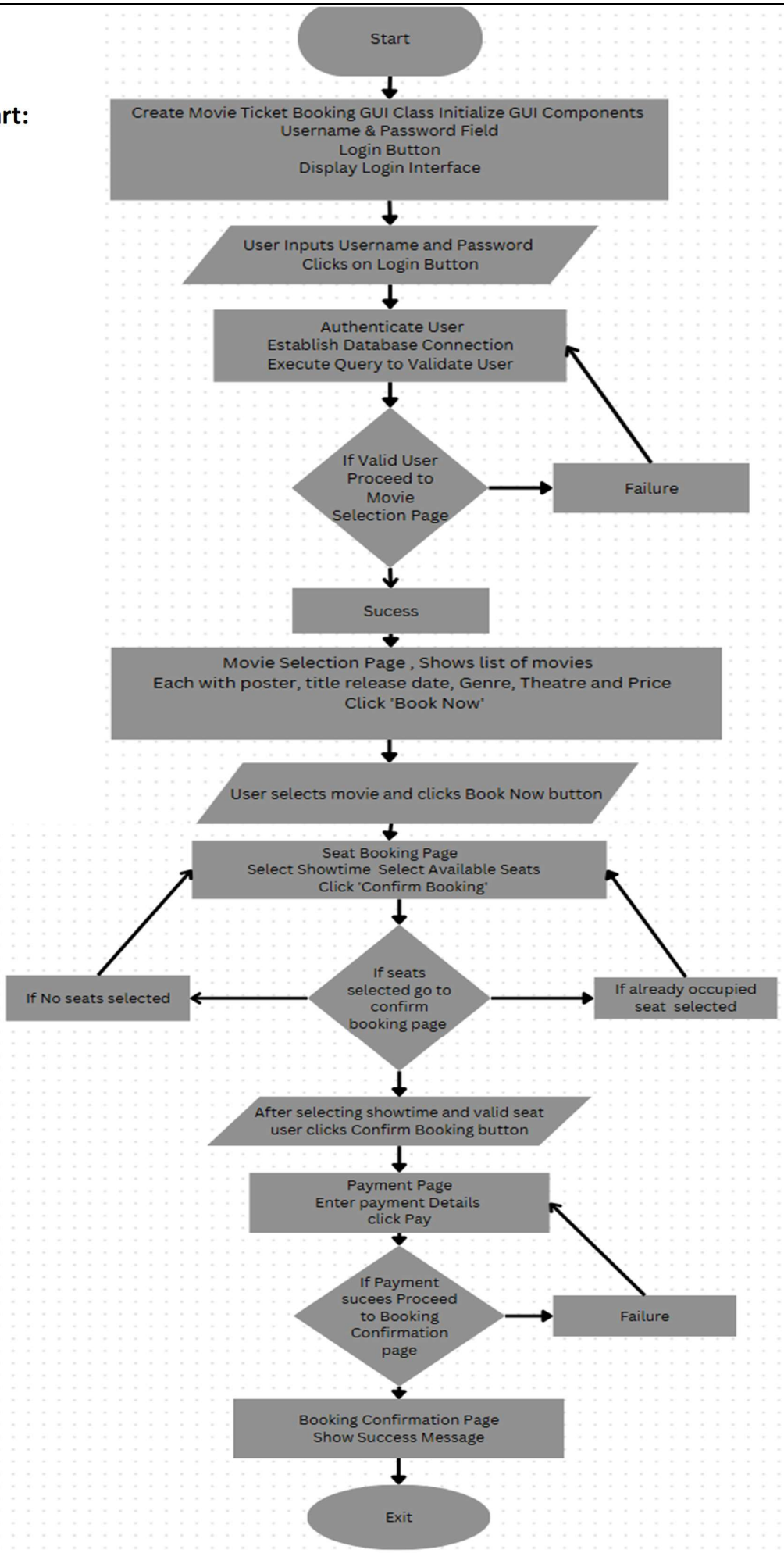
- After seat selection, prompt users to confirm and proceed with payment.
- Upon successful payment:

- Display a confirmation message with movie name, showtime, seats, and a success message.
- Store payment and booking details in the database.

Objectives:

- To develop a user authentication system that restricts access to authorized users.
- To design an intuitive movie selection interface displaying movie details dynamically.
- To implement a visual seat booking module with real-time availability checking.
- To ensure error messages and validations (e.g., seat already booked, no seat selected) are clear and user-friendly.
- To integrate a simple payment simulation interface.
- To store and manage all movie, user, booking, and payment information using MySQL.
- To improve the ticket booking experience by minimizing manual intervention and wait time.

Flowchart:


Start
Create Movie Ticket Booking GUI Class Initialize GUI Components
Username & Password Field
Login Button
Display Login Interface
User Inputs Username and Password
Clicks on Login Button
Authenticate User
Establish Database Connection
Execute Query to Validate User
If Valid User
Proceed to
Movie
Selection Page
Failure
Sucess
Movie Selection Page , Shows list of movies
Each with poster, title release date, Genre, Theatre and Price
Click 'Book Now'
User selects movie and clicks Book Now button
Seat Booking Page
Select Showtime Select Available Seats
Click 'Confirm Booking'
If seats
selected go to
confirm
booking page
If No seats selected
If already occupied
seat selected
After selecting showtime and valid seat
user clicks Confirm Booking button
Payment Page
Enter payment Details
click Pay
If Payment
sucees Proceed
to Booking
Confirmation
page
Failure
Booking Confirmation Page
Show Success Message
Exit

Swing Components

In the Movie Ticket Booking System project, the following Swing components were used:

- **JFrame**: Acts as the main window for each major part of the application — Login Page, Movie Selection Page, Seat Booking Page, and Payment Page.
- **JPanel**: Used extensively to organize layouts within the frames. It structures form elements, buttons, seat grids, and other components in a clean layout.
- **JLabel**: Displays text on the interface such as titles ("Welcome to Movie App"), form labels ("Username", "Password"), and messages ("Invalid Credentials", "Login Successful").
- **JTextField**: Used to input the username in the login form.
- **JButton**: Interactive buttons used for actions such as "Login", "Book Now", "Confirm Booking", and "Pay Now".
- **JComboBox**: Provides a dropdown for selecting showtimes like "10:00 AM", "1:00 PM", etc.
- **JOptionPane**: Used for pop-up messages to show alerts like "Invalid Credentials", "Seat already booked!", or booking confirmation messages.
- **BorderLayout**: Used for placing panels in the main areas of a frame:(NORTH, SOUTH, EAST, WEST, and CENTER)
- **GridLayout:** Used for arranging seat buttons in a uniform grid structure (4row *4col).

Event Handling:

The system is event-driven, meaning actions are triggered based on user interactions:

- **Login Button Click**: Triggers authentication logic. If credentials match, it navigates to the movie selection page; otherwise, shows an error.
- **"Book Now" Button**: On movie cards, clicking this button opens the SeatBookingPage with the selected movie's data.
- **Seat Button Click**: Each seat button responds to clicks. If it's already booked, a message is shown. If selected, it changes color to indicate selection; clicking again deselects it.
- **Showtime Selection (ComboBox)**: When the user selects a showtime, an event is triggered to reload the seat availability for that specific time.
- **"Confirm Booking" Button**: Validates that at least one seat is selected. If validation passes, it proceeds to the payment window.
- **"Pay Now" Button**: Finalizes the booking after payment and shows a confirmation message.

Well Commented Source Code:

LoginPage.java

```
import java.awt.*;
import java.sql.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;
public class LoginPage extends JFrame {

   // Components for login form
   JTextField userField;
   JPasswordField passField;
   JButton loginButton;
   JLabel statusLabel;

   // Database connection details
   final String DB_URL = "jdbc:mysql://localhost:3306/testdb";
   final String DB_USER = "root";
   final String DB_PASS = "";

   public LoginPage() {
      // Set window title and size
      setTitle("User Login");
      setSize(400, 350);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLocationRelativeTo(null); // Center the window on screen

      // Wrapper panel with vertical layout
      JPanel wrapperPanel = new JPanel();
      wrapperPanel.setLayout(new BoxLayout(wrapperPanel, BoxLayout.Y_AXIS));
```

```
wrapperPanel.setBackground(Color.WHITE);
wrapperPanel.setBorder(BorderFactory.createEmptyBorder(40, 30, 30, 30));

// Form panel using GridBagLayout for alignment
JPanel formPanel = new JPanel(new GridBagLayout());
formPanel.setBorder(new TitledBorder("Login Form"));
formPanel.setBackground(Color.WHITE);

// GridBag constraints for placing components
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(10, 10, 10, 10);
gbc.fill = GridBagConstraints.HORIZONTAL;

// Username label and text field
JLabel userLabel = new JLabel("Username:");
userLabel.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 0; gbc.gridy = 0;
formPanel.add(userLabel, gbc);

userField = new JTextField(20);
gbc.gridx = 1; gbc.gridy = 0;
formPanel.add(userField, gbc);

// Password label and password field
JLabel passLabel = new JLabel("Password:");
passLabel.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 0; gbc.gridy = 1;
formPanel.add(passLabel, gbc);
passField = new JPasswordField(20);
```

```
        gbc.gridx = 1; gbc.gridy = 1;
        formPanel.add(passField, gbc);

        // Login button configuration
        loginButton = new JButton("Login");
        loginButton.setAlignmentX(Component.CENTER_ALIGNMENT);
        loginButton.setBackground(new Color(59, 89, 182));
        loginButton.setForeground(Color.WHITE);
        loginButton.setFocusPainted(false);
        loginButton.setFont(new Font("Tahoma", Font.BOLD, 12));

        // Label for displaying login status messages
        statusLabel = new JLabel("", SwingConstants.CENTER);
        statusLabel.setForeground(Color.RED);
        statusLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Adding form panel and other components to wrapper panel
        wrapperPanel.add(formPanel);
        wrapperPanel.add(Box.createVerticalStrut(20));
        wrapperPanel.add(loginButton);
        wrapperPanel.add(Box.createVerticalStrut(10));
        wrapperPanel.add(statusLabel);

        // Add wrapper to main frame
        add(wrapperPanel, BorderLayout.CENTER);

        // Action listener for login button
        loginButton.addActionListener(e -> authenticate());
    }
```

```
    // Function to validate user credentials from database
    void authenticate() {
        String username = userField.getText();
        String password = new String(passField.getPassword());

        try {
            // Load JDBC driver and establish database connection
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);

            // Prepare SQL query to check credentials
            String sql = "SELECT * FROM users WHERE username=? AND password=?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, username);
            stmt.setString(2, password);

            // Execute query and process results
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                // Login successful - extract user ID
                int userId = rs.getInt("id");

                // Display success message and open movie selection window
                statusLabel.setForeground(new Color(0, 128, 0));
                statusLabel.setText("Login Successful!");
                JOptionPane.showMessageDialog(this, "Welcome, " + username + "!", "Success",
JOptionPane.INFORMATION_MESSAGE);

                // Open MovieSelectionPage and close login window
```

```
            SwingUtilities.invokeLater(() -> new MovieSelectionPage(username,
userId).setVisible(true));
            dispose();
          } else {
            // Invalid credentials entered
            statusLabel.setForeground(Color.RED);
            statusLabel.setText("Invalid Credentials");
          }
          // Close all resources
          rs.close();
          stmt.close();
          conn.close();
        } catch (Exception ex) {
          ex.printStackTrace();
          statusLabel.setText("Database Error");
        }
    }
    // Main method to launch login window
    public static void main(String[] args) {
      SwingUtilities.invokeLater(() -> new LoginPage().setVisible(true));
    }
}
```

MovieSelectionPage.java

```
import java.awt.*;
import java.sql.*;
import java.util.ArrayList;
import javax.swing.*;
public class MovieSelectionPage extends JFrame {
   // Inner class to hold movie details
   class Movie {
      int id;
      String title, releaseDate, genre, theatre, imagePath;

      Movie(int id, String title, String releaseDate, String genre, String theatre, String imagePath) {
         this.id = id;
         this.title = title;
         this.releaseDate = releaseDate;
         this.genre = genre;
         this.theatre = theatre;
         this.imagePath = imagePath;
      }
   }
   // List to store all movie objects
   ArrayList<Movie> movieList = new ArrayList<>();

   // Database credentials
   final String DB_URL = "jdbc:mysql://localhost:3306/testdb";
   final String DB_USER = "root";
   final String DB_PASS = "";
```

```
// Logged-in user information
int userId;
String username;

// Constructor to initialize the movie selection page
public MovieSelectionPage(String username, int userId) {
    this.username = username;
    this.userId = userId;

    // Set up the frame properties
    setTitle("Welcome " + username + " - Movie Booking App");
    setSize(900, 600);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    // Create and add a heading label at the top
    JLabel header = new JLabel("Welcome To Movies App", SwingConstants.CENTER);
    header.setFont(new Font("Segoe UI", Font.BOLD, 24));
    header.setForeground(new Color(0, 120, 215));
    header.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));
    add(header, BorderLayout.NORTH);

    // Load movie data from database
    loadMoviesFromDB();
    // Create grid layout to display movies in card format
    JPanel movieGrid = new JPanel(new GridLayout(0, 3, 15, 15));
    for (Movie movie : movieList) {
        movieGrid.add(createMovieCard(movie));
```

```
    }

    // Add scrollable panel to hold all movie cards
    JScrollPane scrollPane = new JScrollPane(movieGrid);
    add(scrollPane, BorderLayout.CENTER);
}

// Fetch movie records from database and store in list
private void loadMoviesFromDB() {
    try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS)) {
        ResultSet rs = conn.createStatement().executeQuery("SELECT * FROM movies");
        while (rs.next()) {
            movieList.add(new Movie(
                rs.getInt("id"),
                rs.getString("title"),
                rs.getString("release_date"),
                rs.getString("genre"),
                rs.getString("theatre"),
                rs.getString("image_path")
            ));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
// Create a card component for each movie
private JPanel createMovieCard(Movie movie) {
    JPanel card = new JPanel();
    card.setLayout(new BoxLayout(card, BoxLayout.Y_AXIS));
```

```
        card.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        card.setBackground(Color.WHITE);

        // Load and scale the image to fit in the card
        ImageIcon icon = new ImageIcon(movie.imagePath);
        Image scaledImage = icon.getImage().getScaledInstance(250, 250,
Image.SCALE_SMOOTH);
        JLabel imageLabel = new JLabel(new ImageIcon(scaledImage));
        imageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        card.add(imageLabel);

        // Create and populate the panel with movie info
        JPanel infoPanel = new JPanel();
        infoPanel.setLayout(new BoxLayout(infoPanel, BoxLayout.Y_AXIS));
        infoPanel.setBackground(Color.WHITE);
        infoPanel.setAlignmentX(Component.CENTER_ALIGNMENT);
        infoPanel.setBorder(BorderFactory.createEmptyBorder(10, 0, 10, 0));

        infoPanel.add(new JLabel("Title: " + movie.title));
        infoPanel.add(new JLabel("Release: " + movie.releaseDate));
        infoPanel.add(new JLabel("Genre: " + movie.genre));
        infoPanel.add(new JLabel("Theatre: " + movie.theatre));

        // Price label
        JLabel priceLabel = new JLabel("Price: ₹250");
        priceLabel.setForeground(new Color(0, 128, 0));
        infoPanel.add(priceLabel);

        // "Book Now" button with event to proceed to seat booking
        JButton bookButton = new JButton("Book Now");
```

```
        bookButton.setAlignmentX(Component.CENTER_ALIGNMENT);
        bookButton.setBackground(new Color(0, 120, 215));
        bookButton.setForeground(Color.WHITE);
        bookButton.setFocusPainted(false);
        bookButton.setFont(new Font("Arial", Font.BOLD, 12));
        bookButton.addActionListener(e -> {
            new SeatBookingPage(movie.id, movie.title, userId).setVisible(true);
            dispose();
        });
        infoPanel.add(Box.createRigidArea(new Dimension(0, 10)));
        infoPanel.add(bookButton);
        // Add the info panel below the image in the card
        card.add(infoPanel);
        return card;
    }
}
```

SeatBookingPage.java

```
import java.awt.*;
import java.sql.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
public class SeatBookingPage extends JFrame {

    // Stores booked status of 20 seats (true = booked)
    boolean[] seats = new boolean[20];
    // Movie and user details
    String movieTitle;
    int movieId;
    int userId;

    // Stores user-selected seats
    List<Integer> selectedSeats = new ArrayList<>();
    // References to seat buttons for dynamic updates
    JButton[] seatButtons = new JButton[20];
    // Dropdown to select showtime
    JComboBox<String> showtimeComboBox;

    // Database credentials
    final String DB_URL = "jdbc:mysql://localhost:3306/testdb";
    final String DB_USER = "root";
    final String DB_PASS = "";
```

```
    // Constructor to initialize the booking UI
    public SeatBookingPage(int movieId, String movieTitle, int userId) {
        this.movieId = movieId;
        this.movieTitle = movieTitle;
        this.userId = userId;

        // Frame settings
        setTitle("Book Seats - " + movieTitle);
        setSize(450, 400);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(new BorderLayout());

        // Top panel with title and showtime selection
        JPanel topPanel = new JPanel();
        topPanel.setLayout(new BoxLayout(topPanel, BoxLayout.Y_AXIS));
        JLabel titleLabel = new JLabel("Select Showtime and Seats", SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 18));
        titleLabel.setForeground(new Color(30, 144, 255));
        titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        titleLabel.setBorder(new EmptyBorder(10, 0, 10, 0));
        topPanel.add(titleLabel);

        // Showtime combo box
        JPanel comboPanel = new JPanel();
        comboPanel.add(new JLabel("Showtime:"));
        showtimeComboBox = new JComboBox<>(new String[]{"10:00 AM", "1:00 PM", "4:00
PM", "7:00 PM"});
        showtimeComboBox.addActionListener(e -> loadBookedSeats());
        comboPanel.add(showtimeComboBox);
```

```
topPanel.add(comboPanel);

add(topPanel, BorderLayout.NORTH);
// Seat layout grid (4x5)
JPanel seatPanel = new JPanel(new GridLayout(4, 5, 10, 10));
for (int i = 0; i < 20; i++) {
   JButton btn = new JButton("Seat " + (i + 1));
   int index = i;
   seatButtons[i] = btn;
   btn.setBackground(Color.GREEN); // Default: available

   // Button click to toggle selection
   btn.addActionListener(e -> {
      if (seats[index]) {
         JOptionPane.showMessageDialog(this, "Seat already booked!");
         return;
      }
      if (selectedSeats.contains(index)) {
         selectedSeats.remove((Integer) index);
         btn.setBackground(Color.GREEN);
      } else {
         selectedSeats.add(index);
         btn.setBackground(Color.YELLOW);
      }
   });

   seatPanel.add(btn);
}
add(seatPanel, BorderLayout.CENTER);
```

```
        // Confirm Booking button
        JButton confirmButton = new JButton("Confirm Booking");
        confirmButton.setFont(new Font("Arial", Font.BOLD, 14));
        confirmButton.setBackground(new Color(0, 120, 215));
        confirmButton.setForeground(Color.WHITE);
        confirmButton.setFocusPainted(false);
        confirmButton.setPreferredSize(new Dimension(180, 35));

        // Confirmation action
        confirmButton.addActionListener(e -> {
            if (selectedSeats.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please select at least one seat!");
                return;
            }
            String showtime = (String) showtimeComboBox.getSelectedItem();
            new Payment(this, movieId, movieTitle, selectedSeats, showtime,
userId).setVisible(true);

            // Reset selection after payment
            selectedSeats.clear();
            loadBookedSeats();
        });
        JPanel bottom = new JPanel();
        bottom.add(confirmButton);
        add(bottom, BorderLayout.SOUTH);

        // Initial load of booked seats
        loadBookedSeats();
    }
```

```
// Loads booked seats for selected movie and showtime
private void loadBookedSeats() {
    Arrays.fill(seats, false);
    String showtime = (String) showtimeComboBox.getSelectedItem();

    try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS)) {
        PreparedStatement stmt = conn.prepareStatement(
            "SELECT seat_number FROM bookings WHERE movie_id = ? AND showtime = ?"
        );
        stmt.setInt(1, movieId);
        stmt.setString(2, showtime);
        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            int s = rs.getInt("seat_number");
            if (s >= 1 && s <= 20) {
                seats[s - 1] = true;
             }
    } catch (Exception e) {
        e.printStackTrace();
    }
    // Update seat button colors based on booking status
    for (int i = 0; i < seatButtons.length; i++) {
        if (seats[i]) {
            seatButtons[i].setBackground(Color.RED); // Booked
        } else
            seatButtons[i].setBackground(Color.GREEN); // Available
        }
    }
}
```

Payment.java

```
import java.awt.*;
import java.sql.*;
import java.util.List;
import javax.swing.*;
public class Payment extends JDialog {
   // Database credentials
   final String DB_URL = "jdbc:mysql://localhost:3306/testdb";
   final String DB_USER = "root";
   final String DB_PASS = "";

   // Constructor for Payment Dialog
   public Payment(JFrame parent, int movieId, String movieTitle, List<Integer> seatNumbers,
String showtime, int userId) {
      super(parent, "Payment", true); // Modal dialog
      setSize(400, 320);
      setLocationRelativeTo(parent);
      setLayout(new BorderLayout());

      // Pricing
      int pricePerSeat = 250;
      int totalAmount = seatNumbers.size() * pricePerSeat;

      // Create a comma-separated string of seat numbers
      StringBuilder seatDisplay = new StringBuilder();
      for (int seat : seatNumbers) {
         seatDisplay.append(seat + 1).append(", ");
      }
      String seatsStr = seatDisplay.substring(0, seatDisplay.length() - 2); // Remove trailing
comma
```

```
        // Display booking details in a text area
        JTextArea area = new JTextArea(
            " 🎬 Movie: " + movieTitle +
            "\n🕒 Showtime: " + showtime +
            "\n 💺 Seats: " + seatsStr +
            "\n 💰 Total: ₹" + totalAmount
        );
        area.setEditable(false);
        area.setFont(new Font("Monospaced", Font.PLAIN, 14));
        area.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
        add(area, BorderLayout.CENTER);

        // Pay button setup
        JButton payButton = new JButton("Pay ₹" + totalAmount);
        payButton.setFont(new Font("Arial", Font.BOLD, 16));
        payButton.setBackground(new Color(0, 123, 255));
        payButton.setForeground(Color.WHITE);
        payButton.setFocusPainted(false);

        // Handle payment logic
        payButton.addActionListener(e -> {
            try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS)) {
                conn.setAutoCommit(false); // Start transaction

                // Insert bookings into 'bookings' table
                PreparedStatement bookingStmt = conn.prepareStatement(
                    "INSERT INTO bookings (movie_id, movie_title, showtime, seat_number, user_id)
VALUES (?, ?, ?, ?, ?)"
```

```
            );
            for (int seat : seatNumbers) {
                bookingStmt.setInt(1, movieId);
                bookingStmt.setString(2, movieTitle);
                bookingStmt.setString(3, showtime);
                bookingStmt.setInt(4, seat + 1); // Store seat number as 1-based
                bookingStmt.setInt(5, userId);
                bookingStmt.addBatch();
            }
            bookingStmt.executeBatch();

            // Insert a record into 'payment' table
            PreparedStatement paymentStmt = conn.prepareStatement(
                "INSERT INTO payment (movie_id, movie_title, showtime, seat_number, price, user_id) VALUES (?, ?, ?, ?, ?, ?)"
            );
            paymentStmt.setInt(1, movieId);
            paymentStmt.setString(2, movieTitle);
            paymentStmt.setString(3, showtime);
            paymentStmt.setString(4, seatsStr); // All seat numbers as string
            paymentStmt.setInt(5, totalAmount);
            paymentStmt.setInt(6, userId);
            paymentStmt.executeUpdate();
            conn.commit(); // Commit transaction
            // Show success message
            JOptionPane.showMessageDialog(this,
                "✅ Payment Successful!\n\n" +
                "🎬 Movie: " + movieTitle +
                "\n🕒 Showtime: " + showtime +
```

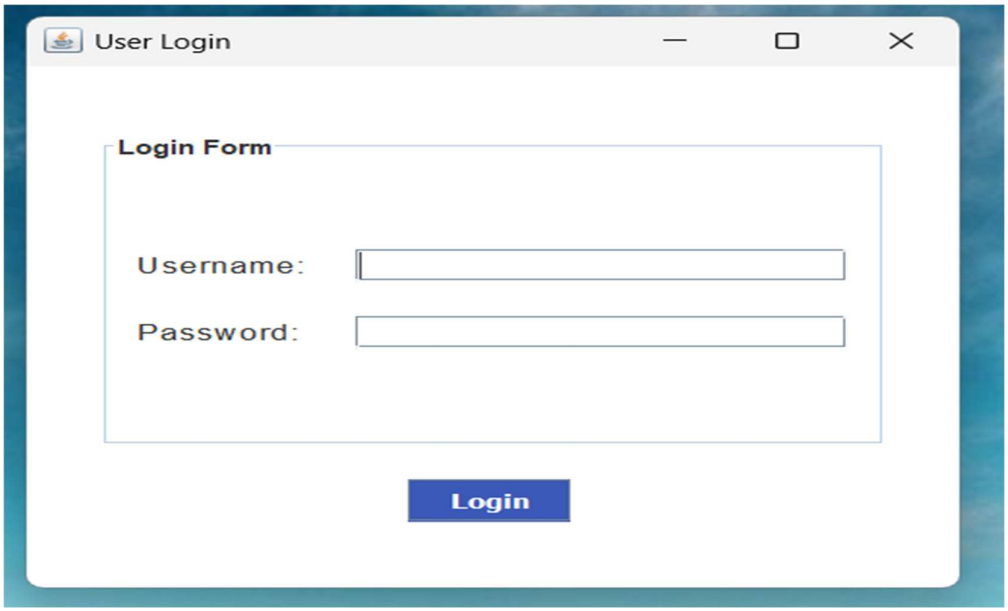
```
                "\n 💺 Seats: " + seatsStr +
                "\n 💰 Paid: ₹" + totalAmount,
                "Booking Confirmed",
                JOptionPane.INFORMATION_MESSAGE);
            dispose(); // Close dialog

        } catch (Exception ex) {
            ex.printStackTrace();
            // Rollback not explicitly done; could be added for safety
            JOptionPane.showMessageDialog(this,
                "❌ Payment Failed: " + ex.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    });
    // Add button to bottom panel
    JPanel buttonPanel = new JPanel();
    buttonPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 20, 10));
    buttonPanel.add(payButton);
    add(buttonPanel, BorderLayout.SOUTH);
  }
}
```

SQL Code:

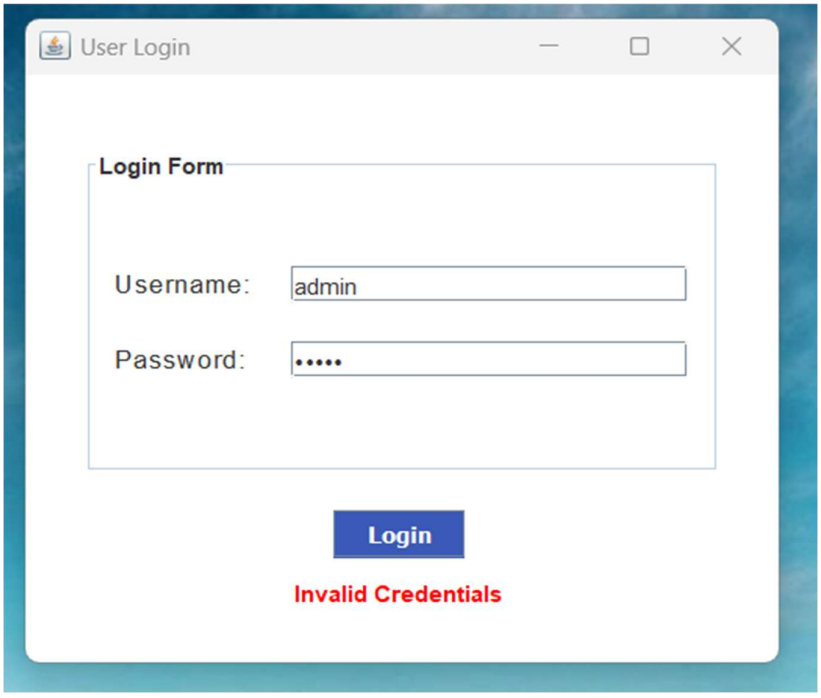
```
CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(100) NOT NULL,
   password VARCHAR(100) NOT NULL
);

CREATE TABLE IF NOT EXISTS movies (
   id INT AUTO_INCREMENT PRIMARY KEY,
   title VARCHAR(100),
   release_date VARCHAR(20),
   genre VARCHAR(50),
   theatre VARCHAR(100),
   image_path VARCHAR(255)
);
CREATE TABLE bookings (
   id INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT,
   FOREIGN KEY (user_id) REFERENCES users(id),
   movie_id INT NOT NULL,
   showtime VARCHAR(20) NOT NULL,
   seat_number INT NOT NULL,
   booking_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   CONSTRAINT fk_movie FOREIGN KEY (movie_id) REFERENCES movies(id)
      ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE payment (
   id INT AUTO_INCREMENT PRIMARY KEY,
   movie_id INT REFERENCES movies(id),
   movie_title VARCHAR(100),
   showtime VARCHAR(20),
   seat_number VARCHAR(100),
   price INT,
   user_id INT,
   FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Screenshots :

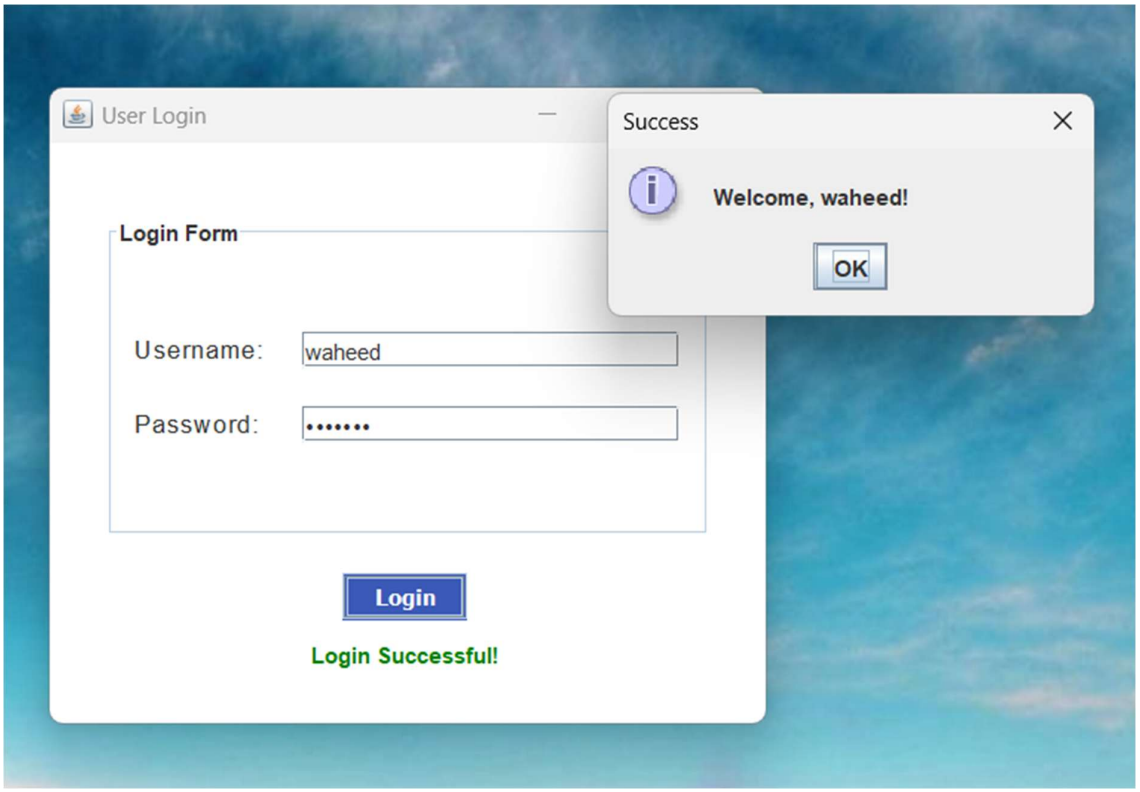
Login Page



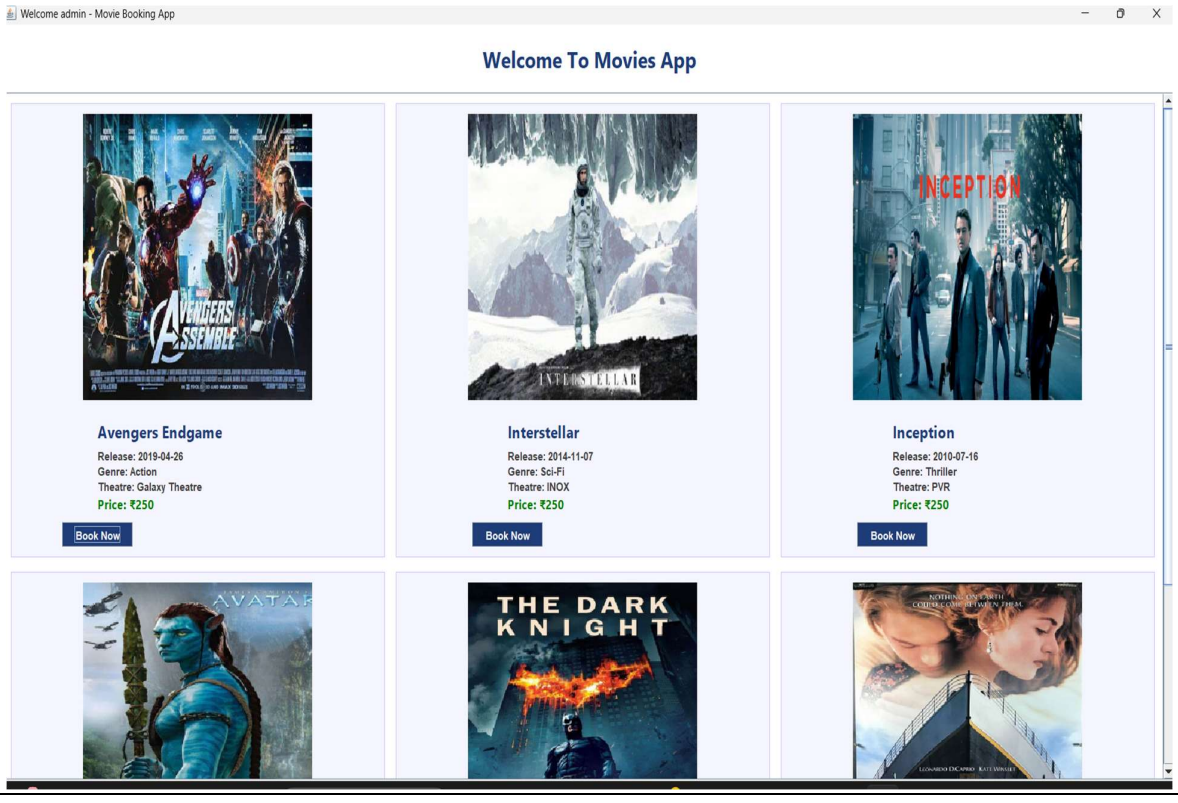
Invalid Credential Error Message



Login Success Message

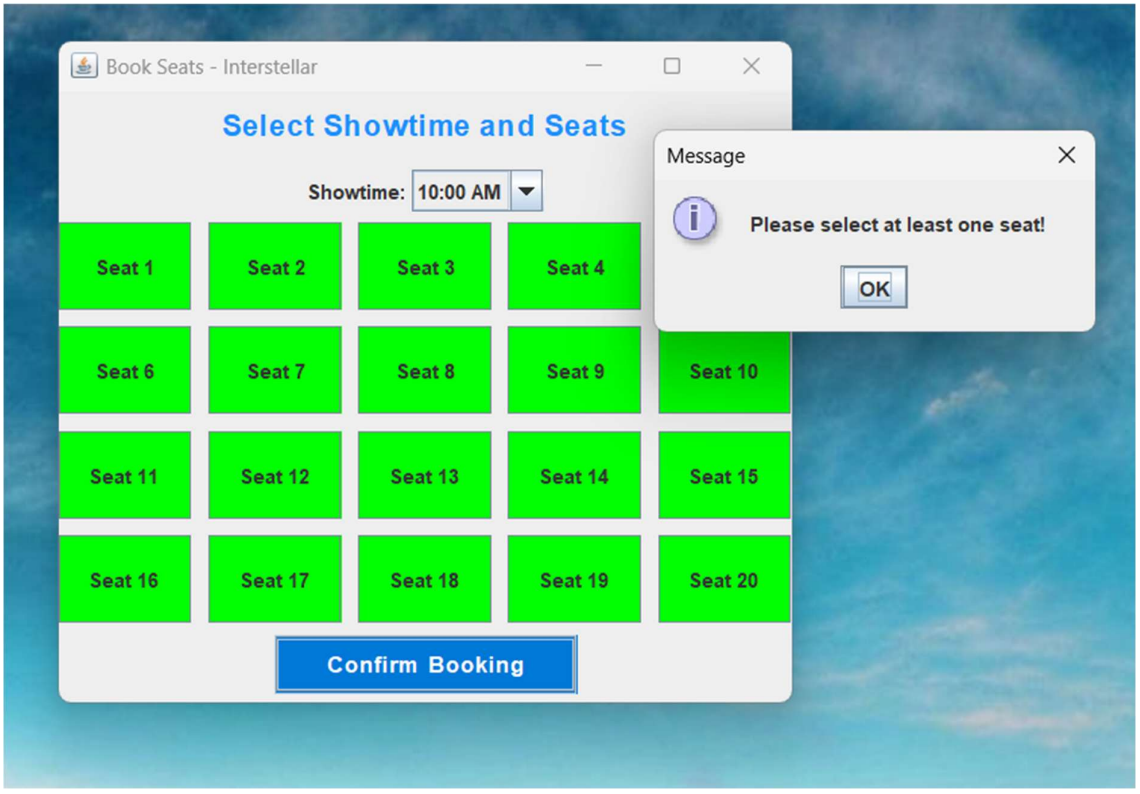


Movies Selection Page- list of available movies, each with a description, showtime, and theatre.



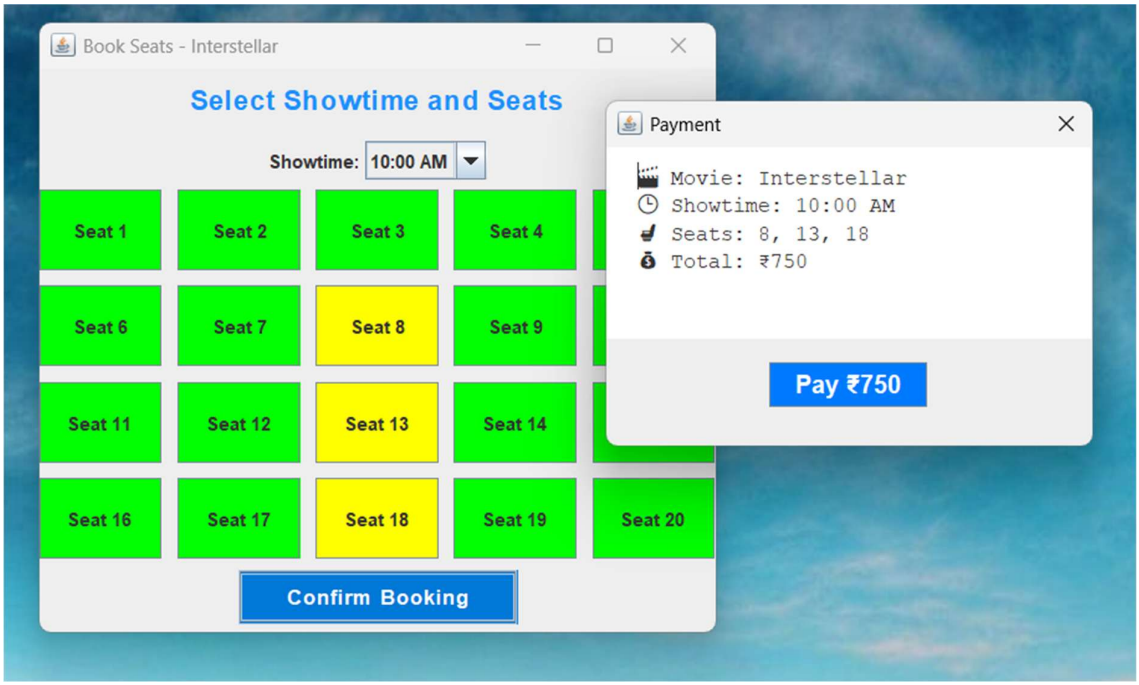
Showtime Selection And Seat Selection Page

Prompt to user To select atleast one seat

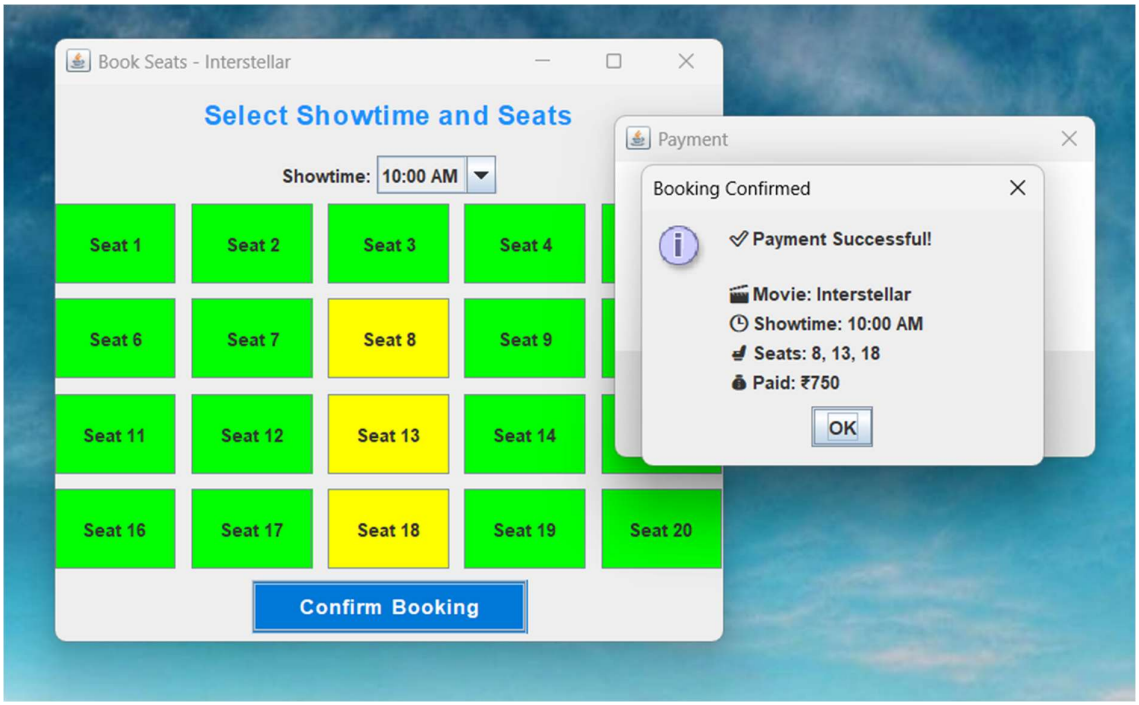


Selected Seats turn Yellow

After confirm Booking, prompt users to proceed with payment.

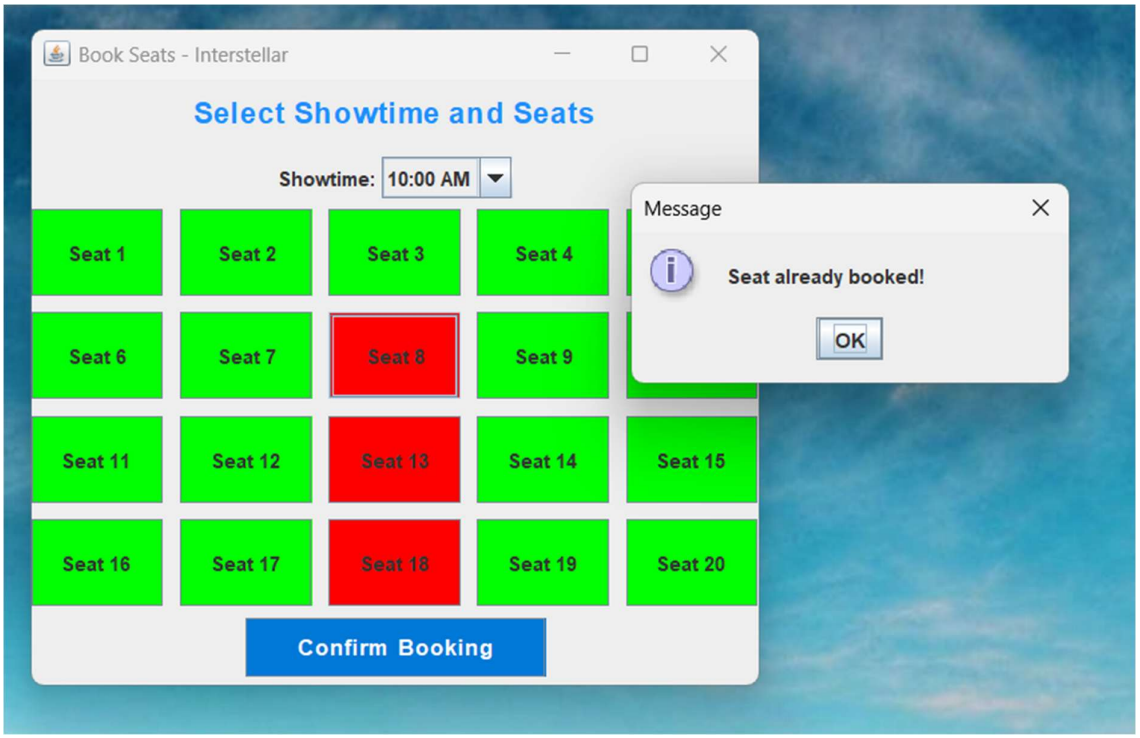


Payment is successful, displays confirmation message with details of the booked ticket.



Already Occupied Seats Turn Red ,

Prompts user with "Seat already booked!"



Database

User table

```
SELECT * FROM `users`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code

Show all | Number of rows: 25 Filter rows:

Extra options

	id	username	password
Edit Copy Delete	1	waheed	wah@123
Edit Copy Delete	2	john	joh@123
Edit Copy Delete	3	sam	sam@123

Movie List Table

```
SELECT * FROM `movies`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	id	title	release_date	genre	theatre	image_path
Edit Copy Delete	1	Avengers	2019-04-26	Action	Galaxy Theatre	image/avengers.jpg
Edit Copy Delete	2	Interstellar	2014-11-07	Sci-Fi	INOX	image/interseller.jpg
Edit Copy Delete	3	Inception	2010-07-16	Thriller	PVR	image/inception.jpg
Edit Copy Delete	4	Avatar	2009-12-18	Fantasy	Cinepolis	image/avatar.jpg
Edit Copy Delete	5	The Dark Knight	2008-07-18	Action	IMAX	image/darkknight.jpg
Edit Copy Delete	6	Titanic	1997-12-19	Romance	Carnival	image/titanic.jpg

Booking Table

```
SELECT * FROM `bookings`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

```
UPDATE `bookings` SET `id` = '4' WHERE `bookings`.`id` = 6;
```

Edit inline] [Edit] [Create PHP code]

Show all | Restore column order | Number of rows: 25 Filter rows: Search this table Sort by

Extra options

	id	user_id	movie_id	showtime	seat_number	booking_time	movie_title
Edit Copy Delete	1	1	2	10:00 AM	8	2025-04-09 21:50:50	Interstellar
Edit Copy Delete	2	1	2	10:00 AM	13	2025-04-09 21:50:50	Interstellar
Edit Copy Delete	3	3	3	7:00 PM	8	2025-04-09 22:00:39	Inception
Edit Copy Delete	4	3	3	7:00 PM	7	2025-04-09 22:00:39	Inception

Payment Table

```
SELECT * FROM `payment`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	id	movie_id	movie_title	showtime	seat_number	price	user_id
Edit Copy Delete	1	2	Interstellar	10:00 AM	8, 13, 18	750	1
Edit Copy Delete	2	2	Interstellar	1:00 PM	13	250	1
Edit Copy Delete	3	3	Inception	7:00 PM	8, 7, 6, 9, 13	1250	3

References:

Cay S. Horstmann. Core Java: Volume I - Fundamentals. 11th Edition, Pearson Education, 2018.

Cay S. Horstmann, Core Java: Volume II – Advanced Features, 11th Edition, Pearson Education, 2019.

Java Swing Documentations: https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/javax/swing/package-summary.html

html Stack Overflow Threads: https://stackoverflow.com/questions/18128966/where-is-the-mysql-jdbc jar-file-in-ubuntu

YouTube Video : Linux JDBC Tutorial | Using Java , MySql And J-Connector | Ubuntu Operating System by Unpossible POG https://youtu.be/4NE1r4Jv9B8?si=ZBho8Cbihzjecad