MOVIE APPLICATION

* SHERYL KATRINA M(230701310)
* SOWNDARIYA S(230701329)

Overview:

The MovieBookingApp is a Java-based graphical user interface (GUI) application that allows users to book movie seats, view reservations, update reservations, and delete reservations using a database backend. The app is built using JDBC for database operations and Swing for the user interface.

Main Features:

Login Panel: User authentication with a username and password.

Movie Booking System: Allows users to book seats, view existing reservations, update or delete reservations.

Database Integration: All reservations are stored and managed in a MySQL database.

---

Components:

1. Login Panel

The LoginPanel is displayed first and prompts the user for a username and password. If the credentials match, the user is authenticated and can access the movie booking system.

Class: LoginPanel

Key Components:

JTextField usernameField: Field for inputting the username.

JPasswordField passwordField: Field for inputting the password.

JButton loginButton: Button that triggers the login action.

Logic:

If the user inputs the correct username (root) and password (root), the movie booking interface is displayed.

If authentication fails, a message dialog is shown to the user.

---

2. Movie Booking System Panel

Once logged in, the user is presented with the main booking system, which includes tabs for different functionalities:

Book Seat: Allows users to book a seat.

View Reservations: Displays all reservations.

Update Reservation: Allows users to update reservation details.

Delete Reservation: Allows users to delete a reservation.

Class: MovieBookingApp

Key Components:

JTabbedPane tabbedPane: A tabbed pane that contains four tabs.

JTextArea outputArea: Displays the results of operations like viewing, updating, and deleting reservations.

JTextField guestNameField, seatNumberField, contactNumberField, reservationIdField, genreField: Various text fields for input.

---

3. Book Seat Tab

This tab allows the user to book a seat by entering their name, seat number, contact number, and genre of the movie.

Components:

JTextField guestNameField: Input for the guest's name.

JTextField seatNumberField: Input for the seat number.

JTextField contactNumberField: Input for the contact number.

JTextField genreField: Input for the genre of the movie.

JButton bookButton: Triggers the booking process.

Logic:

Ensures all fields are filled.

Checks if the seat is already booked in the database.

If valid, the reservation is inserted into the database.

---

4. View Reservations Tab

This tab allows the user to view all reservations stored in the database.

Components:

JButton viewButton: Fetches and displays all reservations.

Logic:

Retrieves all reservations from the database using a SELECT query.

Displays the reservations in the outputArea.

---

5. Update Reservation Tab

This tab allows the user to update an existing reservation by entering the reservation ID and new details (guest name, seat number, contact number).

Components:

JTextField reservationIdField: Input for the reservation ID to be updated.

JTextField newGuestNameField, newSeatNumberField, newContactNumberField: Inputs for the new reservation details.

JButton updateButton: Triggers the update process.

Logic:

If any fields are empty, an error message is shown.

If the reservation ID exists, the record is updated in the database with the new details.

---

6. Delete Reservation Tab

This tab allows the user to delete a reservation by providing the reservation ID.

Components:

JTextField reservationIdToDeleteField: Input for the reservation ID to be deleted.

JButton deleteButton: Triggers the delete operation.

Logic:

If the reservation ID exists, the reservation is deleted from the database.

A success or failure message is displayed in the outputArea.

---

Database Integration:

Database URL: jdbc:mysql://localhost:3306/movie\_db

(Assumes a MySQL database is running locally with the database name movie\_db.)

Database Authentication:

Username: root

Password: Cat@2006

Database Operations:

Book Seat: Inserts a new record into the reservations table.

View Reservations: Fetches all records from the reservations table.

Update Reservation: Updates an existing reservation record based on the provided reservation ID.

Delete Reservation: Deletes a reservation record based on the provided reservation ID.

---

Methods Overview:

1. LoginPanel ActionListener

loginButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// Handle login action: validate credentials

}

});

2. bookSeat()

Books a seat after validating input fields:

private void bookSeat() {

// Validate fields, check if seat is available, and insert into the database.

}

3. viewReservations()

Displays all reservations:

private void viewReservations() {

// Execute SELECT query and display results in the outputArea.

}

4. updateReservation()

Updates an existing reservation:

private void updateReservation(String reservationId, String newGuestName, String newSeatNumber, String newContactNumber) {

// Execute UPDATE query to modify reservation details.

}

5. deleteReservation()

Deletes a reservation based on the reservation ID:

private void deleteReservation(String reservationId, JTextField reservationIdToDeleteField) {

// Execute DELETE query to remove reservation.

}

---

How to Run the Application:

1. Set Up the Database:

Ensure MySQL is installed and running.

Create a database named movie\_db.

Create a table for reservations:

CREATE TABLE reservations (

id INT AUTO\_INCREMENT PRIMARY KEY,

guest\_name VARCHAR(255),

seat\_number INT,

contact\_number VARCHAR(255),

genre VARCHAR(255)

);

2. Run the Java Application:

Ensure JDBC driver for MySQL is in your classpath.

Compile and run the application using any Java IDE or command line.

---

Dependencies:

JDBC: Used for database connectivity.

Swing: GUI framework used for building the user interface.

MySQL: Relational database for storing reservations.

SAMPLE CODE

FORNTEND CODE

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

class MovieBookingApp extends JFrame {

private static final String url = "jdbc:mysql://localhost:3306/movie\_db"; // Change database name

private static final String username = "root";

private static final String password = "Cat@2006";

private JTextField guestNameField, seatNumberField, contactNumberField, reservationIdField,genreField;

private JTextArea outputArea;

private Connection connection;

// Login Panel for username and password authentication

class LoginPanel extends JPanel {

private JTextField usernameField;

private JPasswordField passwordField;

public LoginPanel() {

setLayout(new BorderLayout());

setBackground(new Color(46, 46, 46));

// Title Label

JLabel titleLabel = new JLabel("Movie Plus +", JLabel.CENTER);

titleLabel.setFont(new Font("Arial", Font.BOLD, 30));

titleLabel.setForeground(Color.ORANGE);

add(titleLabel, BorderLayout.NORTH);

// Form Panel

JPanel formPanel = new JPanel(new GridBagLayout());

formPanel.setBackground(new Color(46, 46, 46)); // Match background color

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

gbc.fill = GridBagConstraints.HORIZONTAL;

Font customFont = new Font("Arial", Font.BOLD, 18);

Color textColor = Color.WHITE;

// Username Label and Field

JLabel usernameLabel = new JLabel("Username:");

usernameLabel.setFont(customFont);

usernameLabel.setForeground(textColor);

gbc.gridx = 0;

gbc.gridy = 0;

formPanel.add(usernameLabel, gbc);

usernameField = new JTextField(20);

gbc.gridx = 1;

gbc.gridy = 0;

formPanel.add(usernameField, gbc);

// Password Label and Field

JLabel passwordLabel = new JLabel("Password:");

passwordLabel.setFont(customFont);

passwordLabel.setForeground(textColor);

gbc.gridx = 0;

gbc.gridy = 1;

formPanel.add(passwordLabel, gbc);

passwordField = new JPasswordField(20);

gbc.gridx = 1;

gbc.gridy = 1;

formPanel.add(passwordField, gbc);

// Login Button

JButton loginButton = new JButton("Login");

loginButton.setFont(customFont);

loginButton.setBackground(new Color(70, 130, 180)); // Steel blue

loginButton.setForeground(Color.WHITE);

loginButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String enteredUsername = usernameField.getText().trim();

String enteredPassword = new String(passwordField.getPassword()).trim();

if (enteredUsername.equals("root") && enteredPassword.equals("root")) {

showMovieBookingSystem();

} else {

JOptionPane.showMessageDialog(MovieBookingApp.this, "Invalid username or password", "Login Failed", JOptionPane.ERROR\_MESSAGE);

}

}

});

gbc.gridx = 0;

gbc.gridy = 2;

gbc.gridwidth = 2;

gbc.anchor = GridBagConstraints.CENTER;

formPanel.add(loginButton, gbc);

// Add form panel to the center

add(formPanel, BorderLayout.CENTER);

}

}

public MovieBookingApp() {

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Database connection failed", "Error", JOptionPane.ERROR\_MESSAGE);

}

setTitle("Movie Booking System");

setSize(470, 430);

setLocationRelativeTo(null); // Center the frame on the screen

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

initUI();

}

private void initUI() {

// First show the login screen

setContentPane(new LoginPanel());

}

// Method to show the movie booking system after login

private void showMovieBookingSystem() {

// Change the content pane to the actual movie booking system

setContentPane(createMovieBookingUI());

revalidate(); // Refresh the frame to show the movie booking system

}

private JPanel createMovieBookingUI() {

JPanel panel = new JPanel();

panel.setLayout(new BorderLayout());

// Create a tabbed pane to organize operations

JTabbedPane tabbedPane = new JTabbedPane();

tabbedPane.addTab("Book Seat", createBookSeatTab());

tabbedPane.addTab("View Reservations", createViewReservationsTab());

tabbedPane.addTab("Update Reservation", createUpdateReservationTab());

tabbedPane.addTab("Delete Reservation", createDeleteReservationTab());

panel.add(tabbedPane, BorderLayout.CENTER);

// Output Area to show results

outputArea = new JTextArea(8, 7);

outputArea.setEditable(false);

JScrollPane scrollPane = new JScrollPane(outputArea);

panel.add(scrollPane, BorderLayout.SOUTH);

return panel;

}

// Create Book Seat Tab

private JPanel createBookSeatTab() {

JPanel bookPanel = new JPanel();

bookPanel.setLayout(new GridLayout(5, 2, 10, 10));

JLabel guestNameLabel = new JLabel("Name:");

guestNameField = new JTextField();

JLabel seatNumberLabel = new JLabel("Seat Number:");

seatNumberField = new JTextField();

JLabel contactNumberLabel = new JLabel("Contact Number:");

contactNumberField = new JTextField();

JLabel genreLabel = new JLabel("Genre:");

genreField = new JTextField();

JButton bookButton = new JButton("Book Seat");

bookButton.setBackground(new Color(255, 182, 193)); // Light Pink Button

bookButton.setForeground(Color.DARK\_GRAY); // Text color

bookButton.setFont(new Font("Arial", Font.BOLD, 16));

bookButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

bookSeat();

}

});

bookPanel.add(guestNameLabel);

bookPanel.add(guestNameField);

bookPanel.add(seatNumberLabel);

bookPanel.add(seatNumberField);

bookPanel.add(contactNumberLabel);

bookPanel.add(contactNumberField);

bookPanel.add(genreLabel);

bookPanel.add(genreField);

bookPanel.add(new JLabel());

bookPanel.add(bookButton);

return bookPanel;

}

private JLabel createCustomLabel(String text) {

JLabel label = new JLabel(text);

label.setForeground(Color.WHITE);

label.setFont(new Font("Arial", Font.BOLD, 16));

return label;

}

// Create View Reservations Tab

// Updated method to fetch and display data in the table

private JPanel createViewReservationsTab() {

JPanel viewPanel = new JPanel();

viewPanel.setLayout(new BorderLayout());

viewPanel.setBackground(new Color(46, 46, 46));

JButton viewButton = new JButton("View Reservations");

viewButton.setBackground(new Color(70, 130, 180));

viewButton.setForeground(Color.WHITE);

viewButton.setFont(new Font("Arial", Font.BOLD, 14)); // Set smaller font

viewButton.setPreferredSize(new Dimension(100, 30));

viewButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

viewReservations(); // This method will fetch and display reservations

}

});

viewPanel.add(viewButton, BorderLayout.CENTER);

return viewPanel;

}

// Create Update Reservation Tab

private JPanel createUpdateReservationTab() {

JPanel updatePanel = new JPanel();

updatePanel.setLayout(new GridLayout(6, 2));

updatePanel.setBackground(new Color(46, 46, 46)); // Updated to accommodate a message at the bottom

JLabel reservationIdLabel = new JLabel("Reservation ID:");

reservationIdLabel.setForeground(Color.WHITE); // Text color

JTextField reservationIdField = new JTextField();

JLabel newGuestNameLabel = new JLabel("New Name:");

newGuestNameLabel.setForeground(Color.WHITE);

JTextField newGuestNameField = new JTextField();

JLabel newSeatNumberLabel = new JLabel("New Seat Number:");

newGuestNameLabel.setForeground(Color.WHITE);

JTextField newSeatNumberField = new JTextField();

JLabel newContactNumberLabel = new JLabel("New Contact Number:");

newContactNumberLabel.setForeground(Color.WHITE);

JTextField newContactNumberField = new JTextField();

JButton updateButton = new JButton("Update Reservation");

updateButton.setBackground(new Color(70, 130, 180)); // Button color

updateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String reservationId = reservationIdField.getText().trim();

String newGuestName = newGuestNameField.getText().trim();

String newSeatNumber = newSeatNumberField.getText().trim();

String newContactNumber = newContactNumberField.getText().trim();

if (reservationId.isEmpty() || newGuestName.isEmpty() || newSeatNumber.isEmpty() || newContactNumber.isEmpty()) {

outputArea.setText("Please fill all fields.");

} else {

updateReservation(reservationId, newGuestName, newSeatNumber, newContactNumber);

}

}

});

updatePanel.add(reservationIdLabel);

updatePanel.add(reservationIdField);

updatePanel.add(newGuestNameLabel);

updatePanel.add(newGuestNameField);

updatePanel.add(newSeatNumberLabel);

updatePanel.add(newSeatNumberField);

updatePanel.add(newContactNumberLabel);

updatePanel.add(newContactNumberField);

updatePanel.add(updateButton);

return updatePanel;

}

// Create Delete Reservation Tab

private JPanel createDeleteReservationTab() {

JPanel deletePanel = new JPanel();

deletePanel.setLayout(new BorderLayout());

deletePanel.setBackground(new Color(46, 46, 46));

// Label and Text Field for Reservation ID

JLabel reservationIdLabel = new JLabel("Enter Reservation ID to delete:");

reservationIdLabel.setForeground(Color.WHITE);

JTextField reservationIdToDeleteField = new JTextField();

// Button to trigger deletion

JButton deleteButton = new JButton("Delete Reservation");

deleteButton.setBackground(new Color(178, 34, 34));

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String reservationId = reservationIdToDeleteField.getText().trim();

if (!reservationId.isEmpty()) {

deleteReservation(reservationId, reservationIdToDeleteField);

} else {

outputArea.setText("Please enter a Reservation ID.");

}

}

});

// Add components to the panel

JPanel inputPanel = new JPanel(new GridLayout(2, 1));

inputPanel.setBackground(new Color(46, 46, 46));

inputPanel.add(reservationIdLabel);

inputPanel.add(reservationIdToDeleteField);

deletePanel.add(inputPanel, BorderLayout.CENTER);

deletePanel.add(deleteButton, BorderLayout.SOUTH);

return deletePanel;

}

// Book Seat Method

private void bookSeat() {

String guestName = guestNameField.getText().trim();

String seatNumberText = seatNumberField.getText().trim();

String contactNumber = contactNumberField.getText().trim();

String genre = genreField.getText().trim();

if (guestName.isEmpty() || seatNumberText.isEmpty() || contactNumber.isEmpty() || genre.isEmpty()) {

outputArea.setText("Please fill all fields before booking a seat.");

return;

}

int seatNumber;

try {

seatNumber = Integer.parseInt(seatNumberText);

if (seatNumber <= 0) {

outputArea.setText("Seat number must be a positive number.");

return;

}

} catch (NumberFormatException e) {

outputArea.setText("Invalid seat number. Please enter a valid numeric value.");

return;

}

try {

// Check if the seat is already booked

String checkSeatSql = "SELECT COUNT(\*) AS seat\_count FROM reservations WHERE seat\_number = ?";

try (PreparedStatement checkStmt = connection.prepareStatement(checkSeatSql)) {

checkStmt.setInt(1, seatNumber);

ResultSet rs = checkStmt.executeQuery();

if (rs.next() && rs.getInt("seat\_count") > 0) {

outputArea.setText("The seat is already booked. Please choose a different seat.");

return;

}

}

// Insert booking into the database

String sql = "INSERT INTO reservations (guest\_name, seat\_number, contact\_number, genre) VALUES (?, ?, ?, ?)";

try (PreparedStatement stmt = connection.prepareStatement(sql)) {

stmt.setString(1, guestName);

stmt.setInt(2, seatNumber);

stmt.setString(3, contactNumber);

stmt.setString(4, genre);

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

outputArea.setText("Seat booked successfully for " + guestName + " (Seat Number: " + seatNumber + ", Genre: " + genre + ").");

guestNameField.setText("");

seatNumberField.setText("");

contactNumberField.setText("");

genreField.setText("");

} else {

outputArea.setText("Booking failed. Please try again.");

}

}

} catch (SQLException e) {

e.printStackTrace();

outputArea.setText("Error occurred while booking the seat: " + e.getMessage());

}

}

// View Reservations Method

private void viewReservations() {

String sql = "SELECT \* FROM reservations"; // Select all reservations

try (Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

outputArea.setText("");

outputArea.setFont(new Font("Arial", Font.PLAIN, 16)); // Clear previous output

while (rs.next()) {

String id = rs.getString("id");

String guestName = rs.getString("guest\_name");

String seatNumber = rs.getString("seat\_number"); // Fetch seat number as a String

String contactNumber = rs.getString("contact\_number");

String genre = rs.getString("genre"); // Assuming 'genre' is part of the reservations table

// Append the results to the outputArea

outputArea.append("ID: " + id + ", Name: " + guestName + ", Seat: " + seatNumber

+ ", Contact: " + contactNumber + ", Genre: " + genre + "\n");

}

if (outputArea.getText().isEmpty()) {

outputArea.setText("No reservations found.");

}

} catch (SQLException e) {

e.printStackTrace();

outputArea.setText("Error occurred while fetching reservations: " + e.getMessage());

}

}

// Update Reservation Method

private void updateReservation(String reservationId, String newGuestName, String newSeatNumber, String newContactNumber) {

String sql = "UPDATE reservations SET guest\_name = ?, seat\_number = ?, contact\_number = ? WHERE id = ?";

try (PreparedStatement stmt = connection.prepareStatement(sql)) {

// Set parameters for the update query

stmt.setString(1, newGuestName);

stmt.setString(2, newSeatNumber); // Seat number should be a string

stmt.setString(3, newContactNumber);

stmt.setInt(4, Integer.parseInt(reservationId)); // Reservation ID should be an integer

// Execute the update query

int rowsAffected = stmt.executeUpdate();

// Check if any rows were affected (indicating success)

if (rowsAffected > 0) {

outputArea.setText("Reservation updated successfully!");

} else {

// If no rows were affected, the reservation ID might be incorrect

outputArea.setText("No reservation found with the given ID or update failed.");

}

} catch (SQLException e) {

e.printStackTrace();

outputArea.setText("Error occurred while updating the reservation: " + e.getMessage());

}

}

// Delete Reservation Method

private void deleteReservation(String reservationId, JTextField reservationIdToDeleteField) {

String sql = "DELETE FROM reservations WHERE id = ?";

try (PreparedStatement stmt = connection.prepareStatement(sql)) {

stmt.setInt(1, Integer.parseInt(reservationId));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

outputArea.setText("Reservation deleted successfully!");

reservationIdToDeleteField.setText(""); // Clear the input field after successful deletion

} else {

outputArea.setText("No reservation found with the given ID.");

}

} catch (SQLException e) {

e.printStackTrace();

outputArea.setText("Error occurred while deleting the reservation.");

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new MovieBookingApp().setVisible(true);

}

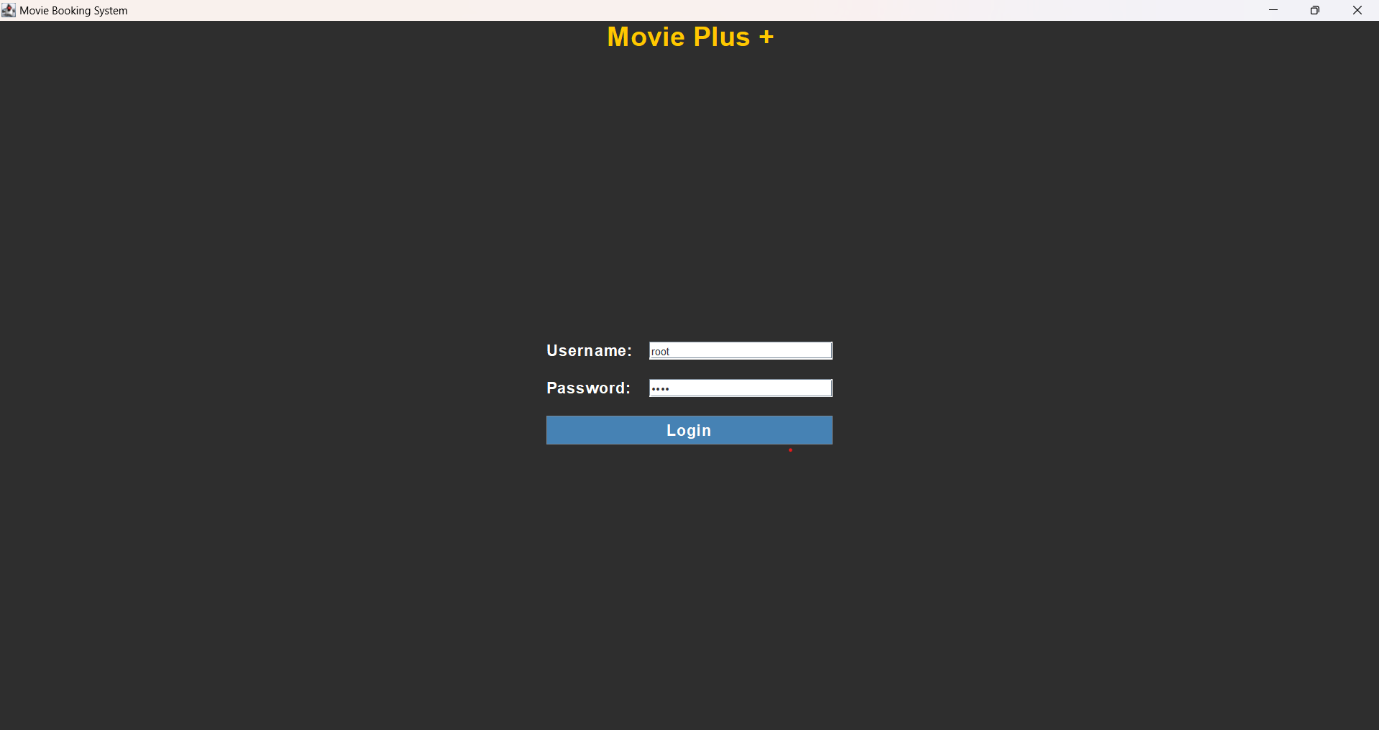
});

}

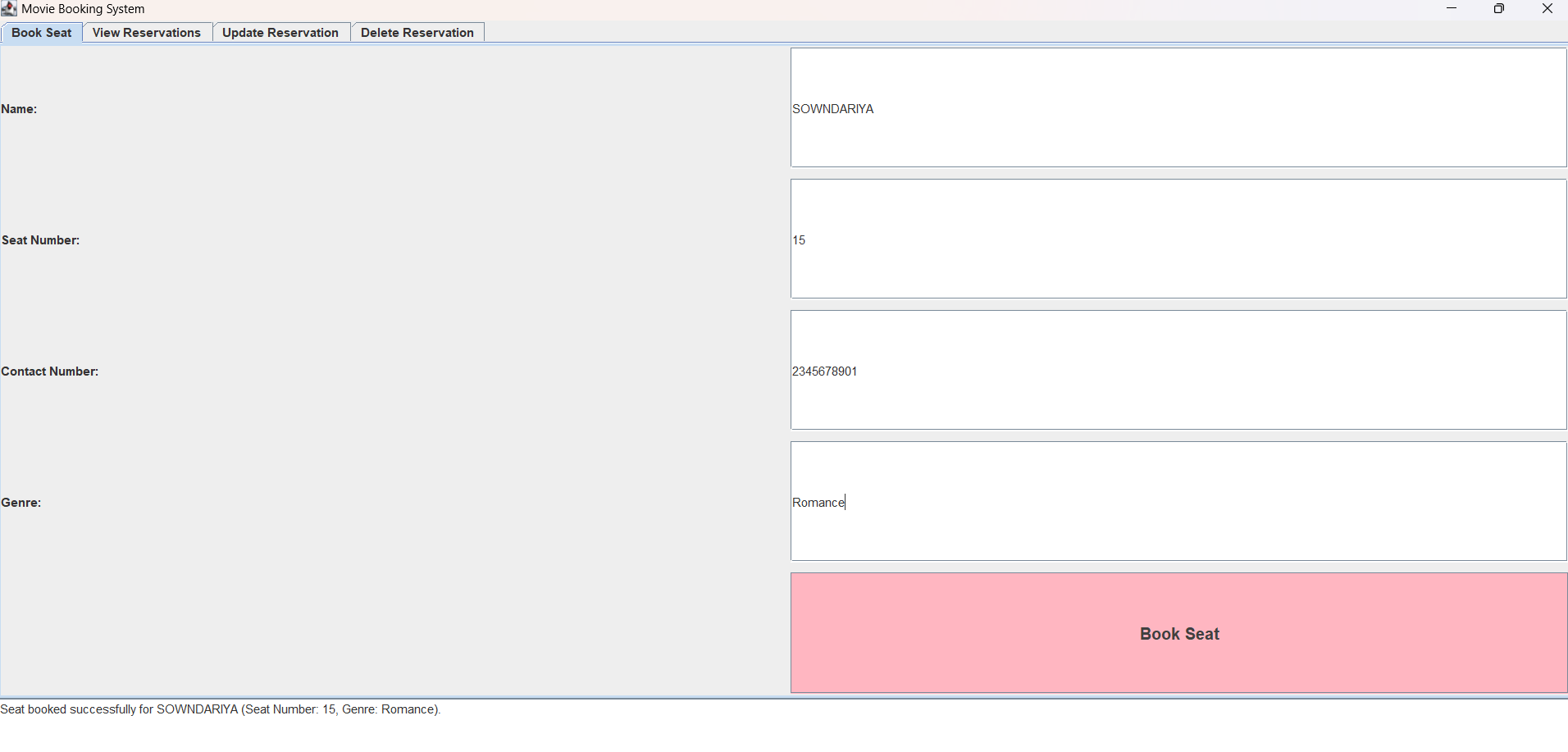
}

SNAPSHOTS

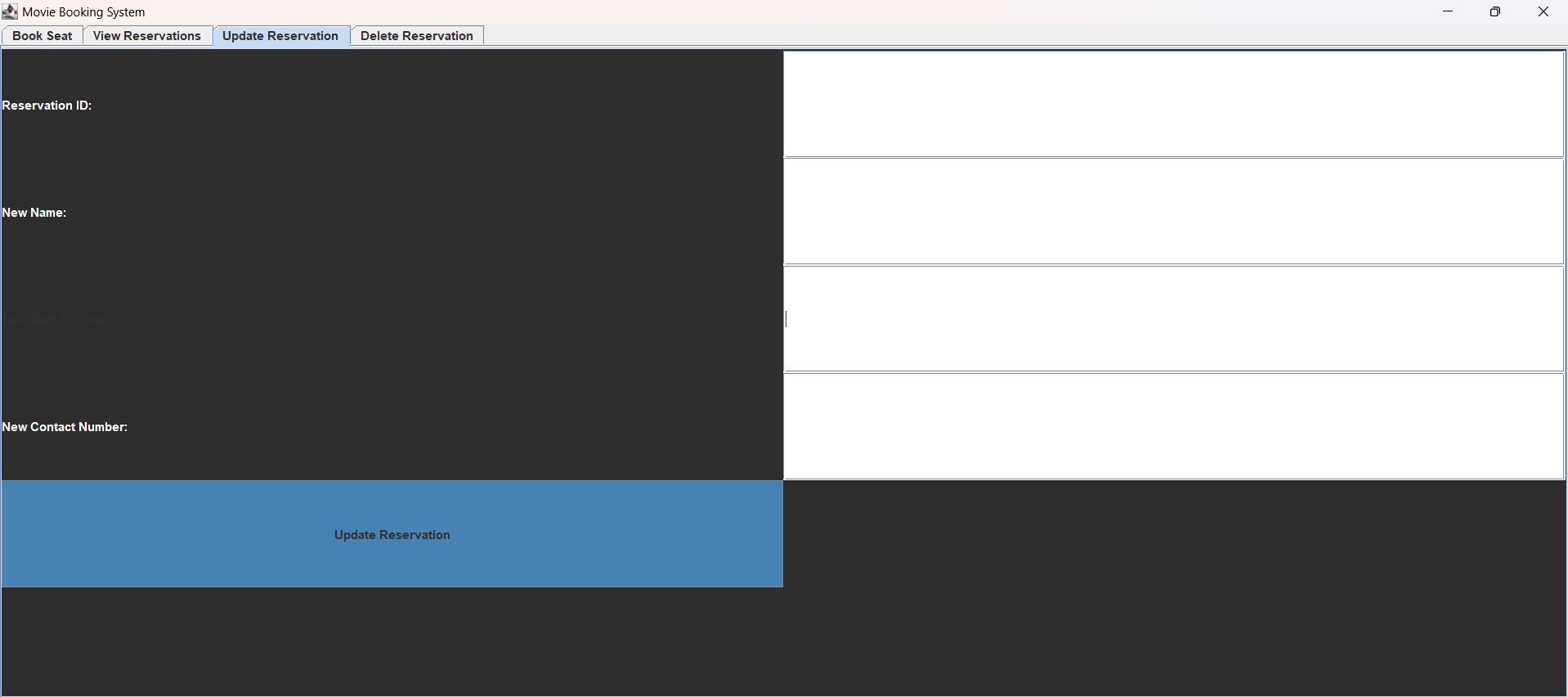
LOGIN PAGE:



BOOK SEAT:



UPDATE RESERVATION:



Conclusion:

The MovieBookingApp is a simple yet functional Java application for booking movie tickets. It demonstrates database connectivity with JDBC and builds a user-friendly interface using Swing. The core functionality includes booking, viewing, updating, and deleting reservations, all of which interact with a MySQL database.