MOVIE BOOKING SYSTEMS.

A MINI-PROJECT REPORT

Submitted by

SHERYL KATRINA M 230701310
SOWNDARIYA R 230701329

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 2023

BONAFIDE CERTIFICATE

Certified that this project “MOVIE BOOKING SYSTEMS” is the

bonafide work of “SHERYL KATRINA M AND SOWNDARIYA R” who carried out the

project work under my supervision.

SIGNATURE

MRS V JANANEE

ASSISTANT PROFESSOR

Dept. of Computer Science and Engg,
Engg,

Rajalakshmi Engineering College

Chennai

SIGNATURE

MR SARAVANA GOKUL

ASSISTANT PROFESSOR

Dept. of Computer Science and

Rajalakshmi Engineering College

Chennai

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

TABLE OF CONTENTS

ABSTRACT

Abstract:

This project presents a comprehensive **Movie Booking System** implemented using **Java Swing** for the user interface and **MySQL** for backend database management. The application offers an intuitive and modern graphical user interface, allowing users to perform key operations such as booking movie seats, viewing reservations, updating reservation details, and deleting reservations.

The system begins with a login panel for authentication, ensuring restricted access to the booking functionalities. Post-login, users are guided through an organized tabbed interface to perform various operations. Data validation and user-friendly messages enhance the experience by ensuring error-free inputs and clear feedback.

The backend leverages a robust MySQL database for secure storage and management of reservation details. Prepared SQL statements are utilized to prevent SQL injection and maintain data integrity. Additionally, the system implements essential features such as duplicate seat number checks and mandatory field validations to ensure accuracy.

This project serves as a practical demonstration of integrating Java Swing's GUI capabilities with database operations in MySQL, offering a reliable and user-centric movie booking solution. Potential future enhancements include integrating dynamic seat layouts and secure credential management to improve scalability and security.

INTRODUCTION

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.

Key Features of the Movie Booking System

1. **User Authentication**:
    - Login panel with username and password fields for secure access.
    - Restricts access to the booking system, ensuring only authorized users can perform operations.
2. **Booking Management**:
    - **Seat Booking**:
        - Allows users to book seats by providing details such as guest name, seat number, contact information, and movie genre.
        - Validates inputs to ensure all fields are filled and seat numbers are positive integers.
        - Prevents booking duplicate seats by checking availability in the database.
3. **Reservation Operations**:

- **View Reservations**:
  - Displays all existing reservations in a clear, user-friendly format.
- **Update Reservations**:
  - Allows users to update guest name, seat number, and contact details for an existing reservation.
  - Validates inputs to ensure proper formatting and prevents empty fields.
- **Delete Reservations**:
  - Enables users to delete a reservation by entering the reservation ID.
  - Provides clear feedback on successful or failed deletion attempts.

4. **Database Integration**:
   - Uses MySQL as the backend for storing and managing reservation data.
   - Ensures data integrity and security through **PreparedStatement** to prevent SQL injection attacks.
   - Includes error handling to notify users of database connectivity or query issues.
5. **User Interface**:
   - Designed with **Java Swing**, featuring a modern and user-friendly graphical interface.
   - Tabbed interface to organize functionalities for easy navigation.
   - Custom styling with consistent color themes, fonts, and layout for improved aesthetics and usability.
6. **Feedback Mechanism**:

- **Output Area**:
  - Provides clear feedback for all operations, including success or error messages.
- Ensures users are informed about the status of their actions in real time.

7. **Validation and Error Handling**:
   - Checks for empty fields, invalid inputs, and duplicate reservations.
   - Displays appropriate error messages for invalid actions or database errors.
8. **Scalability**:
   - Designed to accommodate additional features like dynamic seat layout, genre filtering, and user registration for future development.

This system provides a complete foundation for a movie booking platform, integrating seamless user interaction with robust database management.

SCOPE OF THE PROJECT

The **Movie Booking System** project is designed to address the operational needs of a movie theater by offering a reliable, efficient, and user-friendly solution for seat reservations and management. This project is scalable and versatile, making it suitable for use in small to mid-sized movie theaters. The scope of this project is categorized as follows:

---

1. Functional Scope:

- **User Authentication**:
  - Secure access to the system through a login interface.
  - Prevents unauthorized access and ensures data privacy.
- **Booking Management**:
  - Enables customers to book seats by providing details such as name, seat number, contact information, and movie genre.
  - Prevents duplicate bookings by validating seat availability.
- **Reservation Management**:
  - **View Reservations**: View all existing reservations with complete details.
  - **Update Reservations**: Modify reservation details such as guest name, seat number, and contact information.
  - **Delete Reservations**: Remove reservations from the system by providing a valid reservation ID.
- **Database Integration**:
  - Centralized storage of all reservation data using a MySQL database.
  - Supports CRUD (Create, Read, Update, Delete) operations to manage the reservation lifecycle effectively.

---

2. Technical Scope:

- **Programming Language**: Java with Swing for the graphical user interface (GUI).
- **Database**: MySQL for backend data storage and retrieval.
- **Validation**: Input validation for fields like seat number, contact details, and reservation ID.
- **Security**: Use of Prepared Statements to prevent SQL injection

attacks and ensure data security.

---

3. User Scope:

- **Movie Theater Staff**:
  - Simplifies the process of managing bookings, reducing manual errors.
  - Provides a centralized interface to manage customer reservations.
- **Customers**:
  - Streamlines the booking experience with real-time feedback and error handling.
  - Allows customization of reservation details, such as genre preferences.

---

4. Future Enhancements:

- **Dynamic Seat Layouts**:
  - Incorporate a graphical seat selection feature with real-time updates on availability.
- **Customer Registration**:
  - Add user accounts with saved preferences and booking history.
- **Payment Integration**:
  - Integrate secure payment gateways for online ticket purchases.
- **Genre and Time Filtering**:
  - Allow users to filter movies by genre or show timings for more personalized options.
- **Report Generation**:
  - Generate daily, weekly, or monthly reports on seat bookings and revenue.

---

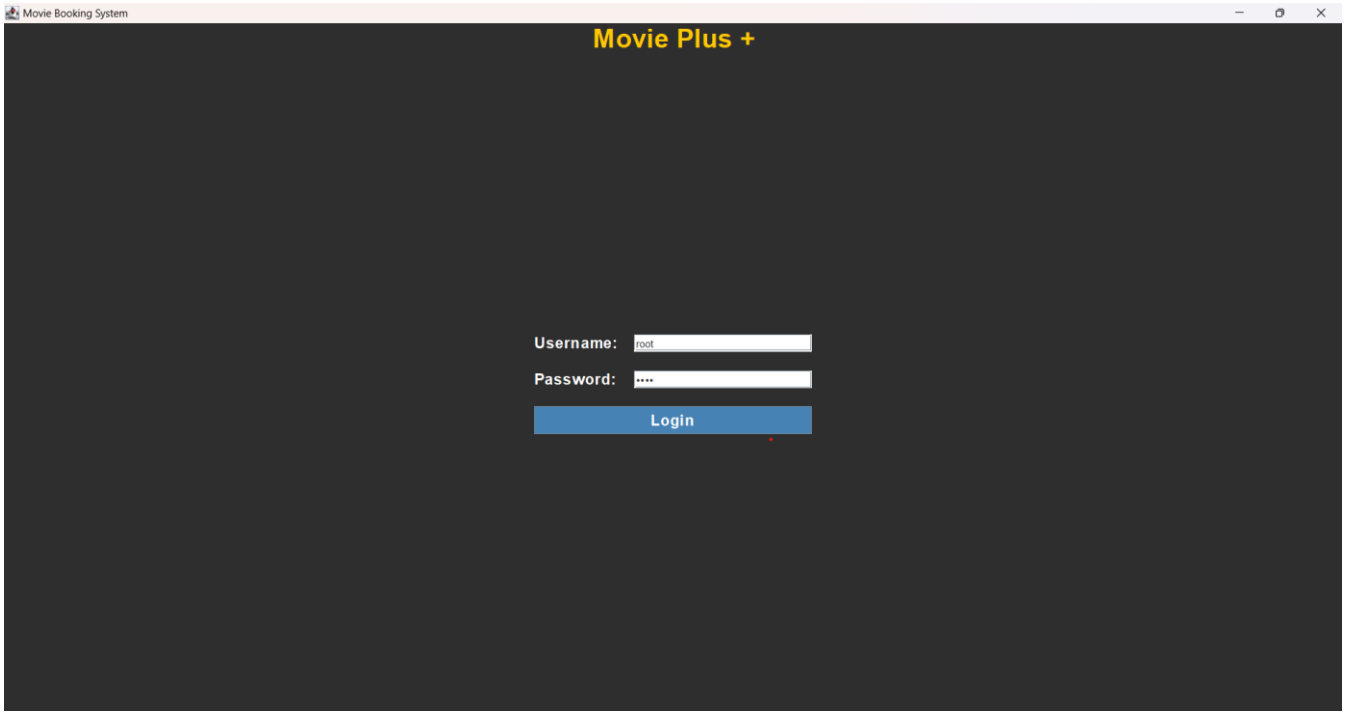
5. Limitations:

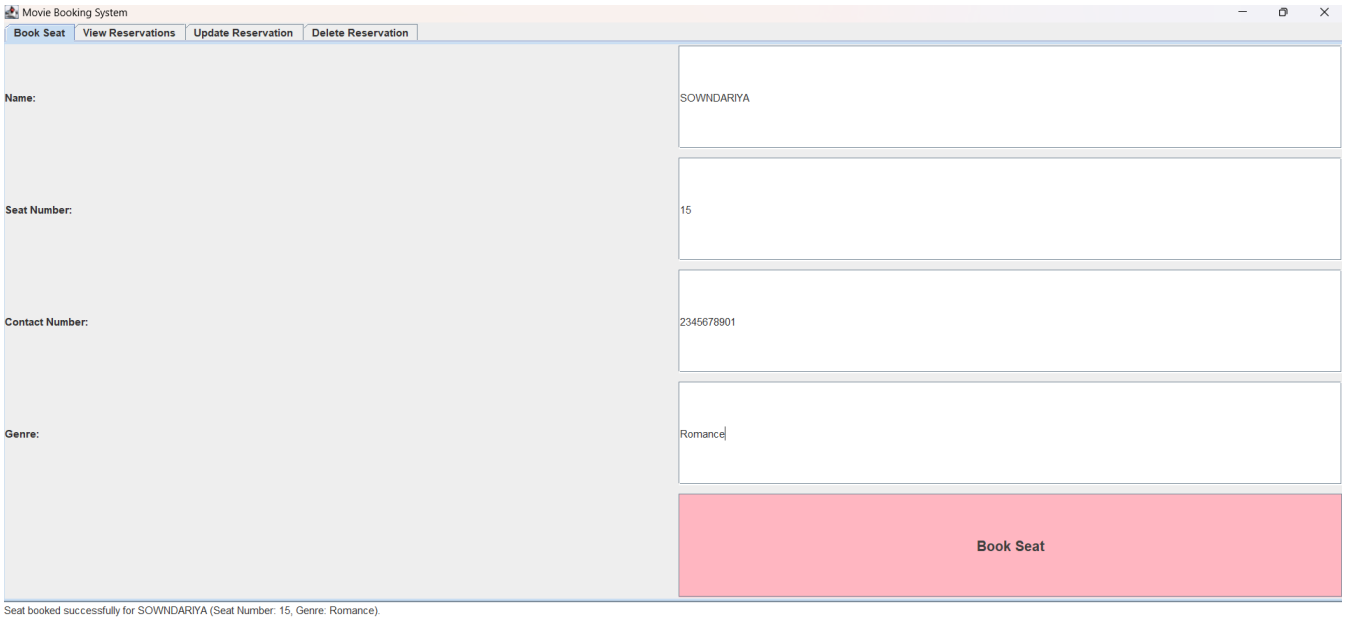
- Currently tailored for small to mid-sized theaters.
- Requires manual database setup and configuration.
- Lacks advanced features like real-time online ticket sales and seat maps.

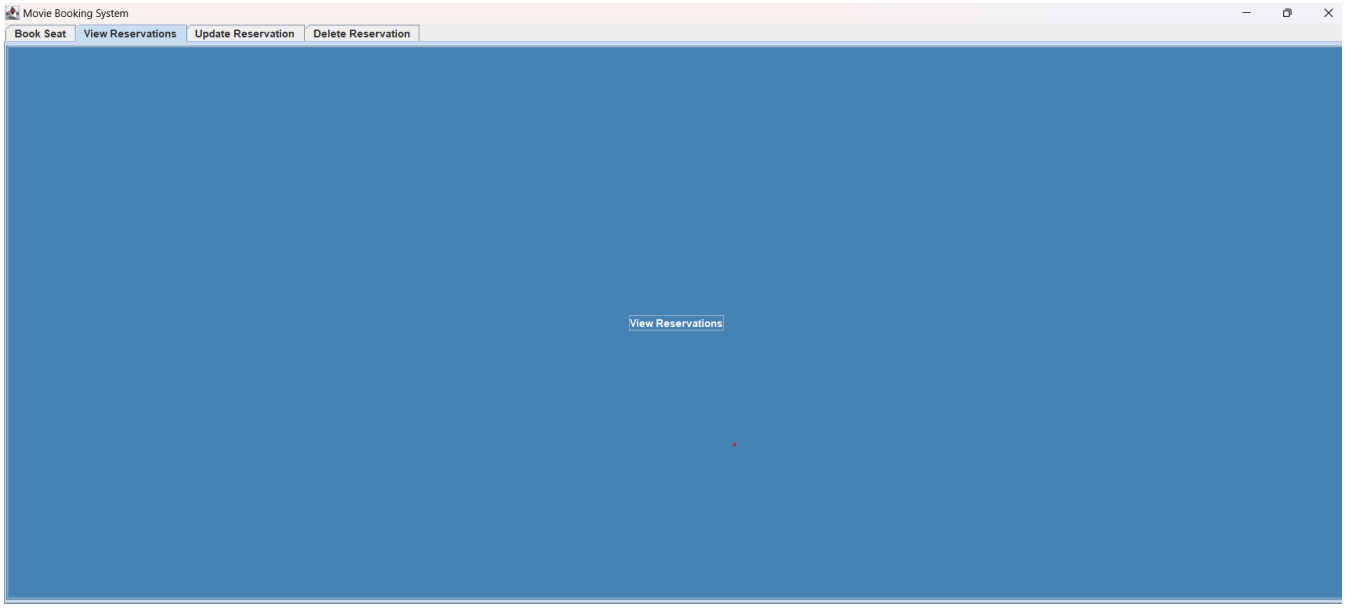
This project lays the groundwork for a scalable movie booking system, enabling automation and efficiency in reservation

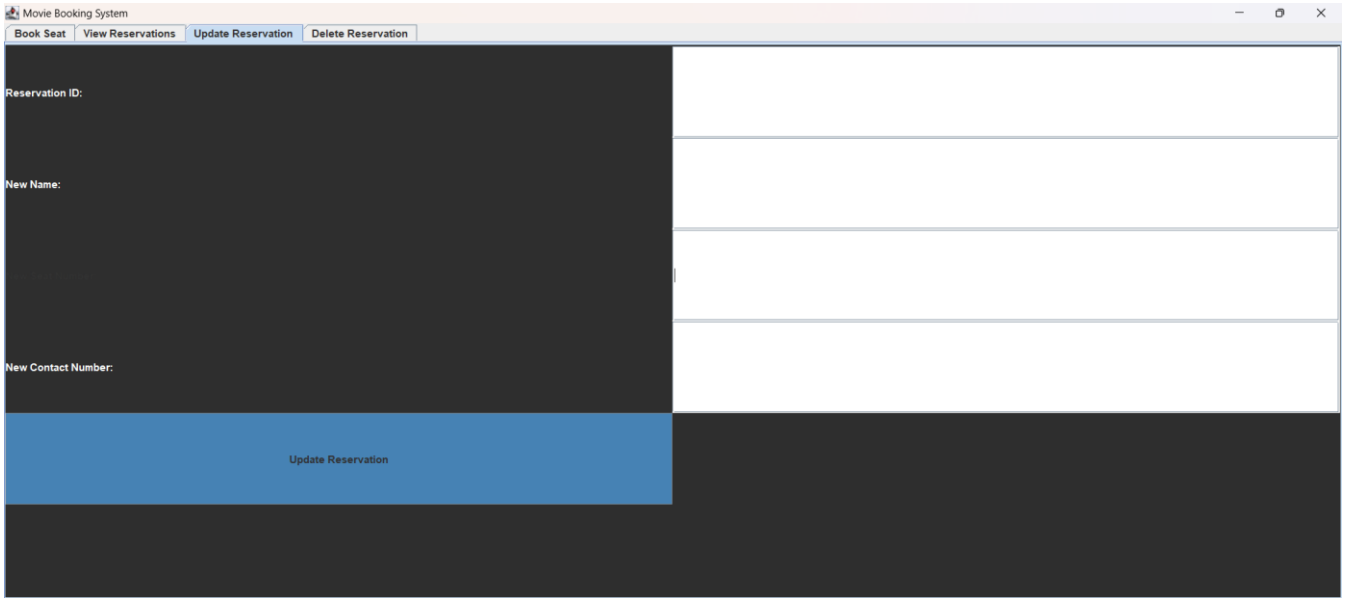
management while providing room for future growth and integration of additional features.

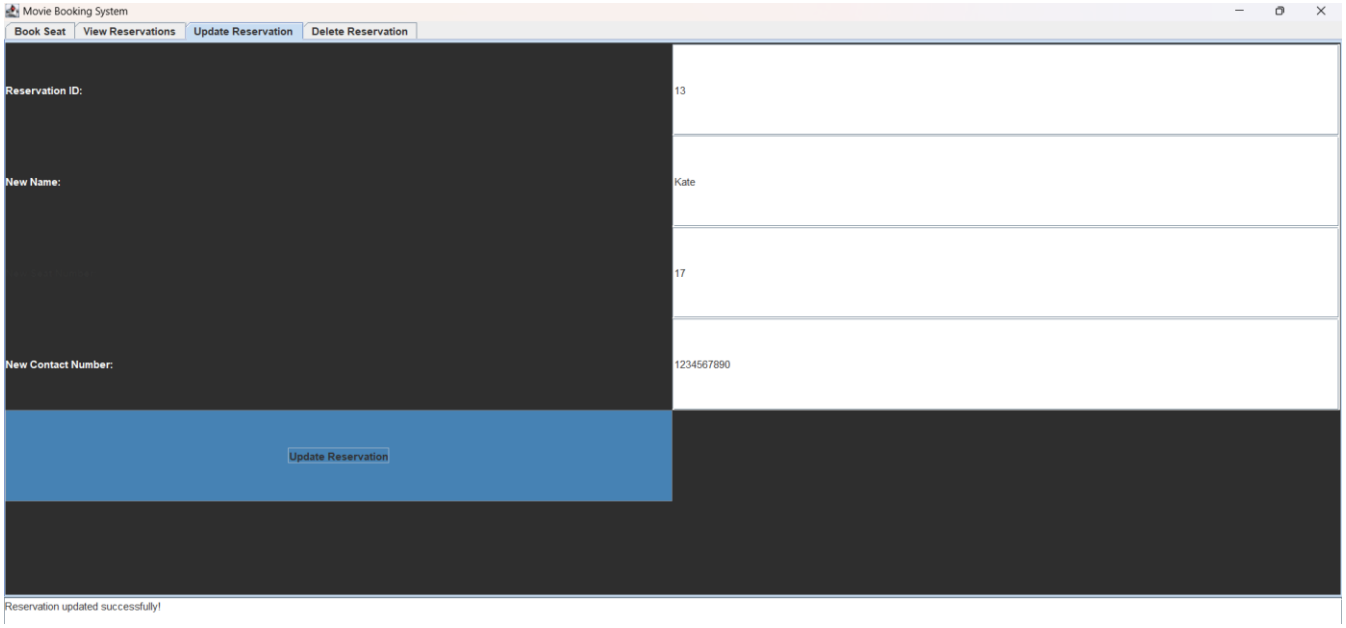
JAVA UI PICTURES

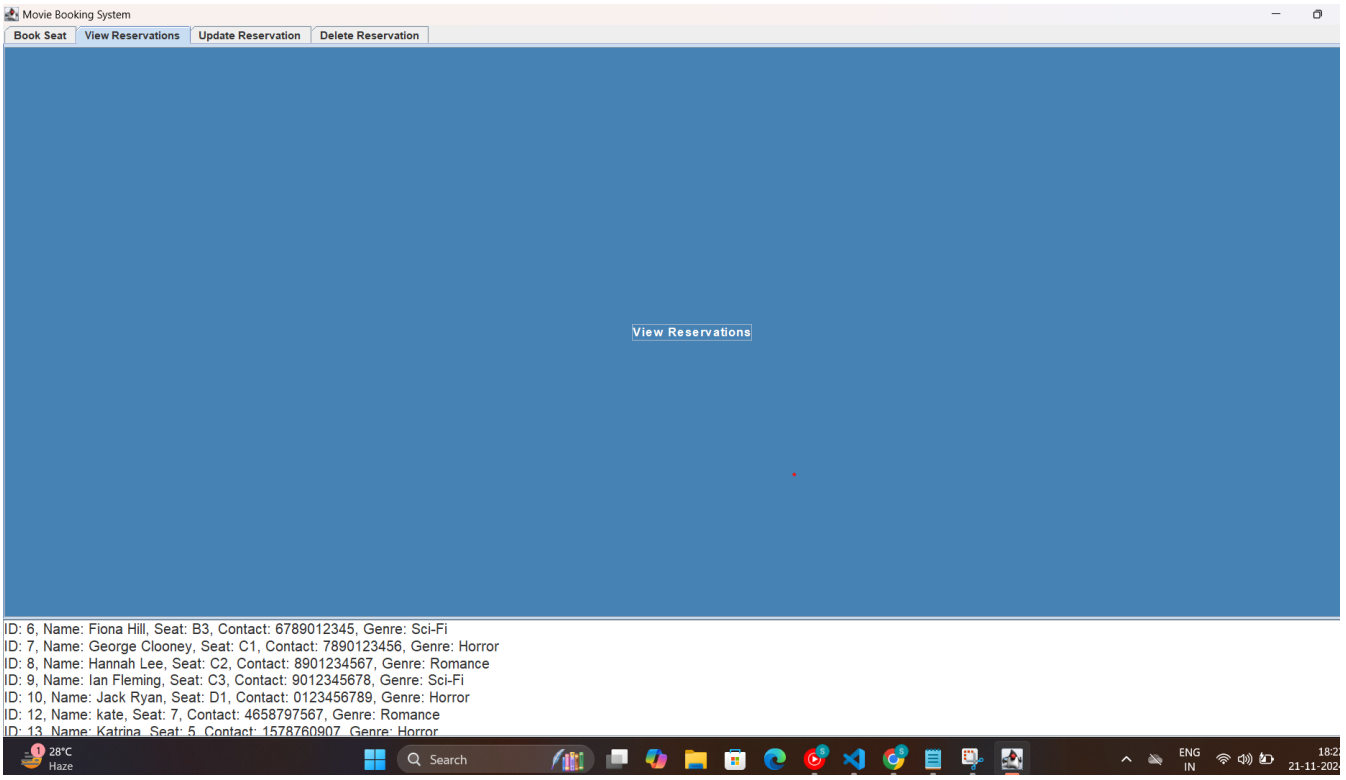


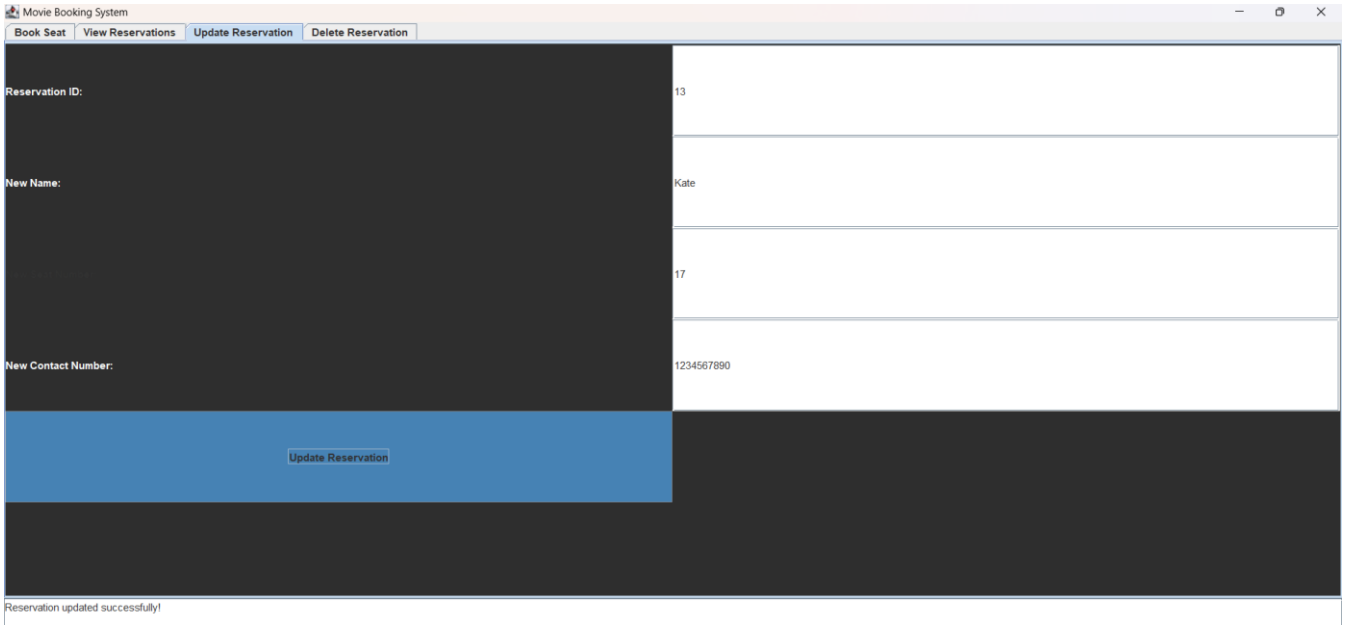

Movie Booking System
Book Seat
View Reservations
Update Reservation
Delete Reservation
Name:
SOWNDARIYA
Seat Number:
15
Contact Number:
2345678901
Genre:
Romance
Book Seat
Seat booked successfully for SOWNDARIYA (Seat Number: 15, Genre: Romance).


Movie Booking System
Book Seat
View Reservations
Update Reservation
Delete Reservation
View Reservations


Movie Booking System
Book Seat
View Reservations
Update Reservation
Delete Reservation
Reservation ID:
New Name:
New Contact Number:
Update Reservation


Movie Booking System
Book Seat
View Reservations
Update Reservation
Delete Reservation
Reservation ID:
13
New Name:
Kate
17
New Contact Number:
1234567890
Update Reservation
Reservation updated successfully!





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.

PROGRAM :

```
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);
            }
        });
    }
}
```

Conclusion

The Movie Booking System is a robust and user-friendly application designed to streamline the process of managing movie theater seat reservations. By incorporating key functionalities such as seat booking, reservation updates, and deletion, the system ensures an efficient and organized workflow for theater staff while providing a seamless experience for customers.

Through the integration of a secure login system and a centralized MySQL database, the project prioritizes data security and reliability. The intuitive Java Swing interface further enhances usability, making the application accessible to users with minimal technical expertise. With features such as real-time validation and a clear separation of functionalities, the system significantly reduces errors and manual efforts associated with traditional booking methods.

While the project addresses current operational needs effectively, it also offers ample scope for future enhancements, such as graphical seat selection, customer registration, and payment integration. These additions will not only broaden the system's capabilities but also make it suitable for larger theaters and online usage.

In conclusion, the Movie Booking System provides a strong foundation for automating and improving reservation management processes, ensuring operational efficiency and customer satisfaction. Its scalability and adaptability make it a valuable tool for modern movie theaters looking to enhance their services and embrace digital solutions.

References

1. **Java Swing Documentation**
   Oracle. "Creating a GUI With Swing." Java Platform SE Documentation, Oracle Corporation.

2. **MySQL Database Integration**
   MySQL. "MySQL 8.0 Reference Manual." MySQL Documentation, Oracle Corporation.

3. **Java JDBC Guide**
   Oracle. "Java Database Connectivity (JDBC) API." Java SE Documentation, Oracle Corporation.

4. **Event Handling in Java**
   Oracle. "How to Use Action Listeners." Java Swing Tutorial, Oracle Corporation.

5. **Best Practices for Secure Login Systems**
   OWASP. "Authentication Cheat Sheet." OWASP Foundation.

6. **Database Normalization Guidelines**
   IBM. "A Guide to Database Normalization." IBM Knowledge Center.

7. **User Experience in GUI Design**
   Nielsen, Jakob. "10 Usability Heuristics for User Interface Design." Nielsen Norman Group.

These resources were instrumental in guiding the development of the **Movie Booking System**, particularly in implementing secure and efficient database operations, creating a user-friendly GUI, and adhering to best practices for application design and usability.