**Bus Reservation System**

**PROJECT REPORT**

**CSA0961-** **Programming in Java for Android Application**

***Submitted by***

**K. L. V Jayaram**

**(192210314)**

***In partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE**



**SAVEETHA SCHOOL OF ENGINEERING, SAVEETHA NAGAR, THANDALAM,**

**SIMATS, CHENNAI – 602105.**

**JUNE-2024**

# BONAFIDE CERTIFICATE

This is to certify that the project report entitled “Bus Reservation System” submitted by “K. L. V Jayaram (192210314)”, to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of Bonafide work carried out by him/her under my guidance. The project fulfill the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

**Dr. A. Jegatheesan**

Professor

Department of Knowledge

Engineering,

Saveetha School of Engineering,

SIMATS, Chennai – 602 105

Internal examiner External Examiner

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO** |
| 1 | ABSTRACT | 4 |
| 2 | INTRODUCTION | 5 |
| 3 | DESCRIPTION | 6 |
| 4 | SOFTWARE REQUIREMENTS | 7 |
| 5 | EXISTING WORK | 8 |
| 6 | PROPOSED WORK | 9 |
| 7 | TECHNOLOGY USED | 10 |
| 8 | USE CASE DIAGRAM | 11 |
| 9 | SOURCE CODE | 12 |
| 10 | OUTPUTS (SCREENSHOTS) | 22 |
| 11 | CONCLUSION & FUTURE ENHANCEMENTS | 25 |
| 12 | REFERENCES | 26 |

# ABSTRACT

A Bus Reservation System is a digital platform designed to streamline the booking process for bus tickets. It provides users with an intuitive interface to search for available buses, select seats, and confirm bookings based on their preferred travel dates and destinations. This system not only enhances convenience by allowing bookings from anywhere at any time but also improves operational efficiency for bus operators by automating seat management and reservation handling.

Administrators benefit from tools to manage bus routes, schedules, and user accounts securely. Integrated features like payment gateways ensure secure transactions, while notifications keep users informed about booking statuses. Overall, the Bus Reservation System aims to simplify travel planning, increase accessibility, and optimize resource utilization in the transportation industry.

Moreover, the system enhances customer satisfaction by providing intuitive interfaces for booking and managing reservations, coupled with robust support for cancellation and refund processes. It adapts seamlessly to diverse user needs, supporting multiple languages and currencies, and fostering loyalty through personalized promotions and rewards. This abstract highlights how a bus reservation system not only facilitates seamless travel planning but also transforms the passenger experience through technological innovation and operational excellence.

# INTRODUCTION

A Bus Reservation System revolutionizes the way travelers plan and secure their journeys by leveraging digital technology to streamline the booking process for bus tickets. This system serves as a centralized platform where users can effortlessly search for available buses, select preferred seats, and make reservations according to their travel preferences and schedules.

With intuitive user interfaces accessible via web or mobile applications, passengers can conveniently browse through routes, check seat availability in real-time, and complete bookings with ease.

From the perspective of bus operators and transportation companies, the Bus Reservation System plays a pivotal role in enhancing operational efficiency. It automates critical tasks such as seat management, scheduling, and passenger data management, thereby reducing manual errors and optimizing resource utilization.

Administrators can efficiently oversee bus routes, monitor occupancy rates, and analyze booking trends to make informed decisions that improve service delivery and customer satisfaction.

By integrating secure payment gateways and providing seamless communication channels for notifications and updates, the system ensures a smooth and secure booking experience for travelers while driving business growth and operational effectiveness for bus operators.

# DESCRIPTION

A bus reservation system is a comprehensive software application designed to facilitate the booking and management of bus tickets for both passengers and bus operators. By integrating various features, it streamlines the entire reservation process, making it more convenient and efficient. Users can search for buses based on their origin, destination, and travel dates, view real-time seat availability, and select their preferred seats using an interactive seat map. This system significantly enhances the user experience by providing a seamless, user-friendly interface that simplifies the booking process.

One of the primary advantages of a bus reservation system is its ability to handle secure online transactions. The system typically supports multiple payment gateways, allowing users to pay for their tickets using credit or debit cards, net banking, or digital wallets. Upon successful booking, electronic tickets (e-tickets) are generated and can be downloaded or sent via email or SMS. This not only reduces the need for physical tickets but also ensures that users have easy access to their travel documents.

The bus reservation system also offers additional functionalities such as mobile app integration, multi-language and multi-currency support, and loyalty programs. These features cater to a diverse user base and enhance the overall travel experience. By incorporating customer feedback mechanisms, operators can continually improve their services based on user reviews and ratings. Overall, a bus reservation system not only improves operational efficiency for bus operators but also provides a convenient, hassle-free booking experience for passengers.

# SOFTWARE REQUIREMENTS

**User Authentication:**

The system shall require user authentication for access.

Different user roles shall have distinct privileges.

**Java Development Kit (JDK)**

The JDK is essential for compiling and running Java programs. It provides the necessary tools, libraries, and runtime environment to execute the Currency Converter application. It's crucial to have the right version of JDK installed to ensure compatibility with the application's code and dependencies.

**Java Swing Framework**

Java Swing is a powerful GUI toolkit used to develop user-friendly interfaces. It provides a wide range of components like buttons, text fields, and drop-down lists, which are essential for creating the Currency Converter's graphical user interface.

**Integrated Development Environment (IDE):**

An IDE facilitates coding, debugging, and managing Java projects. It offers features like code completion, debugging tools, and project management capabilities.

**Internet Connectivity**

The Bus Reservation System relies on real-time booking tickets. This requires a stable internet connection to access the online API that provides the latest exchange data. Without internet connectivity, the application won't be able to function effectively, and users won't be able to perform conversions.

# EXISTING WORK

* The existing work on bus reservation systems spans various implementations and technologies aimed at improving booking efficiency and passenger convenience.
* Traditional systems relied on manual booking processes at physical counters, which have gradually evolved into sophisticated digital platforms
* Modern bus reservation systems utilize web and mobile applications to allow passengers to browse schedules, select seats, and make payments online, thereby reducing queue times and enhancing accessibility.
* Technological advancements have also facilitated real-time updates on bus availability, routes, and seat occupancy, enabling travelers to make informed decisions.
* Integration with GPS and mapping technologies enables tracking of buses in real-time, providing accurate arrival and departure information to passengers.
* Additionally, these systems often incorporate features such as secure payment gateways, customer support interfaces, and loyalty programs to enhance user experience and encourage repeat bookings.

# PROPOSED WORK

* Enhanced User Experience: Introducing a more intuitive and responsive user interface that simplifies the booking process, enhances usability across different devices, and provides personalized recommendations based on user preferences and travel history.
* Proper Billing and Payment Integration: Implementing a robust billing system that ensures accurate invoicing and transparent pricing for passengers. This includes automated billing processes that generate detailed invoices and receipts.
* Ensuring billing transparency builds trust with passengers and facilitates seamless financial transactions, contributing to overall customer satisfaction and operational efficiency within the bus reservation system

# TECHNOLOGY USED

# A bus reservation system typically incorporates several technologies to provide a seamless booking experience for users and efficient management for operators. Here are some key technologies used:

# Java Swing Framework

# Java Swing is a powerful and versatile graphical user interface (GUI) toolkit that provides a comprehensive set of components for building robust and visually appealing desktop applications.

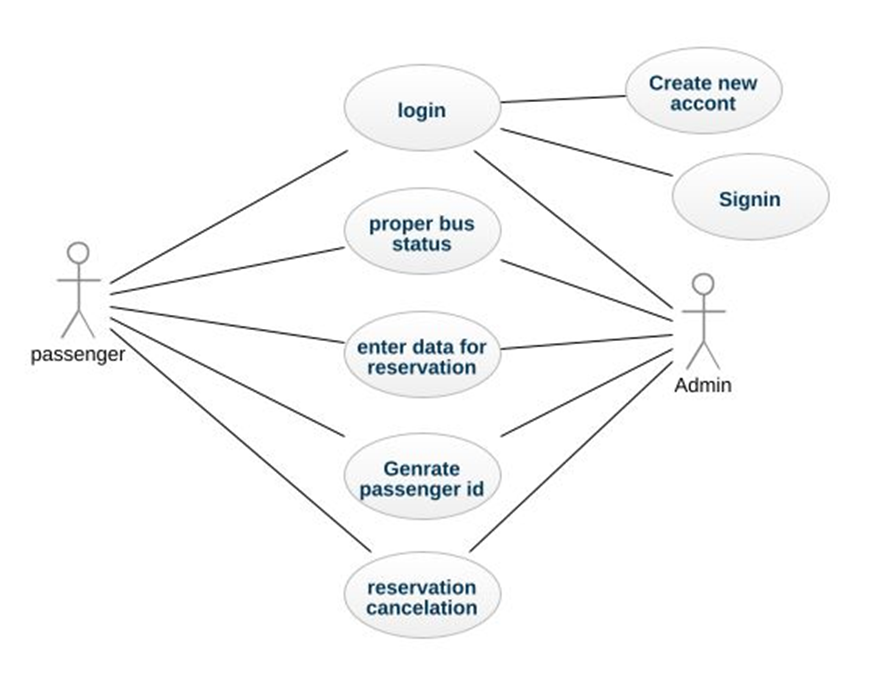
# Swing offers a wide array of widgets, including buttons, text fields, labels, scroll panes, and tables, allowing for the creation of interactive and user-friendly interfaces.

# Its platform-independent nature ensures that applications built with Swing can run seamlessly across different operating systems without requiring code modifications.

# Swing's flexibility and extensibility enable developers to customize and extend its functionality to meet specific application requirements.

* Employing a variety of Swing components such as JFrame, JPanel, and JButton, the system provides users with a visually appealing and responsive interface, facilitating easy navigation and execution of inventory management tasks.

# USE CASE DIAGRAM



**Fig 1 Use case diagram of attendance management system**

**Fig 1:** The various interactions and functionalities of the system from the perspective of different actors involved. The diagram includes two primary actors: Passenger and Admin.

Each actor is associated with specific use cases that outline the actions they can perform within the system. The admin actor is responsible for proper bus status, enter data for reservation, and generating passenger ID. They have administrative privileges to perform tasks such as adding or editing Bus status information, creating new routes, and adding new buses related to destination and source.

**SOURCE CODE**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

class King extends JFrame implements ActionListener {

private JLabel nameLabel, sourceLabel, destLabel, dateLabel, numPassengersLabel, mobileLabel, ticketPreviewLabel, seatLabel;

private JTextField nameField, numPassengersField, mobileField, seatField;

private JComboBox<String> sourceCombo, destCombo, dayCombo, monthCombo, yearCombo;

private JButton submitButton, exitButton, okButton, resetButton;

private JCheckBox returnCheckBox;

private JTextArea ticketTextArea;

private LinkedHashMap<String, Integer> sourceMap = new LinkedHashMap<>();

private LinkedHashMap<String, Integer> destMap = new LinkedHashMap<>();

private String[] days = new String[31];

private String[] months = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};

private String[] years = new String[50];

private static final Set<String> MONTHS\_WITH\_31\_DAYS = new HashSet<>(Arrays.asList("January", "March", "May", "July", "August", "October", "December"));

public King() {

setTitle("Bus Ticket Reservation");

setSize(1200, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(null);

getContentPane().setBackground(Color.LIGHT\_GRAY);

setExtendedState(JFrame.MAXIMIZED\_BOTH);

// Populate source and destination maps

initializeLocationMaps();

// Title label

addTitleLabel();

// Ticket preview components

addTicketPreviewComponents();

// Input fields and labels

addInputFieldsAndLabels();

setVisible(true);

}

private void initializeLocationMaps() {

sourceMap.put("New Delhi", 750);

sourceMap.put("Mumbai", 600);

sourceMap.put("Chennai", 350);

sourceMap.put("Kolkata", 700);

sourceMap.put("Bengaluru", 450);

sourceMap.put("Pune", 550);

destMap.put("New Delhi", 750);

destMap.put("Mumbai", 600);

destMap.put("Chennai", 350);

destMap.put("Kolkata", 700);

destMap.put("Bengaluru", 450);

destMap.put("Pune", 550);

}

private void addTitleLabel() {

JLabel titleLabel = new JLabel("Bus Ticket Reservation");

titleLabel.setFont(new Font("Serif", Font.BOLD, 60));

titleLabel.setForeground(Color.BLUE);

titleLabel.setBounds(100, 10, 800, 100);

add(titleLabel);

}

private void addTicketPreviewComponents() {

ticketPreviewLabel = new JLabel("Ticket Preview");

ticketPreviewLabel.setFont(new Font("Serif", Font.BOLD, 36));

ticketPreviewLabel.setBounds(850, 30, 300, 50);

add(ticketPreviewLabel);

ticketTextArea = new JTextArea();

ticketTextArea.setEditable(false);

ticketTextArea.setFont(new Font("Serif", Font.PLAIN, 20));

JScrollPane scrollPane = new JScrollPane(ticketTextArea);

scrollPane.setBounds(850, 100, 300, 400);

add(scrollPane);

}

private void addInputFieldsAndLabels() {

addLabelAndField("Passenger Name:", 120, nameField = new JTextField());

addLabelAndComboBox("Source:", 170, sourceCombo = new JComboBox<>(sourceMap.keySet().toArray(new String[0])));

addLabelAndComboBox("Destination:", 220, destCombo = new JComboBox<>(destMap.keySet().toArray(new String[0])));

addLabelAndField("Mobile Number:", 270, mobileField = new JTextField());

addLabelAndDateComboBoxes();

addLabelAndField("Number of Passengers:", 370, numPassengersField = new JTextField());

returnCheckBox = new JCheckBox("Return Journey");

returnCheckBox.setFont(new Font("Serif", Font.PLAIN, 20));

returnCheckBox.setBounds(20, 420, 300, 30);

add(returnCheckBox);

addLabelAndField("Seat Number:", 470, seatField = new JTextField());

addButtons();

}

private void addLabelAndField(String labelText, int yPos, JTextField textField) {

JLabel label = new JLabel(labelText);

label.setFont(new Font("Serif", Font.PLAIN, 20));

label.setBounds(20, yPos, 200, 30);

add(label);

textField.setFont(new Font("Serif", Font.PLAIN, 20));

textField.setBounds(250, yPos, 300, 30);

add(textField);

}

private void addLabelAndComboBox(String labelText, int yPos, JComboBox<String> comboBox) {

JLabel label = new JLabel(labelText);

label.setFont(new Font("Serif", Font.PLAIN, 20));

label.setBounds(20, yPos, 200, 30);

add(label);

comboBox.setFont(new Font("Serif", Font.PLAIN, 20));

comboBox.setBounds(250, yPos, 300, 30);

add(comboBox);

}

private void addLabelAndDateComboBoxes() {

dateLabel = new JLabel("Travel Date:");

dateLabel.setFont(new Font("Serif", Font.PLAIN, 20));

dateLabel.setBounds(20, 320, 200, 30);

add(dateLabel);

for (int i = 0; i < 31; i++) {

days[i] = Integer.toString(i + 1);

}

dayCombo = new JComboBox<>(days);

dayCombo.setFont(new Font("Serif", Font.PLAIN, 20));

dayCombo.setBounds(250, 320, 60, 30);

add(dayCombo);

monthCombo = new JComboBox<>(months);

monthCombo.setFont(new Font("Serif", Font.PLAIN, 20));

monthCombo.setBounds(320, 320, 120, 30);

monthCombo.addActionListener(this);

add(monthCombo);

Calendar now = Calendar.getInstance();

int year = now.get(Calendar.YEAR);

for (int i = 0; i < 50; i++) {

years[i] = Integer.toString(year + i);

}

yearCombo = new JComboBox<>(years);

yearCombo.setFont(new Font("Serif", Font.PLAIN, 20));

yearCombo.setBounds(450, 320, 100, 30);

add(yearCombo);

}

private void addButtons() {

submitButton = new JButton("Submit");

submitButton.setFont(new Font("Serif", Font.PLAIN, 20));

submitButton.setBounds(50, 530, 150, 40);

submitButton.addActionListener(this);

add(submitButton);

exitButton = new JButton("Exit");

exitButton.setFont(new Font("Serif", Font.PLAIN, 20));

exitButton.setBounds(250, 530, 150, 40);

exitButton.addActionListener(this);

add(exitButton);

okButton = new JButton("OK");

okButton.setFont(new Font("Serif", Font.PLAIN, 20));

okButton.setBounds(850, 520, 100, 40);

okButton.addActionListener(this);

add(okButton);

resetButton = new JButton("Reset");

resetButton.setFont(new Font("Serif", Font.PLAIN, 20));

resetButton.setBounds(970, 520, 100, 40);

resetButton.addActionListener(this);

add(resetButton);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == submitButton) {

handleSubmit();

} else if (e.getSource() == resetButton) {

resetFields();

} else if (e.getSource() == exitButton) {

dispose();

} else if (e.getSource() == okButton) {

printTicketFromPreview();

}

}

private void handleSubmit() {

try {

String name = nameField.getText();

String source = (String) sourceCombo.getSelectedItem();

String dest = (String) destCombo.getSelectedItem();

String mobileNumber = mobileField.getText();

String day = (String) dayCombo.getSelectedItem();

String month = (String) monthCombo.getSelectedItem();

String year = (String) yearCombo.getSelectedItem();

int numPassengers = Integer.parseInt(numPassengersField.getText());

String seatNumber = seatField.getText();

validateInputs(name, source, dest, mobileNumber, day, month, year, numPassengers, seatNumber);

String travelDate = day + " " + month + " " + year;

int distance = Math.abs(sourceMap.get(source) - destMap.get(dest));

float cost = calculateCost(distance, numPassengers);

if (returnCheckBox.isSelected()) {

cost \*= 2; // Double the cost for return journey

}

printTicket(name, source, dest, travelDate, mobileNumber, numPassengers, seatNumber, cost);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Please enter valid numeric values for number of passengers.", "Invalid Input", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void validateInputs(String name, String source, String dest, String mobileNumber, String day, String month, String year, int numPassengers, String seatNumber) throws Exception {

if (name.isEmpty() || source.isEmpty() || dest.isEmpty() || mobileNumber.isEmpty() || day.isEmpty() || month.isEmpty() || year.isEmpty() || seatNumber.isEmpty()) {

throw new Exception("All fields are mandatory.");

}

if (mobileNumber.length() < 10) {

throw new Exception("Mobile number must be at least 10 digits.");

}

Calendar selectedDate = Calendar.getInstance();

selectedDate.set(Integer.parseInt(year), Arrays.asList(months).indexOf(month), Integer.parseInt(day));

if (selectedDate.before(Calendar.getInstance())) {

throw new Exception("Travel date cannot be earlier than the current date.");

}

if (day.equals("31") && !MONTHS\_WITH\_31\_DAYS.contains(month)) {

throw new Exception("Invalid day selected for " + month + ".");

}

if (source.equals(dest)) {

throw new Exception("Source and Destination cannot be the same.");

}

}

private float calculateCost(int distance, int numPassengers) {

float ratePerKm = 5.0f; // Assume a rate per kilometer

return ratePerKm \* distance \* numPassengers;

}

private void printTicket(String name, String source, String dest, String travelDate, String mobileNumber, int numSeats, String seatNumber, float cost) {

String ticketDetails = "Passenger Name: " + name + "\nSource: " + source + "\nDestination: " + dest +

"\nTravel Date: " + travelDate + "\nMobile Number: " + mobileNumber + "\nNumber of Passengers: " + numSeats +

"\nSeat Number: " + seatNumber + "\nTotal Cost: Rs. " + cost;

ticketTextArea.setText(ticketDetails);

}

private void resetFields() {

nameField.setText("");

sourceCombo.setSelectedIndex(0);

destCombo.setSelectedIndex(0);

mobileField.setText("");

dayCombo.setSelectedIndex(0);

monthCombo.setSelectedIndex(0);

yearCombo.setSelectedIndex(0);

numPassengersField.setText("");

returnCheckBox.setSelected(false);

seatField.setText("");

ticketTextArea.setText("");

}

private void printTicketFromPreview() {

String ticketDetails = ticketTextArea.getText();

if (!ticketDetails.isEmpty()) {

JOptionPane.showMessageDialog(this, ticketDetails, "Print Ticket", JOptionPane.PLAIN\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "No ticket to print. Generate a ticket first.", "Print Ticket", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(LoginFrame::new);

}

}

class LoginFrame extends JFrame implements ActionListener {

private JTextField usernameField;

private JPasswordField passwordField;

public LoginFrame() {

setTitle("Login");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(null);

addTitleLabel();

addUsernameAndPasswordFields();

addLoginButton();

setVisible(true);

}

private void addTitleLabel() {

JLabel titleLabel = new JLabel("Bus Reservation System Login");

titleLabel.setFont(new Font("Serif", Font.BOLD, 24));

titleLabel.setBounds(50, 20, 300, 30);

add(titleLabel);

}

private void addUsernameAndPasswordFields() {

addLabelAndField("Username:", 80, usernameField = new JTextField());

addLabelAndField("Password:", 130, passwordField = new JPasswordField());

}

private void addLabelAndField(String labelText, int yPos, JTextField textField) {

JLabel label = new JLabel(labelText);

label.setBounds(50, yPos, 80, 30);

add(label);

textField.setBounds(140, yPos, 200, 30);

add(textField);

}

private void addLoginButton() {

JButton loginButton = new JButton("Login");

loginButton.setBounds(150, 200, 100, 40);

loginButton.addActionListener(this);

add(loginButton);

}

@Override

public void actionPerformed(ActionEvent e) {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

// Check login credentials (for demonstration purposes)

if (username.equals("KING") && password.equals("0314")) {

JOptionPane.showMessageDialog(this, "Login successful!", "Success", JOptionPane.INFORMATION\_MESSAGE);

new King();

dispose(); // Close the login window

} else {

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

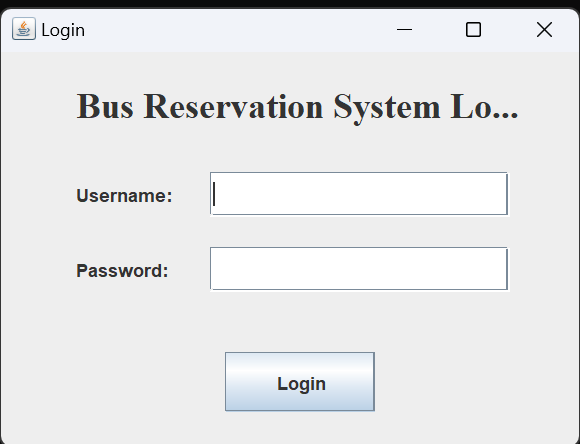
}

}

}

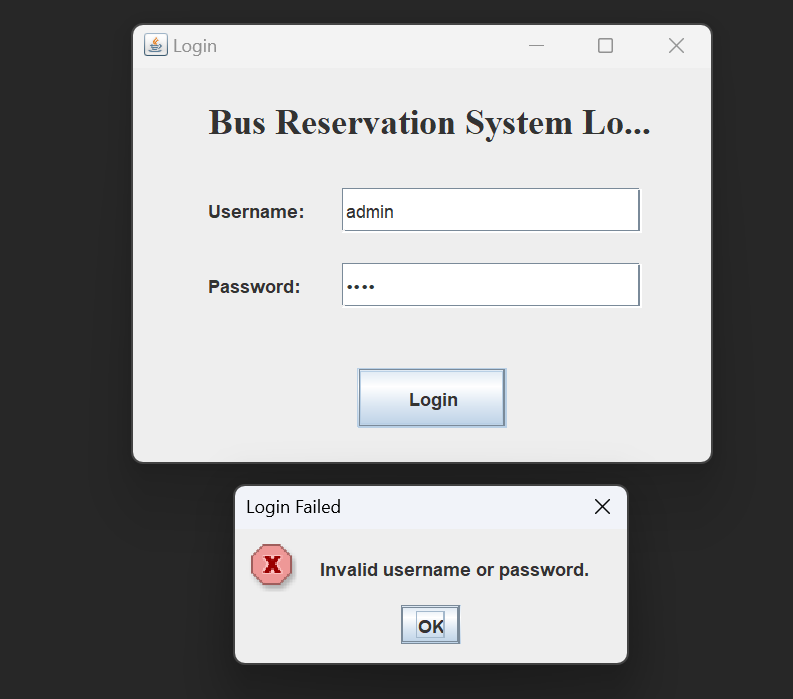
**OUTPUT**

**Fig 2:** The interface of an attendance app typically features a login screen with fields for entering the username and password, ensuring secure and personalized access. The username field allows users to input their unique identification, often assigned by the system administrator.

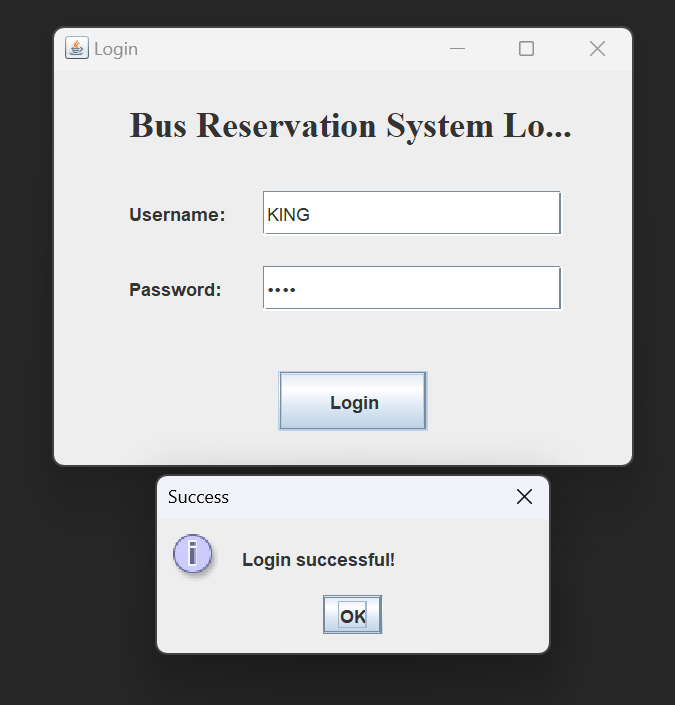


**Fig 2 Login Page**

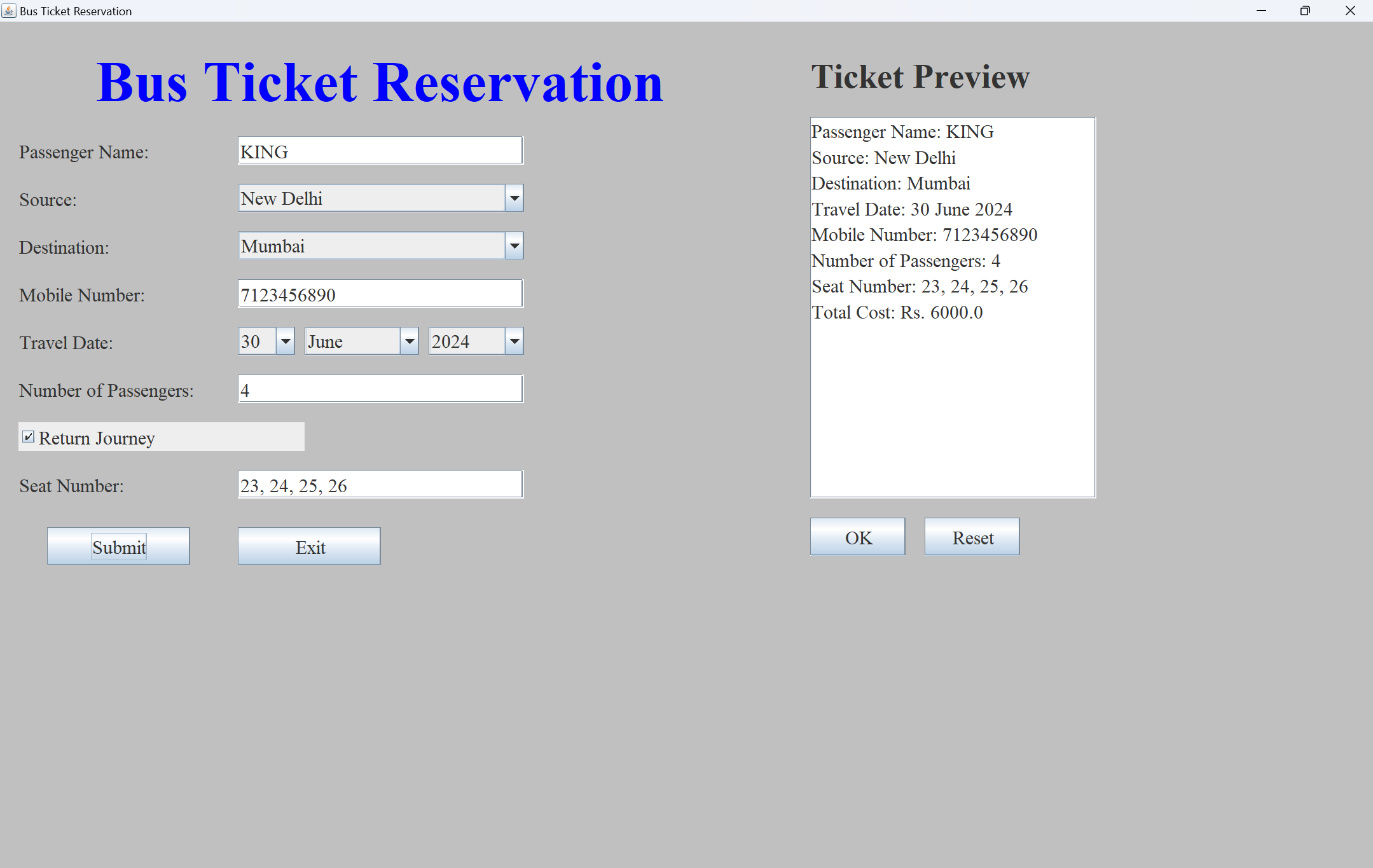
**Fig 3:** It denies the other users to enter.



**Fig 4:** Login Successful for valid user

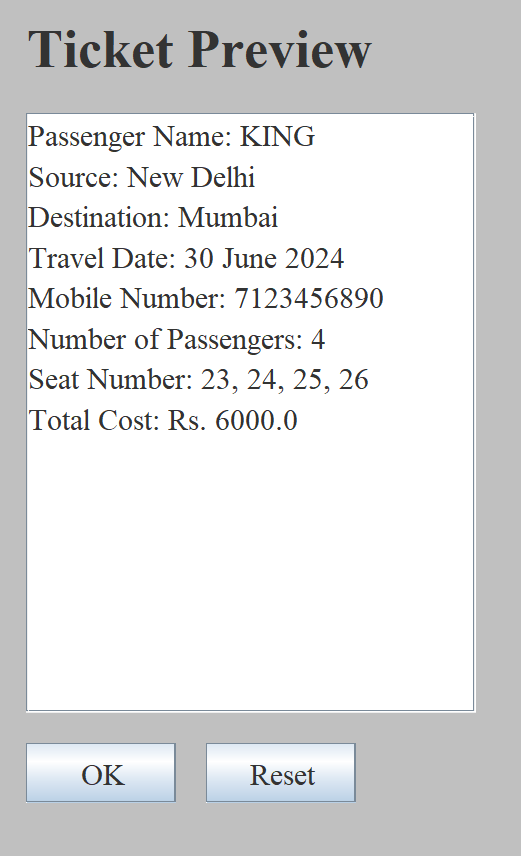


**Fig 5:** Created interface to add name and mobile number. And to select the source and destination.

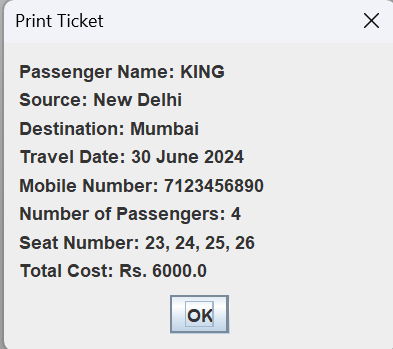


**Fig 5 Bus Reservation System**

**Fig 6:** Ticket Preview



**Fig 7:** Ticket



# CONCLUSION

* In conclusion, the bus reservation system developed in Java Swing offers a robust solution for booking and managing bus tickets through a user-friendly graphical interface. It supports essential functionalities such as input validation for ensuring data integrity, cost calculation based on distance and passenger count, and a convenient ticket preview before final submission.
* The system enhances user experience with error handling for invalid inputs and includes features like ticket printing upon confirmation. While the current implementation demonstrates core functionalities effectively, future iterations could explore enhancements such as database integration for persistent data storage and further refinement of the user interface to optimize usability and accessibility

# FUTURE ENHANCEMENT

* + Database Integration: Implementing a database backend (like MySQL or SQLite) would enable the system to store booking details persistently. This would facilitate features such as booking history retrieval, analytics on popular routes, and better management of passenger data.
  + Advanced Authentication and Security: Enhance security measures by implementing more robust authentication methods, such as user roles and permissions management. This could include secure login protocols, encryption of sensitive data, and protection against common security threats like SQL injection.
  + Real-time Updates and Notifications: Incorporate real-time updates and notifications to inform users about booking confirmations, seat availability changes, and travel updates. This could be achieved through integration with messaging services or email notifications, enhancing communication between passengers and the reservation system.

**REFERENCES**

* A. R. Mohammed and S. S. Kassem, "UML Modeling of Online Public Bus Reservation System in Egypt," <https://ieeexplore.ieee.org/document/9325604/metrics#metrics>
* M. Vlasakis, I. Keramidi, I. Moscholios, P. Sarigiannidis and M. Logothetis, "Call Admission Control under a Probabilistic Bandwidth Reservation Policy and Handover Queueing in Mobile Hotspots," <https://ieeexplore.ieee.org/document/9907974>
* P. Wu, F. Chu, A. Che and Q. Shi, "A bus lane reservation problem in urban bus transit network," <https://ieeexplore.ieee.org/document/6958149>
* Y. H. Choi, S. Seo, S. Kang and B. Lee, "Justification of Effective Reactive Power Reserves With Respect to a Particular Bus Using Linear Sensitivity," <https://ieeexplore.ieee.org/document/5783533>.
* P. Wu, A. Che and F. Chu, "A bi-objective model for bus transit network and lane reservation integrated optimization," <https://ieeexplore.ieee.org/document/8948119>
* S. Zhang, X. Zhang and P. Wu, "Optimal dynamic bus lane reservation via bi-level programming," <https://ieeexplore.ieee.org/document/10004089>