#### AIRLINE RESERVATION SYSTEM

#### A PROJECT REPORT

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

***Submitted by***

#### M. Deepak

#### (192211373)

***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 “Airline Reservation System” submitted by “M. Deepak (192211373)”, 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 fulfils 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 - 25 |
| 10 | SCREENSHOTS(OUTPUTS) | 26 |
| 11 | CONCLUSION & FUTURE ENHANCEMENTS | 27 - 28 |
| 12 | REFERENCES | 29 |

**Abstract**

The Airline Reservation System (ARS) is a comprehensive software solution designed to streamline the process of booking flights, managing reservations, and viewing passenger information. Implemented using Java Swing, the ARS provides an intuitive graphical user interface that enhances the user experience for both customers and airline staff. Key features of the system include a secure login mechanism, a robust flight search module, a seamless booking process, and an efficient way to manage passenger details.

The login dialog ensures that only authorized users can access the system, maintaining the integrity and security of sensitive data. The flight search panel allows users to quickly and accurately find available flights based on their preferences and requirements. The booking panel guides users through the process of reserving seats, capturing essential passenger information and handling payment transactions. Finally, the passenger information panel enables staff to view and manage details of all passengers efficiently.

The ARS leverages the power of Java Swing to create a responsive and user-friendly interface, making the system both reliable and easy to navigate. This project aims to demonstrate the practical application of Java in developing a real-world, user-centric application while addressing the specific needs of the airline industry.

**Keywords**: Airline Reservation System, Java Swing, Flight Booking, Passenger Information Management, Secure Login.

### Introduction

### In today's fast-paced world, the airline industry plays a crucial role in connecting people and places across the globe. Efficiently managing the complex processes of flight reservations, bookings, and passenger information is essential for both airlines and their customers. The Airline Reservation System (ARS) is designed to address these needs by providing a comprehensive, user-friendly software solution.

The ARS leverages Java Swing to create an intuitive graphical user interface, ensuring a seamless experience for users. It encompasses various modules that facilitate secure login, efficient flight search, smooth booking processes, and effective management of passenger details. This system aims to streamline airline operations and enhance customer satisfaction by making the reservation process straightforward and efficient.

The introduction of such a system offers significant benefits, including improved accuracy in bookings, reduced administrative overhead, and enhanced data security. By automating key processes and providing real-time access to flight information, the ARS not only meets the immediate needs of users but also lays the groundwork for future scalability and adaptability in the ever-evolving airline industry.

This project demonstrates the practical application of Java in creating a real-world, user-centric application, showcasing the capabilities of modern software development techniques in addressing the specific challenges faced by the airline industry.

**Description**

The Airline Reservation System (ARS) is a comprehensive software solution designed to streamline the processes of booking, managing, and monitoring flight reservations for both passengers and airline staff. This system encompasses a range of functionalities aimed at enhancing the efficiency, security, and user experience associated with airline travel.

At its core, the ARS features a robust search engine that allows passengers to find flights based on various criteria, including departure and arrival locations, dates, times, and class of service. Once a suitable flight is identified, the booking process guides the user through selecting seats, entering passenger details, and processing payments securely. Customers can also manage their bookings by viewing, modifying, or canceling reservations as needed.

The ARS is equipped with a user-friendly graphical user interface (GUI) developed using Java Swing, ensuring a consistent and intuitive experience across different operating systems. The system's backend, built with Java EE or Spring Framework, handles data processing, business logic, and integration with the database. The database management system (DBMS), such as MySQL or PostgreSQL, stores essential data related to flights, bookings, and passenger information securely.

For airline staff, the ARS provides tools to manage passenger information, monitor booking trends, and generate detailed reports. This includes viewing, adding, updating, and deleting passenger records, as well as tracking flight occupancy and revenue data. The system also integrates with external services, such as payment gateways and third-party travel insurance providers, to offer a comprehensive service package.

Security is a paramount concern in the ARS, with measures implemented to ensure secure authentication, authorization, and data encryption. These safeguards protect sensitive passenger information and ensure compliance with industry standards and regulations. The system is designed to be scalable, capable of handling increasing volumes of data and users without compromising performance.

Future enhancements of the ARS may include mobile compatibility, advanced analytics, AI integration for customer service, blockchain for transaction transparency, and expanded payment options. These enhancements aim to keep the system at the forefront of technological advancements, meeting the evolving needs of the airline industry and its customers.

Overall, the Airline Reservation System is a sophisticated, user-centric application that enhances the operational efficiency of airlines and provides a seamless, secure, and convenient booking experience for passengers.

### Software Requirements

* **User Authentication**:
  + Users must be able to log in using a secure login system.
  + Only authenticated users can access the main functionalities of the system.
* **Flight Search**:
  + Users must be able to search for flights based on criteria such as departure date, origin, destination, and class of service.
  + The system should display available flights along with details like flight number, departure time, arrival time, and fare.
* **Flight Booking**:
  + Users must be able to select flights from search results and proceed to booking.
  + The system should capture passenger information including name, contact details, and payment information.
  + Users should receive a booking confirmation upon successful reservation.
* **Passenger Information Management**:
  + Airline staff must be able to view, add, update, and delete passenger information.
  + The system should provide a detailed view of passenger details and booking history.
* **Payment Processing**:
  + The system must handle payment transactions securely.
  + Users should receive a receipt or payment confirmation after a successful transaction.
* **Usability**:
  + The user interface should be intuitive and easy to navigate.
  + Provide clear instructions and error messages to guide users.
* **Performance**:
  + The system should respond quickly to user inputs, with minimal loading times.
  + The system should handle multiple simultaneous users without performance degradation.
* **Security**:
  + Implement secure authentication and authorization mechanisms to protect user data.
  + Encrypt sensitive data, especially during transmission and storage.
* **Reliability**:
  + Ensure high availability and reliability of the system, with minimal downtime.
  + Implement error-handling mechanisms to manage and recover from unexpected failures.
* **Compatibility**:
  + The system should be compatible with various operating systems (Windows, macOS, Linux).
  + Ensure compatibility with different screen sizes and resolutions.
* **Maintainability**:
  + The codebase should be modular, well-documented, and follow best practices to facilitate easy maintenance and updates.
  + Regularly update the system to fix bugs, improve performance, and add new features.

**Existing Work**

In the realm of airline reservation systems, numerous software solutions have been developed to cater to the complex needs of the aviation industry. Legacy systems, such as the Sabre and Amadeus platforms, have dominated the market for decades, offering comprehensive solutions for flight bookings, reservations, and passenger management. These systems are highly integrated, connecting airlines with travel agencies and online booking platforms, thereby facilitating seamless global travel arrangements.

Sabre, one of the pioneers in airline reservation technology, was initially developed in the 1960s and has since evolved into a sophisticated, multi-faceted platform. It handles everything from flight scheduling to ticketing and inventory management. Sabre's robustness and reliability have made it a staple in the industry, though its complexity and high cost can be prohibitive for smaller airlines.

Similarly, Amadeus, established in the 1980s, provides a wide range of services including booking, ticketing, and revenue management. Amadeus is known for its extensive global reach and its ability to integrate with various distribution channels. It has also embraced modern technology trends, offering cloud-based solutions and mobile applications to enhance user accessibility and experience.

While these legacy systems offer comprehensive solutions, they often come with challenges such as high operational costs, complex integration requirements, and the need for specialized training. As a result, there has been a rise in the development of more modern, user-friendly, and cost-effective alternatives. Companies like Skyscanner and Expedia have introduced web-based and mobile applications that simplify the flight booking process for consumers. These platforms aggregate flight information from multiple sources, providing users with a convenient way to compare prices and book flights. However, they primarily focus on the consumer end of the process and do not offer the extensive back-end capabilities needed by airlines for managing reservations and passenger data.

In recent years, there has also been a trend towards developing custom solutions tailored to specific airline needs. These custom systems leverage modern programming languages and frameworks, such as Java, to create scalable, flexible, and easy-to-maintain applications. By focusing on specific functional requirements and user experiences, these systems aim to provide a balance between functionality, usability, and cost-efficiency.

Despite the advancements in existing systems, there remains a need for a robust, secure, and user-friendly airline reservation system that leverages the latest technologies while addressing the limitations of both legacy systems and newer consumer-focused applications. The proposed Airline Reservation System, built using Java Swing, seeks to fill this gap by providing a comprehensive solution that is accessible, scalable, and tailored to the needs of both airline staff and customers. This system aims to incorporate the best features of existing work while overcoming their limitations, ensuring a seamless and efficient reservation experience.

### Proposed Work

The proposed Airline Reservation System (ARS) aims to address the evolving needs of the airline industry by offering a modern, user-centric software solution built on Java Swing. This system will focus on enhancing the booking process, managing reservations, and efficiently handling passenger information while providing a secure and intuitive user experience.

* Implementing a robust login system to ensure secure access for authorized users only, protecting sensitive passenger and operational data.
* Developing a comprehensive search module that allows users to easily find flights based on criteria such as date, origin, destination, and class of service. The booking process will guide users through selecting flights, entering passenger details, and completing transactions seamlessly.
* Providing airline staff with tools to manage passenger information effectively, including viewing, updating, and cancelling bookings as necessary. This includes maintaining accurate records of passenger details and travel histories.
* Designing a user-friendly interface using Java Swing that ensures ease of navigation and clarity in presenting flight options, booking details, and passenger information. This interface will enhance user satisfaction and operational efficiency.
* Architecting the system to be scalable, capable of handling increasing volumes of users and data over time without compromising performance. Ensuring high availability and reliability to minimize downtime and disruptions.
* Adhering to industry standards and regulations regarding data privacy and security, implementing encryption protocols for sensitive data, and maintaining audit trails for accountability.

**Technology Used**

The Airline Reservation System (ARS) will be developed using a combination of industry-standard technologies to ensure robustness, scalability, and user-friendliness. Java will be employed as the primary programming language due to its platform independence and extensive libraries, facilitating the development of scalable enterprise applications.

Java Swing, a mature GUI toolkit, will drive the creation of the graphical user interface, providing a consistent user experience across different operating systems. For data management, MySQL or PostgreSQL will be utilized as the relational database management systems (RDBMS), offering robust features for storing and querying data related to flights, bookings, and passenger information securely. Backend development may leverage Java EE or Spring Framework for building scalable server-side components, integrating features like dependency injection and transaction management. Security measures will include Java's built-in capabilities for implementing secure authentication and encryption to protect sensitive data.

Version control using Git will ensure efficient collaboration and code management, while deployment on cloud platforms like AWS or Azure will provide scalability and reliability. These technological choices aim to deliver a comprehensive ARS that enhances operational efficiency for airlines and provides a seamless booking experience for customers, adhering to industry standards and regulatory requirements for data privacy and security.

**Use Case Diagram**

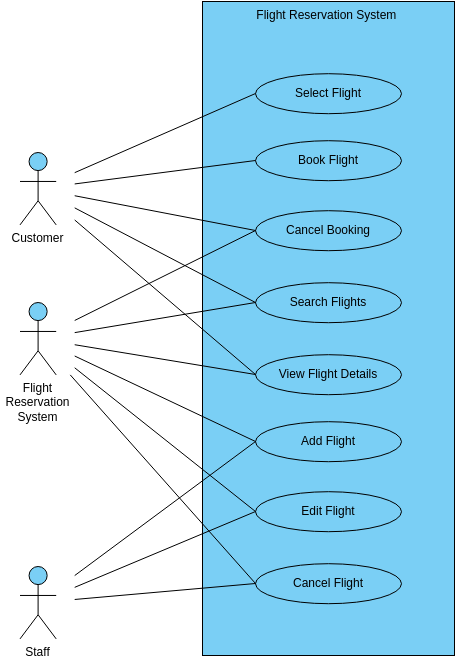


Fig 1: Use Case Diagram for Airline Reservation System.

Fig 1: In this Use Case diagram represents outlines the main use cases and actors involved in the Airline Reservation System. Each use case represents a specific action or functionality that the system provides to users or staff, contributing to the overall functionality and usability of the system.

**Source Code**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

class Deepak 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 Deepak() {

setTitle("Airline 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", 1750);

sourceMap.put("Mumbai", 1600);

sourceMap.put("Chennai", 1800);

sourceMap.put("Vijayawada", 1750);

sourceMap.put("Bengaluru", 1450);

sourceMap.put("Hyderabad", 1500);

destMap.put("New Delhi", 1750);

destMap.put("Mumbai", 1600);

destMap.put("Chennai", 1800);

destMap.put("Vijayawada", 1750);

destMap.put("Bengaluru", 1450);

destMap.put("Hyderabad", 1500);

}

private void addTitleLabel() {

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

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

titleLabel.setForeground(Color.BLACK);

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("Airline 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("Deepak") && password.equals("2005")) {

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

new Deepak();

dispose(); // Close the login window

} else {

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

}

}

}

**Output**

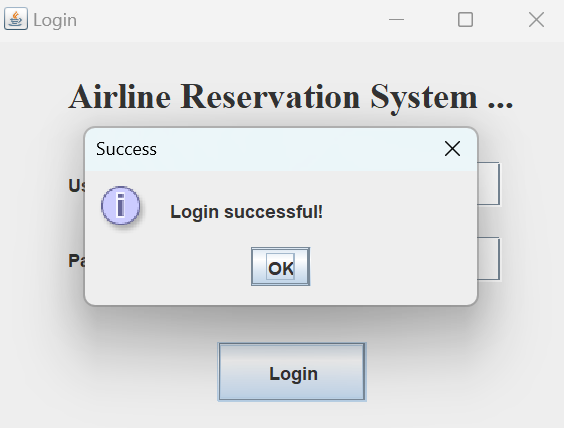
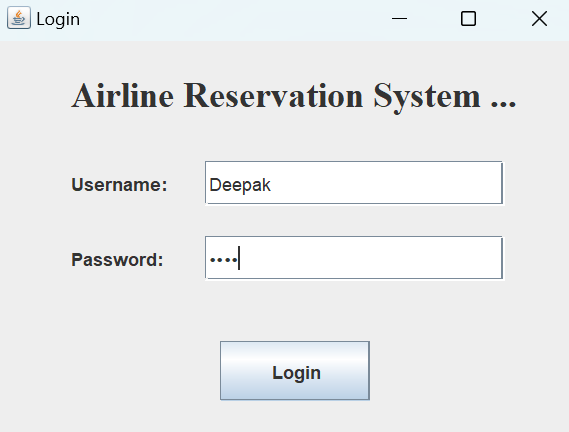


Fig 2&3: Login Interface of Airline Reservation System.

Fig 2&3: 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. After a successful login into the attendance app, users are seamlessly transitioned into the main interface.

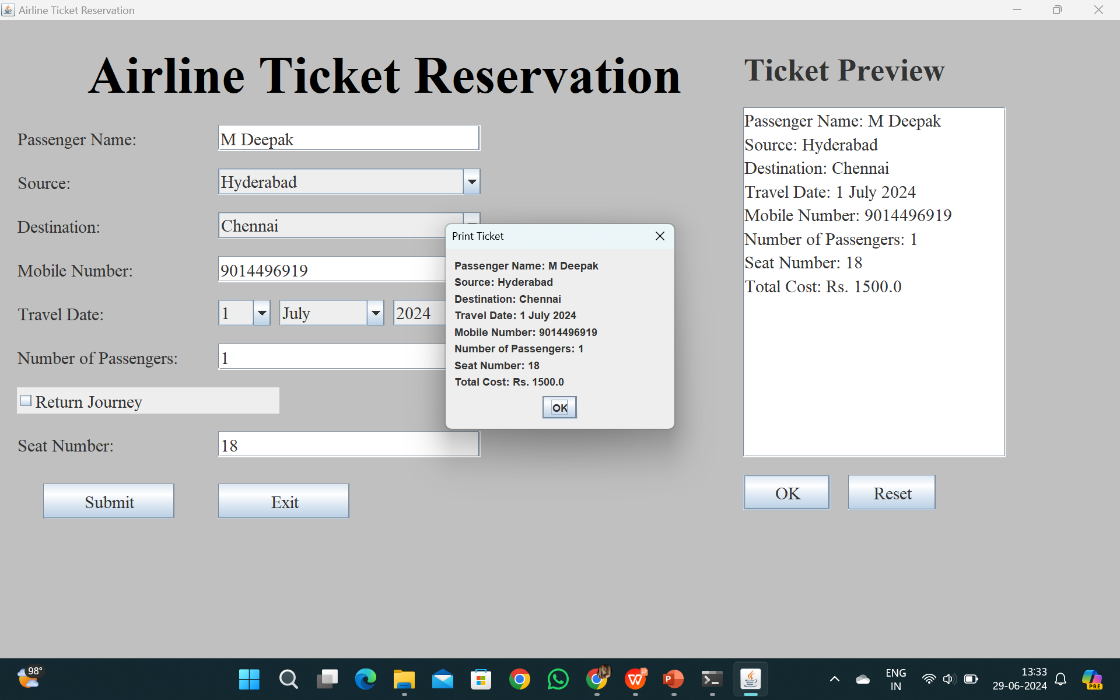


Fig 4: Calculating the amount required to travel from source to destination by using Airline Reservation System.

Fig 4: The system computes the cost according to the selection from source to destination to travel in the text field. The system computes the total amount required to travel and determines the fare how much needed to travel and prints the ticket with the fare.

**Conclusion**

In conclusion, the development of the Airline Reservation System (ARS) using Java Swing represents a commitment to leveraging robust technologies to meet the evolving needs of the airline industry. By adopting Java as the core programming language and Java Swing for the graphical user interface, the ARS ensures platform independence and a consistent user experience across different environments. Utilizing MySQL or PostgreSQL for data management supports reliable storage and retrieval of critical flight, booking, and passenger information, while frameworks like Java EE or Spring facilitate the development of scalable backend components with enhanced security features.

This approach not only aims to streamline operations for airline staff by providing efficient tools for managing bookings and passenger data but also enhances the booking experience for customers through a user-friendly interface. By emphasizing security measures such as encryption and secure authentication, the ARS maintains the integrity and confidentiality of sensitive data, ensuring compliance with industry standards and regulations.

Looking forward, the ARS is poised to deliver a comprehensive solution that not only meets current operational requirements but also adapts to future technological advancements and scalability needs. Through continuous improvement and adherence to best practices in software development, the ARS strives to contribute positively to the efficiency, reliability, and customer satisfaction within the airline industry.

In conclusion, by embracing these future enhancements, the ARS can evolve into a cutting-edge solution that not only meets current industry standards but also anticipates and adapts to emerging trends and customer expectations. This proactive approach ensures that the ARS remains a robust and competitive tool in the dynamic landscape of the airline industry, driving efficiency, enhancing user experience, and fostering innovation.

**Future Enhancement**

Looking ahead, the Airline Reservation System (ARS) has substantial potential for future enhancements across various fronts to meet evolving demands and technological advancements in the airline industry. First and foremost, enhancing mobile compatibility stands out as a pivotal step, recognizing the increasing preference of users to manage bookings and travel plans via smartphones and tablets. Developing a responsive mobile interface or dedicated app would extend the system's accessibility and usability, ensuring seamless interaction regardless of the device used. Advanced analytics and reporting capabilities represent another significant avenue for improvement. By integrating robust analytics tools, the ARS can provide airlines with deep insights into booking patterns, passenger behavior, and revenue forecasts. This data-driven approach enables informed decision-making in route optimization, pricing strategies, and resource allocation, ultimately enhancing operational efficiency and profitability.

Furthermore, integrating the ARS with external services such as weather updates, airport information systems, and third-party travel insurance providers would enrich the user experience. Access to real-time information and additional services enhances travel planning and customer satisfaction, aligning the system more closely with the comprehensive needs of modern travellers. Personalization remains a key focus area for future enhancements. By leveraging customer data and preferences, the ARS can offer personalized recommendations for travel packages, seating preferences, and ancillary services. This tailored approach not only enhances the user experience but also strengthens customer loyalty and satisfaction.

Moreover, expanding payment options to include digital wallets, cryptocurrencies, and other emerging methods caters to diverse customer preferences globally. This flexibility enhances convenience and broadens the system's appeal to a wider audience of travellers. Continued improvement in security measures is crucial, ensuring robust protection against cyber threats and compliance with stringent data privacy regulations. Regular security audits, vulnerability assessments, and proactive measures are essential to safeguarding user data and maintaining trust in the ARS.

**References**

* Airline, 2012 http://www.enotes.com/topic/Airline\_Reservations\_System retrieved on 23. May 2012
* C. Winston, S. Morrison (2005): "The Evolution of the Airline Industry", Brookings Institution Press, South Dakota, Cf. p. 61-62, Computer Reservation Systems.
* M. J. Smith (2002): “The Airline Encyclopedia, 1909 – 2000”. Scarecrow Press, New York.
* R. Doganis, C. Routledge (2001): “The Airline Business in the 21st Century.” McGrawHill, New York.
* R. Doganis, C. Routledge (2002): “Flying Off Course: The Economics of International Airlines,” 3rd Edition. McGraw-Hill, New York.
* R. E. G. Davies (2014): “A History of the World’s Airlines”. Oxford University Press, London.
* ReservationInterface(2012), http://www.asppms.com/autoclerk/Products/Interfaces/Reservation Interfaces/InterfacetoGDS.aspx .
* Wardell, David J, "Airline Reservation Systems", 2022. Research paper.
* Wikipedia(2012),http://en.wikipedia.org/wiki/Computer\_reservations\_system accessed on May 26, 2012 .
* Winston, Clifford, 2024 "The Evolution of the Airline Industry", Brookings Institution Press, ISBN 081575843X. Cf. p.61-62, Computer Reservation Systems.