



# **AIRLINE RESERVATION SYSTEM**

## **A PROJECT REPORT**

*Submitted by*

**HARI PRASATH C(8115U23EC032)**

*in partial fulfillment of requirements for the award of the course*

**EGB1201- JAVA PROGRAMMING**

*in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by  
AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**DECEMBER - 2024**

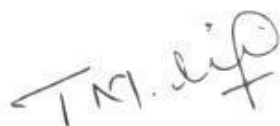
**K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “**AIRLINE RESERVATION SYSTEM**” is the bonafide work of **HARI PRASATH C (8115U23EC032)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



**SIGNATURE**

Dr. T. M. NITHYA, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

ASSOCIATE PROFESSOR

Department of CSE

K.Ramakrishnan College of Engineering  
(Autonomous)

Samayapuram-621112.



**SIGNATURE**

Mr.V.KUMARARAJA, M.E., (Ph.D.),

**SUPERVISOR**

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Engineering  
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 06/12/24



INTERNAL EXAMINER



EXTERNAL EXAMINER

## **DECLARATION**

I declare that the project report on “**AIRLINE RESERVATION SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **EGB1201 - JAVA PROGRAMMING**.

**signature**

---

**HARI PRASATH C**

Place: Samayapuram

Date: 06.12.2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Engineering (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR.V.KUMARARAJA, M.E., (Ph.D.)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To achieve a prominent position among the top technical institutions.

## **MISSION OF THE INSTITUTION**

M1: To bestow standard technical education par excellence through state of the art

infrastructure, competent faculty and high ethical standards.

M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.

M3: To provide education for developing high-quality professionals to transform the society.

## **VISION OF DEPARTMENT**

To create eminent professionals of Computer Science and Engineering by imparting quality education.

## **MISSION OF DEPARTMENT**

M1: To provide technical exposure in the field of Computer Science and Engineering through

state of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of Computer

Science and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects for

addressing the societal needs.

## **PROGRAM EDUCATIONAL OBJECTIVES**

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

## **PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.

**PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

## **PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## **ABSTRACT**

The Airline Reservation System helps airlines to manage ticket bookings. The airline staff can book tickets for the customer and the customer can view their tickets by logging into their account. This application aims to develop a basic interface for airline reservation that implements functionality like user management, login, verifying username and password, ticket booking, saving ticket information and viewing the tickets. With user verification and ability to save ticket across logins, the java application presents a simple system for airline reservation. After integrating a database for the tickets and implementing encrypted login, this application can also be developed as a web server for online bookings.



## ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
This application aims to develop a basic interface for airline reservation	PO 2	
Implements functionality like user management, login, verifying username and password, ticket booking, saving ticket information and viewing the tickets.	PO 3	PSO 1
Integrating a database for the tickets and implementing encrypted login.	PO 5	PSO 2
It can also be implemented as a web server.	PO 12	

Note: 1- Low, 2-Medium, 3- High

# TABLE OF CONTENTS

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAG</b>
<b>No.</b>		<b>No.</b>
	<b>ABSTRACT</b>	<b>viii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
<b>2</b>	<b>PROJECT METHODOLOGY</b>	<b>6</b>
	2.1 Proposed Work	6
	2.2 Block Diagram	7
<b>3</b>	<b>MODULE DESCRIPTION</b>	<b>8</b>
	3.1 Module 1 - LoginType	8
	3.2 Module 2 - LoginPage	8
	3.3 Module 3 - Booking	9
	3.4 Module 4 - Ticket	10
	3.5 Module 5 – AirlineReservation	10
<b>4</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>12</b>
	4.1 Conclusion	12
	4.2 Future Scope	12
	<b>APPENDIX A (SOURCE CODE)</b>	<b>13</b>
	<b>APPENDIX B (SCREENSHOTS)</b>	<b>18</b>
	<b>REFERENCES</b>	<b>21</b>

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Objective**

The goal is to design and develop a basic Airline Reservation System using Java, providing features such as ticket booking, user account management, and secure login functionality. The application is intended to offer a straightforward interface for airline staff to efficiently manage bookings while enabling customers to easily view their tickets. This system aims to simplify the reservation process for both administrators and users, ensuring a smooth and user-friendly experience.

### **1.2 Overview**

This project uses Java and the Swing library to create a desktop application that offers a simple interface for airline staff and customers. The application enables users to log in, book tickets, and view booked tickets. Login credentials and ticket details are managed as class members within the program. The interface is designed with multiple pages using various Swing components, providing a clean and intuitive user experience.

The application supports navigation between pages, making it easy for users to move through different functionalities. A dedicated ticket page is available, accessible from both the airline staff and customer logins. This page dynamically updates whenever a ticket is booked by the airline staff, ensuring real-time synchronization of ticket information.

By leveraging Swing, the project delivers a functional and visually organized interface while maintaining simplicity in its design. The application is structured to handle basic airline reservation tasks efficiently, with a focus on usability and seamless interaction between different roles within the system. This project demonstrates a practical implementation of Java and Swing for building an interactive and purpose-driven application.

## 1.3 Java Programming Concepts

### Basics of Object Oriented Programming:

Object-Oriented Programming (OOP) in Java is a programming paradigm that organizes software design around objects, rather than functions or logic. The core concepts of OOP in Java are:

- **Class:** A blueprint for creating objects. A class defines the properties (fields) and behaviors (methods) that its objects will have. For example, a Car class could define fields like color, model, and speed, and methods like accelerate() and brake().
- **Object:** An instance of a class. It is created based on the blueprint provided by the class. An object represents real-world entities, for example, a specific car like a "red Ferrari."
- **Encapsulation:** The technique of hiding the internal details of an object and only exposing necessary parts. This is typically achieved by using private fields and providing public getter and setter methods to access and modify the values.
- **Inheritance:** A mechanism by which one class can inherit the properties and behaviors of another. The extends keyword is used to create subclasses, enabling code reuse. For instance, a SportsCar class can inherit from the Car class.
- **Polymorphism:** The ability for a single method to perform different behaviors based on the object it is acting on. This can be achieved through method overriding and method overloading.
- **Abstraction:** Hiding complex implementation details and showing only the essential features. Abstract classes or interfaces are commonly used for abstraction.

Together, these concepts help in writing modular, reusable, and maintainable code.

### Inheritance and Polymorphism:

In this application, Java's principles of inheritance and polymorphism are utilized. Each page of the application is represented as a class that inherits from the JPanel class of the Swing library. This inheritance allows each page to have access to the features and functionalities of JPanel, such as layout management and event handling.

The main class, which manages the overall functionality of the application, implements the ActionListener interface. This interface requires the implementation of the actionPerformed() method, which takes an ActionEvent as an argument. Through

polymorphism, the actionPerformed() method can handle different button press events across the application, delegating actions based on which button was clicked.

### **Class Variables and Static Variables:**

Static variables play a significant role in storing the ticket details. These details are stored in static variables, meaning that they do not depend on an instance of the class. This makes the ticket details accessible globally, without needing to create objects of the class. The static nature of these variables ensures that the ticket data remains consistent throughout the application, regardless of the user session or page. Additionally, the textFields, buttons, and labels are instantiated as members of the JPanel class, which is essential for the user interface (UI) components.

### **Strings:**

Strings are an integral part of the application. The username and password for logging into the system are hardcoded as strings. The equals() method is used to compare the entered username and password with the hardcoded values, ensuring that the correct credentials are used for authentication. Furthermore, strings are also employed to store and display ticket details. For example, the ticket information, such as the passenger's name, flight number, and seat number, is managed as strings for easy retrieval and display on the user interface.

### **Switch-Case Statements:**

Switch-case statements are used to handle the button press events across the application. When a user interacts with any button, the event is passed to the actionPerformed() method, where a switch-case structure is used to differentiate between the various buttons. Each case corresponds to a specific action, such as navigating to a different page or booking a ticket. This approach helps in simplifying event handling, making the code more organized and efficient. It also provides an easy way to switch between pages and perform different actions based on user input.

## Swing Library:

The Swing library in Java is extensively used in this application to develop the graphical user interface (GUI). Several components from the Swing library are employed to create the window and layout of the application.

- **JFrame:** The JFrame represents the main window of the Swing application. It serves as the container for the entire user interface. The title of the window can be set using the setTitle() method, allowing the application to display a relevant title, such as "Airline Reservation System." The add() method is used to add a panel to the JFrame, which houses the actual content of each page.
- **JPanel:** Each page in the application is implemented as a subclass of JPanel. The JPanel class serves as a container for components, such as buttons, labels, and text fields. The add() method is used to add these components to the panel. By extending JPanel, each page gains access to the functionality of this container, making it easier to organize and manage the layout of UI components.
- **JButton:** The JButton class defines the buttons used in the application. Each button has a label or text displayed on it, such as "Login" or "Book Ticket." To make the buttons interactive, the addActionListener() method is used to link each button to an action listener. This method ensures that when a button is clicked, the associated action is triggered, such as navigating to the login page or processing a ticket booking.
- **JLabel:** The JLabel is used to display text in the application. This can include informative messages, instructions, or error notifications. The setText() method is used to update the text displayed by the label. For example, the "Invalid username or password" message is shown using a JLabel, and its text can be dynamically changed based on user input or system responses.
- **JTextField:** JTextField is used for inputting text, such as entering the username, password, or ticket details. The getText() method is used to retrieve the string entered by the user, which is then processed for authentication or displayed in the ticket details section.
- **BoxLayout:** The BoxLayout class is utilized to align components either horizontally or vertically. This layout manager helps in organizing the UI components efficiently, ensuring that elements such as buttons, labels, and text fields are neatly arranged

within the panel, providing a user-friendly interface.

These fundamental Java concepts, combined with the powerful Swing library, contribute to the development of a robust and interactive airline reservation system with a clean and functional user interface.

## **CHAPTER 2**

### **PROJECT METHODOLOGY**

#### **2.1 Proposed Work**

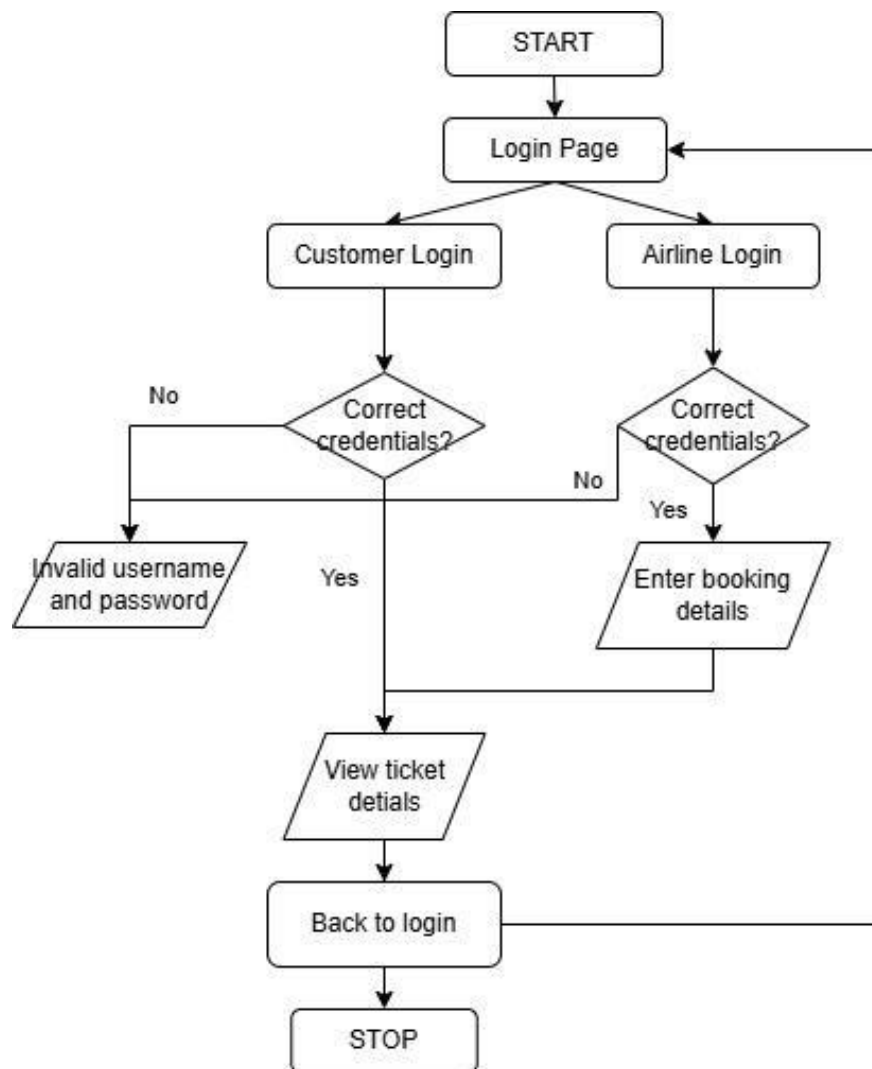
This project focuses on implementing basic login functionality for an Airline Reservation Application. The system allows both airline staff and customers to log in using their respective credentials. Once logged in, users gain access to the ticket details relevant to their role. Essential ticket information, such as passenger name, flight details, and seat number, is stored and displayed through a user-friendly interface.

Airline staff have the capability to book tickets by entering the necessary details into the system. This data is then saved and reflected on the tickets page, ensuring that all ticket-related information is up-to-date and accessible. Customers can subsequently log in to the application, navigate to the tickets page, and verify their ticket information with ease.

The project leverages a straightforward interface to facilitate seamless interaction between the airline staff and customer functionalities. By enabling efficient booking and retrieval of ticket details, the application demonstrates a functional approach to managing basic airline reservation tasks. This system highlights the integration of user-specific features, ensuring that both airline staff and customers can effectively perform their respective operations within the application.



## 2.2 Block Diagram



## **CHAPTER 3**

### **MODULE DESCRIPTION**

#### **3.1 Module 1 – LoginType**

This module implements the first page of the Airline Reservation Application, serving as the entry point for users. It inherits from the JPanel class and provides an intuitive interface for users to choose between two login options: Airline login and Customer login. The page offers clear options for both airline staff and customers to proceed based on their role. When the user selects "Airline login," the system directs them to the Airline login page, where they can enter their credentials. Alternatively, if the user clicks "Customer login," they are taken to the customer login page to input their details.

The page is designed to ensure smooth navigation between these two login options, providing a straightforward and user-friendly experience. By presenting these two choices, the module ensures that users can easily access the appropriate login page, whether they are airline staff or customers. The use of JPanel allows for a clean layout and seamless integration of the login options. This module sets the foundation for the application's functionality by enabling the user to quickly identify their role and proceed with the relevant login process.

#### **3.2 Module 2 – LoginPage**

This module implements the login page of the Airline Reservation Application, allowing users to enter their username and password to gain access. Upon submission, the system verifies the provided credentials to determine if they are correct. If the entered username and password match the stored credentials, the user is granted access to the system. However, if the credentials are incorrect, the application displays an error message stating, "Invalid username and password," notifying the user of the mismatch.

The verification process is carried out through a simple string comparison, where the input values for the username and password are compared against pre-defined valid credentials. If

the entered data does not match the stored information, the system responds by indicating an authentication failure. This straightforward approach ensures that only users with the correct credentials can access the system, maintaining security for both airline staff and customers.

By using basic string comparison for validation, this module provides a simple yet effective method for managing user authentication. It is designed to offer clear feedback to users in case of incorrect login attempts, improving the overall user experience while ensuring a secure login process. This module is a key component of the application's authentication system.

### **3.3 Module 3 – Booking**

This module is responsible for implementing the ticket booking functionality within the Airline Reservation Application. It is designed to be accessible exclusively to airline staff, who can enter the necessary details for booking tickets. The staff is required to input information such as the passenger's name, flight number, and seat number. Once the airline staff enters this data and confirms the booking, the details of the booked ticket are saved and displayed on the Ticket page.

After successfully booking the ticket, the system automatically redirects the airline staff to the Ticket page, where they can verify the details of the ticket they just booked. This ensures that the staff can review and confirm the ticket information, including the passenger's name, flight, and seat number, before finalizing the process.

The ticket booking functionality is a crucial part of the system, allowing airline staff to efficiently manage reservations. By storing the ticket details on the Ticket page, the application ensures that all information is readily available and easily accessible for verification. This module helps streamline the booking process, providing both security and efficiency while ensuring that customers' ticket details are correctly recorded and displayed for verification.

### **3.4 Module 4 – Ticket**

This module represents the Ticket page of the Airline Reservation Application, which is accessible to both the airline staff and the customer. It displays essential ticket details, including the passenger's name, flight number, and seat number. If no ticket has been booked, the page shows a message stating, "Tickets not booked," indicating that no reservation has been made yet.

The Ticket page also includes a button that allows the user to navigate back to the login page. Importantly, the Ticket page is not destroyed when the user logs out, ensuring that the data remains intact. When the user logs in again, the previously displayed ticket details are restored, providing a seamless experience for the user.

This design enhances the convenience of ticket booking and airline management by allowing users to easily verify their ticket information after logging in. For airline staff, it simplifies the process of managing bookings, while customers can quickly access their ticket details. The persistent nature of the Ticket page ensures that users don't lose their ticket information, making it easier for both customers and staff to track and manage bookings efficiently. This functionality contributes to a smoother, more user-friendly experience for all parties involved.

### **3.5 Module 5 – AirlineReservation**

This is the main class of the Airline Reservation Application, responsible for implementing the ActionListener interface to handle all button clicks. The class acts as the central controller, delegating tasks to other classes and managing the flow of the application. It initializes the window and manages user interface interactions using Swing components, ensuring smooth navigation between different parts of the application.

The main class serves as the core of the program, maintaining instances of all the pages, including the login, booking, and ticket pages. By overriding the actionPerformed() method of the ActionListener interface, it listens for and processes button click events, triggering

the appropriate actions for each event, such as transitioning between pages or performing operations like ticket booking.

As the entry point of the application, this class contains the `main()` function, which starts the execution of the program. This function is responsible for setting up the application window and launching the user interface. The main class plays a vital role in coordinating the various modules, ensuring the proper functioning of the entire system. It provides the necessary structure for interaction, effectively managing user actions and providing a seamless experience for both airline staff and customers.

## **CHAPTER 4**

### **CONCLUSION & FUTURE SCOPE**

#### **4.1 CONCLUSION**

This application enables both the airlines and the customer to book tickets seamlessly. It manages login credentials, ticket details and helps them to manage ticket booking using a desktop interface. The Airline Reservation System provides a foundational framework for managing airline ticket bookings and user accounts. By implementing features such as secure login, ticket booking, and ticket viewing, the application addresses essential requirements for an efficient reservation process.

With future enhancements like database integration and encryption for user data, the system can achieve higher reliability and security. Furthermore, transitioning to a web-based platform can expand its usability, enabling seamless online bookings and scalability to meet the demands of modern airline services. This project demonstrates a practical approach to building a robust reservation system while offering potential for further development.

#### **4.2 FUTURE SCOPE**

The future scope of the Airline Reservation System includes integrating a robust database for persistent storage, enabling real-time flight availability, and expanding it into a web-based application for online booking. Enhanced security features like advanced encryption and two-factor authentication can be implemented.

Additionally, payment gateway integration, mobile app development, and advanced features like customer feedback systems and dynamic pricing can be added. The system can also be expanded to support internationalization, allowing for multiple languages and currencies. These enhancements will improve user experience, scalability, and security, transforming the system into a comprehensive solution for modern airline reservations.

## APPENDIX A

### (SOURCE CODE)

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import javax.swing.border.EmptyBorder;

class LoginType extends JPanel {
    BoxLayout boxlayout = new BoxLayout(this, BoxLayout.Y_AXIS);
    JButton customer_login = new JButton("Customer Login");
    JButton airline_login = new JButton("Airline Login");
    LoginType() {
        this.customer_login.setAlignmentX(CENTER_ALIGNMENT);
        this.airline_login.setAlignmentX(CENTER_ALIGNMENT);
        this.setLayout(boxlayout);
        this.setBorder(new EmptyBorder(new Insets(100, 150, 100,
150)));
        this.add(customer_login);
        this.add(airline_login);
    }
}

class LoginPage extends JPanel
{ static String login_type = "";
    BoxLayout boxlayout = new BoxLayout(this, BoxLayout.Y_AXIS);
    JTextField user_name = new JTextField(16);
    JTextField password = new JTextField(16);
    JButton login_button = new JButton("Login");
    JLabel label = new JLabel("Enter "+login_type+" Login");
    LoginPage() {
        this.user_name.setAlignmentX(CENTER_ALIGNMENT);
        this.password.setAlignmentX(CENTER_ALIGNMENT);
        this.login_button.setAlignmentX(CENTER_ALIGNMENT);
        this.label.setAlignmentX(CENTER_ALIGNMENT);
        this.setLayout(boxlayout);
        this.setBorder(new EmptyBorder(new Insets(100, 150, 100,
150)));
        this.add(user_name);
        this.add(password);
        this.add(login_button);
        this.add(label);
    }
}
```

```

}

class Booking extends JPanel {
    BoxLayout boxlayout = new BoxLayout(this, BoxLayout.Y_AXIS);
    JLabel l1 = new JLabel("Name");
    JTextField name = new JTextField(16);
    JLabel l2 = new JLabel("Flight");
    JTextField flight = new JTextField(16);
    JLabel l3 = new JLabel("Seat No");
    JTextField seat_no = new JTextField(16);
    JButton booking_button = new JButton("Book Tickets");
    JButton view_tickets = new JButton("View Ticket");
    Booking() {
        this.l1.setAlignmentX(LEFT_ALIGNMENT);
        this.name.setAlignmentX(CENTER_ALIGNMENT);
        this.l2.setAlignmentX(LEFT_ALIGNMENT);
        this.flight.setAlignmentX(CENTER_ALIGNMENT);
        this.l3.setAlignmentX(LEFT_ALIGNMENT);
        this.seat_no.setAlignmentX(CENTER_ALIGNMENT);
        this.booking_button.setAlignmentX(CENTER_ALIGNMENT);
        this.view_tickets.setAlignmentX(CENTER_ALIGNMENT);
        this.setLayout(boxlayout);
        this.setBorder(new EmptyBorder(new Insets(100, 150, 100,
150)));
        this.add(l1);
        this.add(name);
        this.add(l2);
        this.add(flight);
        this.add(l3);
        this.add(seat_no);
        this.add(booking_button);
        this.add(view_tickets);
    }
}

```

```

class Ticket extends JPanel {
    BoxLayout boxlayout = new BoxLayout(this, BoxLayout.Y_AXIS);
    static boolean tickets_booked = false;
    static String name;
    static String flight;
    static String seat_no;
    JLabel n = new JLabel();
    JLabel f = new JLabel();
    JLabel s = new JLabel();
    JLabel l = new JLabel();
    JButton home = new JButton("Back to login");
}

```



```

void update_ticket()
{ this.setLayout(boxlayout)
;
this.setBorder(new EmptyBorder(new Insets(100, 150, 100,
150)));
if(tickets_booked==false)
{ l.setText("Tickets not booked");
l.setAlignmentX(CENTER_ALIGNMENT);
home.setAlignmentX(CENTER_ALIGNMENT);
this.add(l);
this.add(home);
}
else if (tickets_booked==true)
{ n.setText(name);
f.setText(flight);
s.setText(seat_no);
l.setText("Tickets Booked");
n.setAlignmentX(CENTER_ALIGNMENT);
f.setAlignmentX(CENTER_ALIGNMENT);
s.setAlignmentX(CENTER_ALIGNMENT);
l.setAlignmentX(CENTER_ALIGNMENT);
home.setAlignmentX(CENTER_ALIGNMENT);
this.add(n);
this.add(f);
this.add(s);
this.add(l);
this.add(home);
}
}
}
}

```

```

public class AirlineReservation implements ActionListener
{static JFrame frame;
static LoginType lt;
static LoginPage lp;
static Booking bk;
static Ticket tk = new Ticket();
static AirlineReservation main_class;
public static void main(String[] args) {
try {

```

```

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName())
;
}
catch(Exception e)
{ System.out.println("Invalid Platform.");
}
}

```

```

main_class = new AirlineReservation();
frame = new JFrame("AirlineReservation");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
lt = new LoginType();
lt.customer_login.addActionListener(main_class);
lt.airline_login.addActionListener(main_class);
tk.home.addActionListener(main_class);
tk.update_ticket();
frame.add(lt);
frame.pack();
frame.setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e)
{String s = e.getActionCommand();
switch (s) {
    case "Customer Login":
        lp.login_type = "customer";
        lp = new LoginPage();
        lp.login_button.addActionListener(this);
        frame.remove(lt);
        frame.add(lp);
        frame.pack();
        break;
    case "Airline Login":
        lp.login_type = "airline";
        lp = new LoginPage();
        lp.login_button.addActionListener(this);
        frame.remove(lt);
        frame.add(lp);
        frame.pack();
        break;
    case "Login":
        String un = lp.user_name.getText();
        String pw = lp.password.getText();

if(lp.login_type.equals("customer")&&un.equals("passenger")&&pw.equals("123")){
            tk.update_ticket();
            frame.remove(lp);
            frame.add(tk);
            frame.pack();
        }
if(lp.login_type.equals("airline")&&un.equals("staff")&&pw.equals("123")){
            bk = new Booking();

```

```

        bk.booking_button.addActionListener(this);
        bk.view_tickets.addActionListener(this);
        frame.remove(lp);
        frame.add(bk);
        frame.pack();
    }
    else {
        lp.label.setText("Invalid Username and
Password.");
    }    break;
case "Book Tickets":
    tk.tickets_booked = true;
    tk.name = "Name: "+bk.name.getText();
    tk.flight = "Flight: "+bk.flight.getText();
    tk.seat_no = "Seat No: "+bk.seat_no.getText();
    tk.update_ticket();
    frame.remove(bk);
    frame.add(tk);
    frame.pack();
    break;
case "Back to login":
    frame.remove(tk);
    frame.add(lt);
    frame.pack();
    break;
case "View Ticket":
    tk.update_ticket();
    frame.remove(bk);
    frame.add(tk);
    frame.pack();
    break;
default:
    break;
    }
    }
}

```

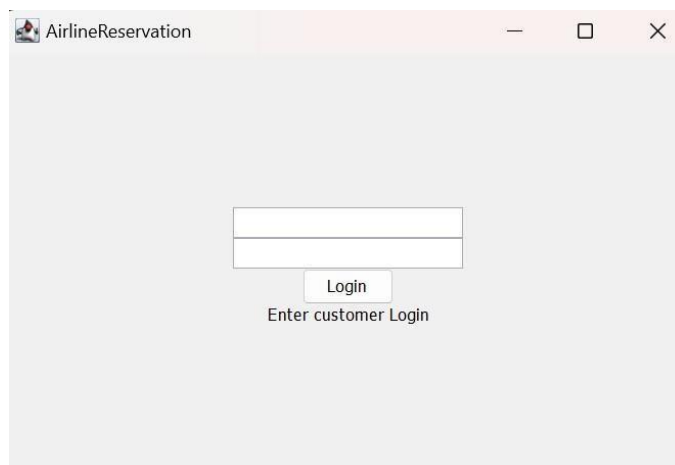
## APPENDIX B

### (SCREENSHOTS)

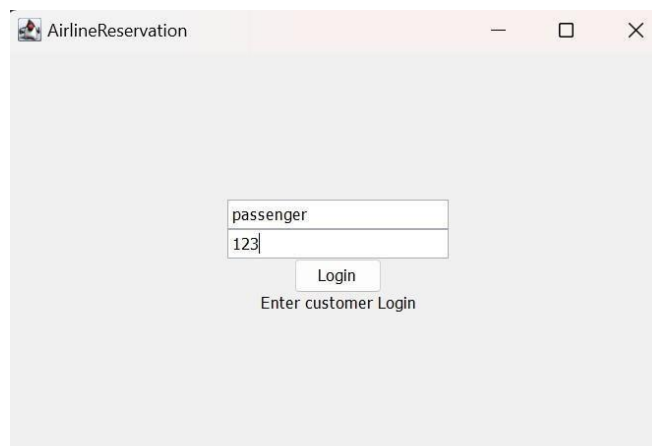
**Starting window to choose between Customer Login and Airline Login:**



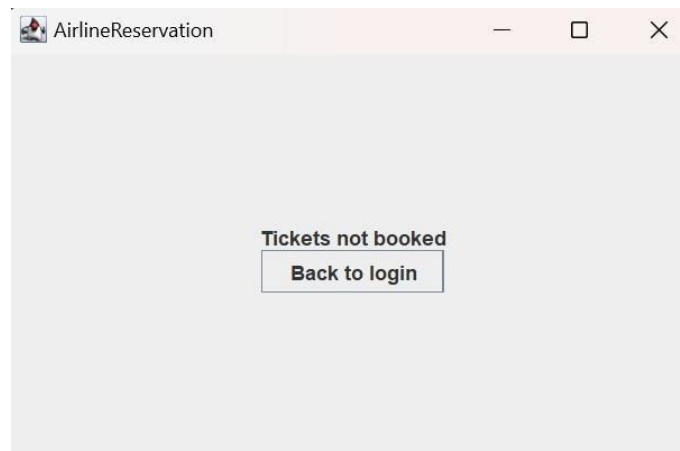
**Customer Login:**



**Correct credentials:**

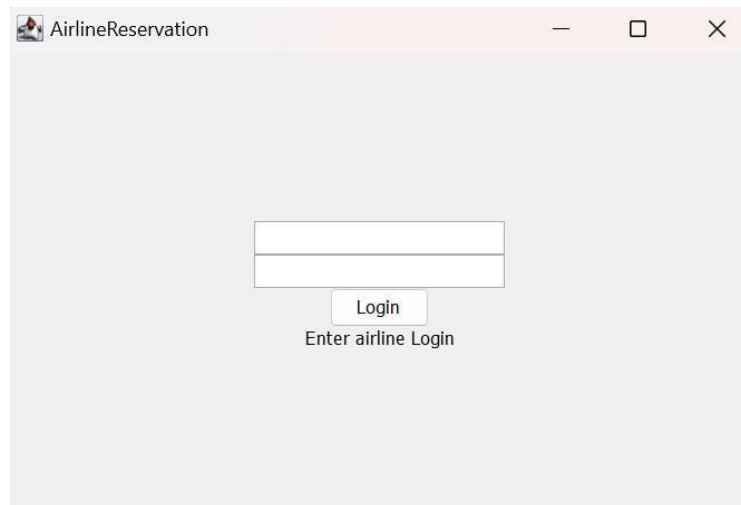


### **Ticket window without tickets:**

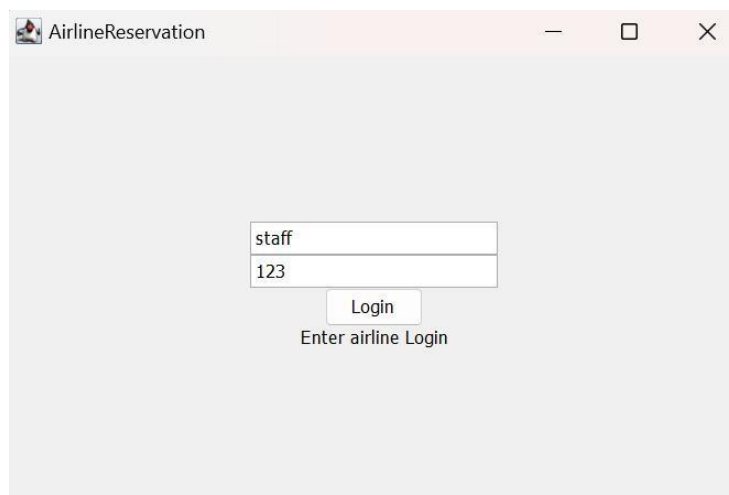


**Back to login displays the starting window**

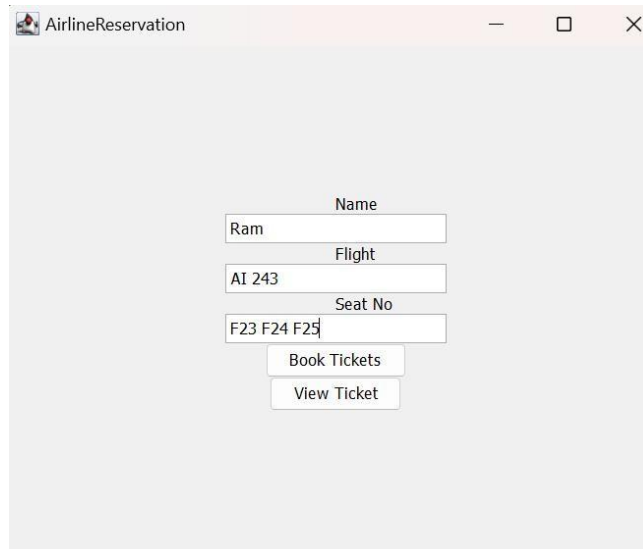
### **Airline Login:**



### **Correct credentials:**



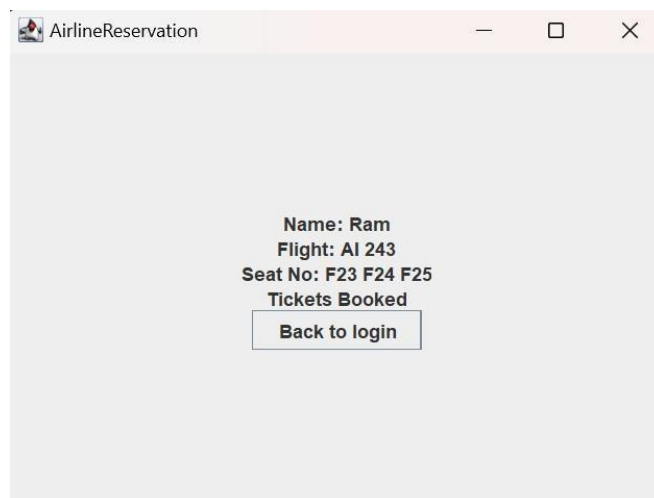
### Booking page with booking for 3 seats:



A screenshot of a web application window titled "AirlineReservation". The window contains a booking form with the following fields and buttons:

- Name:** A text input field containing "Ram".
- Flight:** A text input field containing "AI 243".
- Seat No:** A text input field containing "F23 F24 F25".
- Buttons:** Two buttons labeled "Book Tickets" and "View Ticket" are positioned below the input fields.

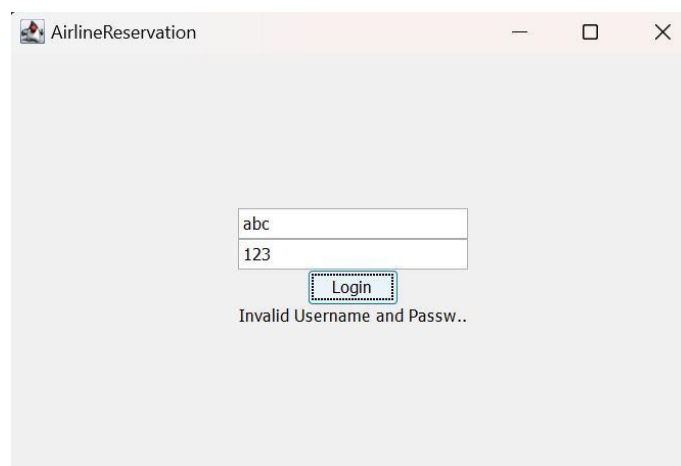
### Ticket page after booking:



A screenshot of the "AirlineReservation" window after a successful booking. The window displays the following confirmation details:

- Name:** Ram
- Flight:** AI 243
- Seat No:** F23 F24 F25
- Status:** Tickets Booked
- Button:** A button labeled "Back to login" is located below the confirmation text.

### Incorrect credentials:



A screenshot of the "AirlineReservation" window showing a failed login attempt. The window displays the following information:

- Username:** A text input field containing "abc".
- Password:** A text input field containing "123".
- Button:** A button labeled "Login" is positioned below the password field.
- Message:** A message "Invalid Username and Passw.." is displayed below the "Login" button.

## REFERENCES:

1. Schildt, H., & Coward, D. (2021). Java: The Complete Reference (13th ed.). McGraw Hill.

(This comprehensive guide covers core Java programming concepts, advanced features, and includes updates for the latest Java version).

2. GeeksforGeeks. (n.d.). Java Swing | JTextField. Retrieved from <https://www.geeksforgeeks.org/java-swing-jtextfield/>

(This article provides an overview of the JTextField component in Java Swing, including its functionality and implementation examples).

3. GeeksforGeeks. (n.d.). Java AWT | BoxLayout class. Retrieved from <https://www.geeksforgeeks.org/java-awt-boxlayout-class/>

(This tutorial explains the BoxLayout class in Java AWT, detailing how to arrange components either horizontally or vertically with examples).

4. Oracle. (n.d.). How to Use Panels. Retrieved from <https://docs.oracle.com/javase/tutorial/uiswing/components/panel.html>

(This tutorial explains the usage of JPanel in Java Swing, demonstrating how to group components together and manage layouts effectively).

5. Oracle. (n.d.). Modifying the Look and Feel. Retrieved from <https://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html>

(This guide discusses how to customize the look and feel of Java Swing applications, including changing themes and implementing the pluggable look-and-feel architecture).