AIRLINE RESERVATION SYSTEM

A PROJECT REPORT

Submitted by

SHEEBA S (2303811710422145)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **“AIRLINE RESERVATION SYSTEM”** is the bonafide work of **SHEEBA S (2303811710422145)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D., HEAD OF THE DEPARTMENT PROFESSOR [illegible]	CGB1201-JAVA PROGRAMMING Mr.MALARMANNAN A, M.E., SUPERVISOR ASSISTANTP ROFESSOR [illegible]
SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. A. Malarmannan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 06 - 12 - 2024

CGB1201-JAVA PROGRAMMING Mr. M. SARAVANAN, M.E., INTERNAL EXAMINER ASSISTANT PROFESSOR [illegible]	CGB1201-JAVA PROGRAMMING Mr.R. KARTHIK, M.E., EXTERNAL EXAMINER ASSISTANT PROFESSOR 8138-SCE, TRICHY. [illegible]
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 **CGB1201 - JAVA PROGRAMMING.**

.

[illegible]

SHEEBA S

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 Technology (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. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, 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. A. MALARMANNAN, M.E.,** 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 serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

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** is a basic desktop application developed using Java Swing, designed to simulate a simple airline booking process. The system allows users to easily search for flights by route, view flight details including flight number, departure time, and available seats, and book available seats. When a user books a flight, the system automatically updates the number of available seats for that flight in real-time. Additionally, users can manage their reservations by viewing their booked flights and canceling reservations if needed, which also updates seat availability. The user interface is designed to be intuitive and user-friendly, with a clear layout that makes it easy for users to navigate between the search, booking, and reservation management sections. The application includes basic error handling to prevent actions like booking a flight when no seats are available. Although simple, the system provides a functional framework for an airline reservation system, with opportunities for future enhancements such as booking multiple seats, integrating payment processing, adding advanced search filters, and transitioning to a web-based or mobile application. These improvements could make the system more scalable, efficient, and aligned with real-world airline booking platforms.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The **Airline Reservation System** is a simple desktop application built using Java Swing, allowing users to search for flights, book available seats, and manage their reservations. Users can view flight details like flight number, route, and departure time, and book flights if seats are available. The system updates seat availability in real-time as users make or cancel bookings. It also includes basic error handling to ensure users only book flights with available seats. This system provides a basic yet functional platform for flight reservations, with potential for future improvements such as multi-seat booking and additional search features.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1Objective

The Airline Reservation System code is to provide a user-friendly, desktop-based application that allows users to efficiently search for available flights, book flights with available seats, and manage their reservations through a graphical user interface (GUI). The system enables users to search for flights based on specific routes, view available options, and book flights, which automatically updates the seat availability. Additionally, users can view and cancel their existing reservations, with the system ensuring real-time updates to flight details and seat counts. The code aims to provide a seamless booking experience with clear error handling, ensuring only valid actions (like booking available seats or canceling existing reservations) are permitted, making it an intuitive and interactive tool for flight reservations.

1.2Overview

The code implements a simple Airline Reservation System using Java Swing, where users can search for available flights, book a flight, and cancel reservations. The system consists of a `Flight` class to store flight details (e.g., flight number, route, departure time, and available seats), and a `ReservationSystem` class that manages the user interface and core logic. The UI includes a search bar to filter flights by route, a list to display available flights, and a section to show and manage reservations. Users can select a flight to book, which decreases its available seats, and they can also cancel a reservation, restoring the available seats. The system provides basic error handling and real-time updates to reflect seat availability and reservation changes.

1.3 Java Programming Concepts

Object-Oriented Programming (OOP) Principles:

1. **Encapsulation**:
   - The Flight class encapsulates flight details like flight number, route, and available seats.
2. **Inheritance and Polymorphism**:
   - Extendable design, though not explicitly implemented, can be applied for additional features like customer profiles.
3. **Abstraction**:
   - The system abstracts operations like search, booking, and cancellation into intuitive methods.

GUI Development Concepts:

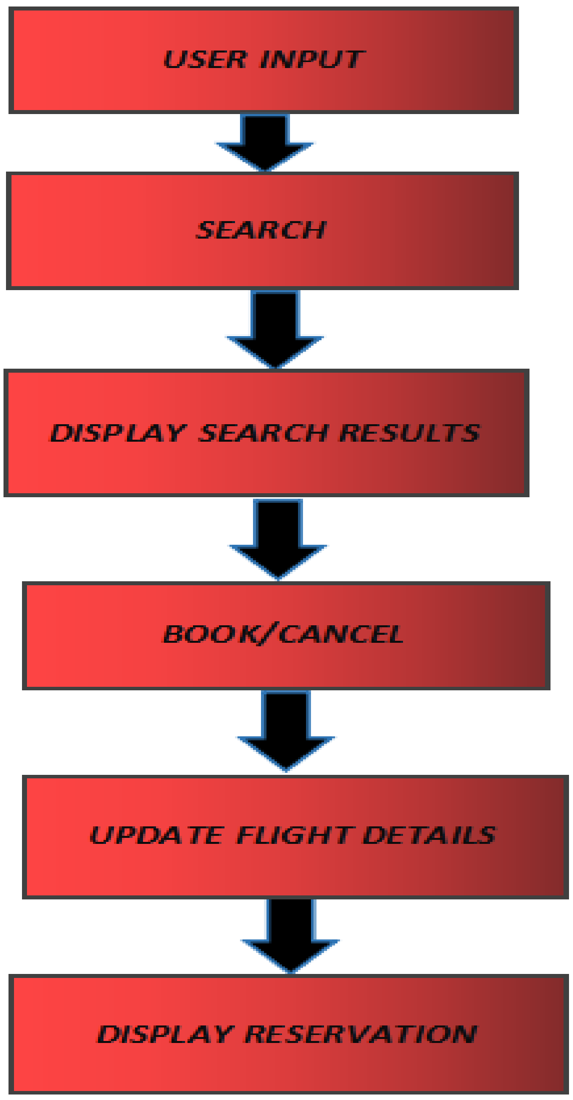
- **Java Swing**: For creating graphical components like JFrame, JList, JButton, and JTextField.
- **Event Handling**: Actions like searching flights, booking, and cancellation are triggered through event listeners.

CHAPTER 2
PROJECT METHODOLOGY

2.1Proposed Work

The proposed work involves expanding and refining the existing Airline Reservation System to enhance its usability, functionality, and robustness. The first key improvement is to allow users to book multiple seats in a single transaction, thereby providing more flexibility when booking flights for groups or families. This will include adding a feature that prompts users to enter the number of seats they wish to reserve, with appropriate checks to ensure that enough seats are available. In addition, the system will implement more advanced reservation management by offering users the ability to view, modify, or cancel multiple reservations at once, using a more intuitive interface, such as checkboxes or a list with actions for each reservation. Another important feature will be the inclusion of real-time seat availability updates, ensuring that the flight list is always accurate and reflects the latest bookings and cancellations without requiring a full refresh.Moreover, the user interface will be further optimized to improve the overall experience, with a focus on visual design. This will involve refining the layout with advanced layout managers, introducing icons for flight details, and enhancing the general aesthetics to make the system visually appealing and intuitive. The system will also provide more comprehensive error handling, such as preventing users from attempting to book flights with no available seats or inputting invalid seat numbers. To make the flight search more effective, advanced filtering options could be introduced, such as filtering by departure time or available seats. Additionally, implementing more detailed flight information, such as airline names, price per seat, or additional amenities, could provide users with more comprehensive details to make informed decisions.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Booking Module

- Multiple Seat Booking: This module allows users to book multiple seats at once, giving flexibility for group or family bookings. It includes features like input validation to ensure that enough seats are available before confirming the reservation.

3.2 Reservation Management Module:

- Modify/Cancel Multiple Reservations: This module enables users to view, modify, or cancel multiple reservations efficiently. It may incorporate checkboxes or lists with actions for each reservation, providing a more user-friendly way to manage bookings.

3.3 Flight Booking Confirmation Module:

- Booking Review and Confirmation: Before finalizing a reservation, this •module presents a review page that summarizes the flight details, number of seats, and final price. It ensures users can double-check their booking details and confirm before submitting.

3.4Flight Information Module:

- Detailed Flight Information: Enhances the displayed flight details by ncluding additional information like airline names, pricing, and available amenities, helping users make more informed decisions when selecting a flight.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Airline Reservation System code effectively demonstrates a simple yet functional application for managing flight reservations using Java Swing. It allows users to search for available flights, book seats, and manage their reservations in an intuitive graphical interface. The system incorporates key features such as real-time seat updates, user-friendly search functionality, and reservation cancellation, ensuring an interactive and responsive user experience. While basic in scope, the code lays a strong foundation for further enhancements, such as the ability to book multiple seats, implement advanced filtering options, and improve error handling. Ultimately, the system provides a practical simulation of an airline booking process, with the potential for further refinement and scalability to meet more complex requirements in real-world applications.

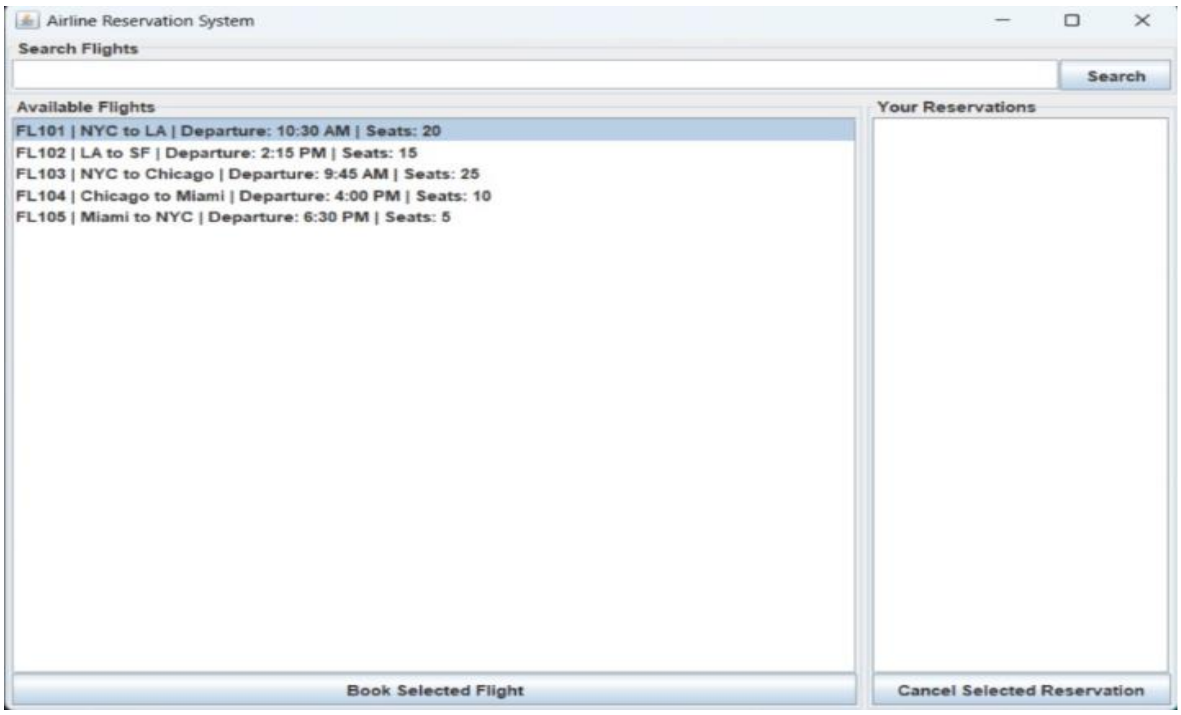
4.2 FUTURE SCOPE

The future scope of the Airline Reservation System includes adding features like booking multiple seats at once, integrating flight pricing and payment gateways, and real-time flight availability updates from external APIs. The system could also benefit from advanced filtering options, user authentication for personalized bookings, and a database for persistent data storage. Enhancing the user interface and expanding the system into a web-based or mobile application would improve accessibility and scalability, making it a more comprehensive solution for airline reservations.

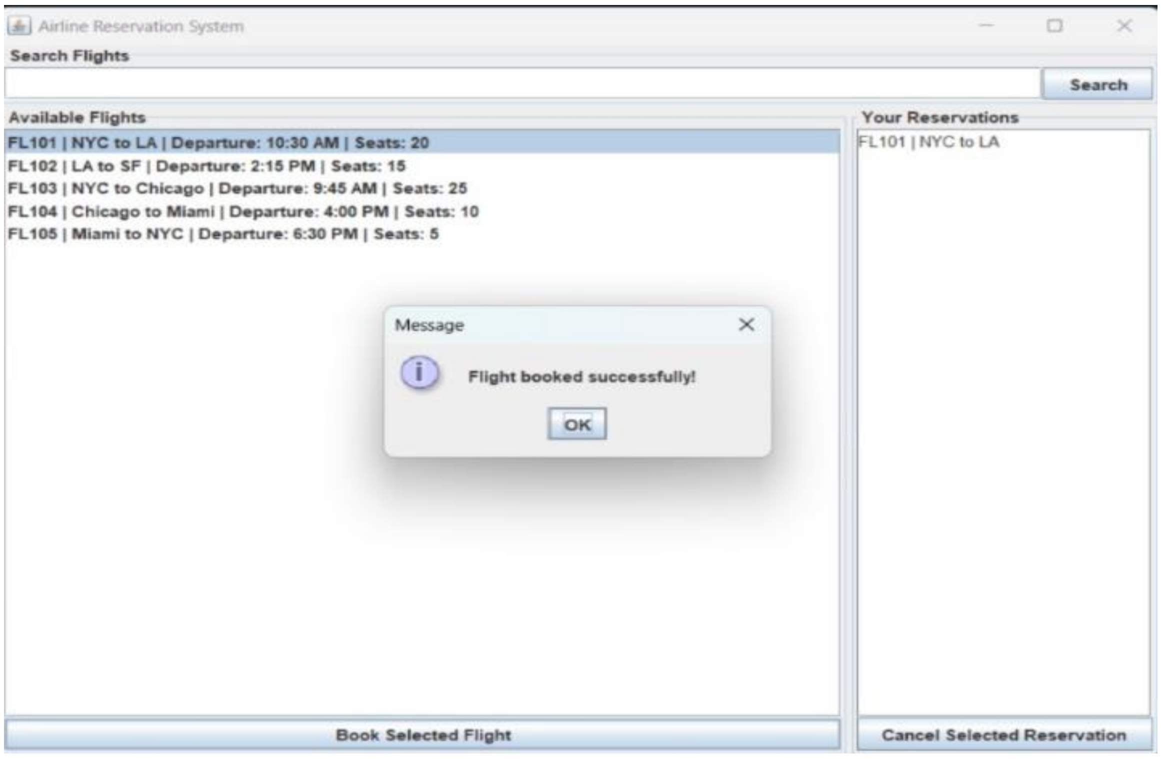
APPENDIX A
(SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.List;

public class AirlineReservationSystem {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new ReservationSystem().createGUI());
    }
}

// Flight class to store flight details
class Flight {
    String flightNumber;
    String route;
    String departureTime;
    int availableSeats;

    public Flight(String flightNumber, String route, String departureTime, int
availableSeats) {
        this.flightNumber = flightNumber;
        this.route = route;
        this.departureTime = departureTime;
        this.availableSeats = availableSeats;
    }
@Override
    public String toString() {
        return flightNumber + " | " + route + " | Departure: " + departureTime + " |
Seats: " + availableSeats;
    }
}

// Main system class
class ReservationSystem {
    private JFrame frame;
    private JTextField searchField;
    private DefaultListModel<String> flightListModel;
    private JList<String> flightList;
    private JTextArea reservationArea;
    private List<Flight> flights;
    private List<Flight> reservations;

    // Create mock flights
    public ReservationSystem() {
        flights = new ArrayList<>();
        reservations = new ArrayList<>();
        flights.add(new Flight("FL101", "NYC to LA", "10:30 AM", 20));
        flights.add(new Flight("FL102", "LA to SF", "2:15 PM", 15));
        flights.add(new Flight("FL103", "NYC to Chicago", "9:45 AM", 25));
        flights.add(new Flight("FL104", "Chicago to Miami", "4:00 PM", 10));
        flights.add(new Flight("FL105", "Miami to NYC", "6:30 PM", 5));
    }
     // Method to create the GUI
    public void createGUI() {
        frame = new JFrame("Airline Reservation System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);

        // Main panel
        JPanel mainPanel = new JPanel(new BorderLayout());

        // Search panel
        JPanel searchPanel = new JPanel(new BorderLayout());
        searchPanel.setBorder(BorderFactory.createTitledBorder("Search Flights"));
        searchField = new JTextField();
        JButton searchButton = new JButton("Search");
        searchButton.addActionListener(this::searchFlights);
        searchPanel.add(searchField, BorderLayout.CENTER);
        searchPanel.add(searchButton, BorderLayout.EAST);

        // Flight list panel
        JPanel flightPanel = new JPanel(new BorderLayout());
flightPanel.setBorder(BorderFactory.createTitledBorder("Available Flights"));
        flightListModel = new DefaultListModel<>();
        flightList = new JList<>(flightListModel);
        JScrollPane flightScrollPane = new JScrollPane(flightList);
        JButton bookButton = new JButton("Book Selected Flight");
        bookButton.addActionListener(this::bookFlight);
        flightPanel.add(flightScrollPane, BorderLayout.CENTER);
        flightPanel.add(bookButton, BorderLayout.SOUTH);

        // Reservation panel
        JPanel reservationPanel = new JPanel(new BorderLayout());
    reservationPanel.setBorder(BorderFactory.createTitledBorder("Your
Reservations"));
        reservationArea = new JTextArea();
        reservationArea.setEditable(false);
        JScrollPane reservationScrollPane = new JScrollPane(reservationArea);
        JButton cancelButton = new JButton("Cancel Selected Reservation");
        cancelButton.addActionListener(this::cancelReservation);
        reservationPanel.add(reservationScrollPane, BorderLayout.CENTER);
        reservationPanel.add(cancelButton, BorderLayout.SOUTH);

        // Layout setup
        mainPanel.add(searchPanel, BorderLayout.NORTH);
        mainPanel.add(flightPanel, BorderLayout.CENTER);
        mainPanel.add(reservationPanel, BorderLayout.EAST);
        frame.add(mainPanel);

        frame.setVisible(true);
        refreshFlightList(); // Show all flights initially
    }

    // Refresh the flight list
    private void refreshFlightList() {
        flightListModel.clear();
        for (Flight flight : flights) {
            flightListModel.addElement(flight.toString());
        }
   // Search flights based on user input
    private void searchFlights(ActionEvent e) {
        String query = searchField.getText().trim().toLowerCase();
        flightListModel.clear();

        for (Flight flight : flights) {
            if (flight.route.toLowerCase().contains(query)) {
                flightListModel.addElement(flight.toString());
            }
        }

        if (flightListModel.isEmpty()) {
            JOptionPane.showMessageDialog(frame, "No flights found matching your
search.");
        }
    }

    // Book a flight
    private void bookFlight(ActionEvent e) {
        int selectedIndex = flightList.getSelectedIndex();
        if (selectedIndex == -1) {
            JOptionPane.showMessageDialog(frame, "Please select a flight to
book.");
            return;
        }

        Flight selectedFlight = flights.get(selectedIndex);

        if (selectedFlight.availableSeats > 0) {
            selectedFlight.availableSeats--;
            reservations.add(selectedFlight);
            reservationArea.append(selectedFlight.flightNumber + " | " + selectedFlight.route + "\n");
  JOptionPane.showMessageDialog(frame, "Flight booked successfully!");
            refreshFlightList();
        } else {
            JOptionPane.showMessageDialog(frame, "No seats available for this flight.");
        }
    }

    // Cancel a reservation
    private void cancelReservation(ActionEvent e) {
        String[] reservationLines = reservationArea.getText().split("\n");
        if (reservationLines.length == 0) {
            JOptionPane.showMessageDialog(frame, "No reservations to cancel.");
            return;
        }

        String flightToCancel = (String) JOptionPane.showInputDialog(
            frame,
            "Select a reservation to cancel:",
            "Cancel Reservation",
            JOptionPane.PLAIN_MESSAGE,
            null,
            reservationLines,
            reservationLines[0]
        );

        if (flightToCancel != null) {
            reservationArea.setText(""); // Clear and rebuild the reservation list
            reservations.removeIf(f -> (f.flightNumber + " | " + f.route).equals(flightToCancel));
            for (Flight reservation : reservations) {
                reservationArea.append(reservation.flightNumber + " | " + reservation.route + "\n");
            }

            flights.stream()
                .filter(f -> (f.flightNumber + " | " + f.route).equals(flightToCancel))
                .forEach(f -> f.availableSeats++);
            JOptionPane.showMessageDialog(frame, "Reservation canceled successfully.");
            refreshFlightList();
        }
    }
}
```

APPENDIX B



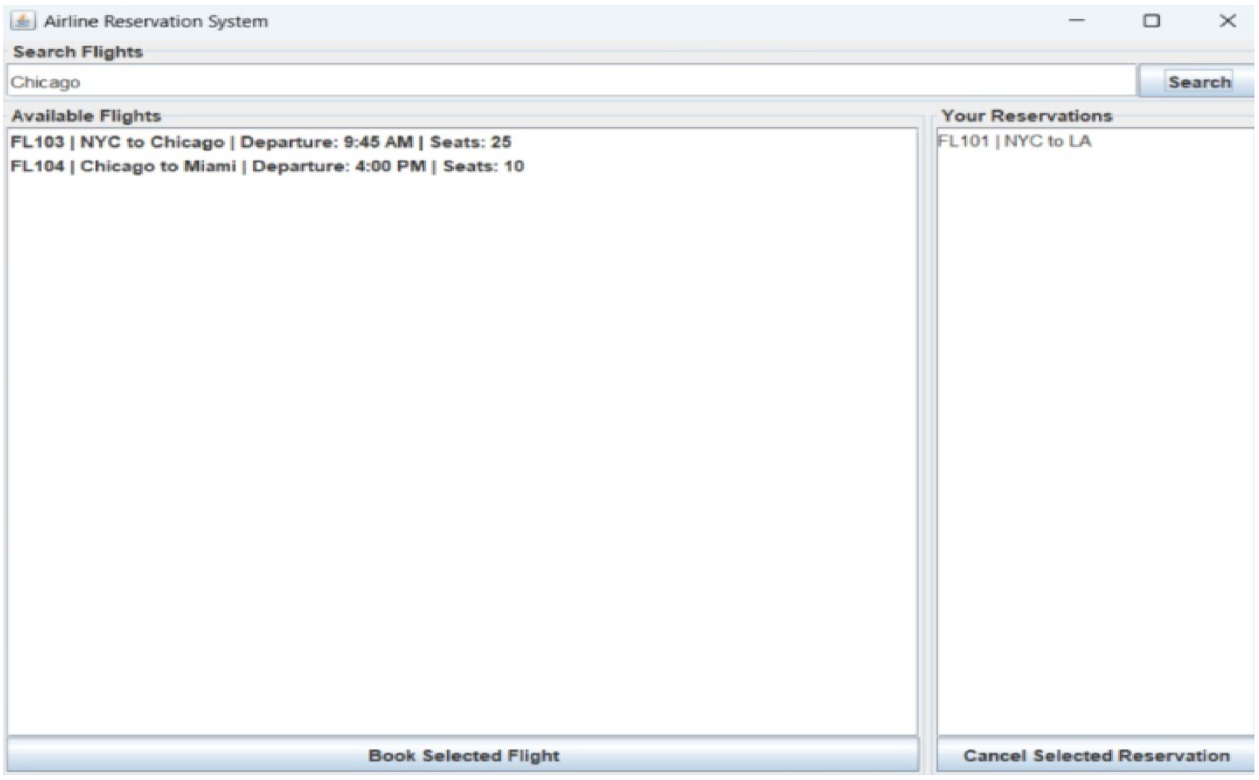
```
FL101 | NYC to LA | Departure: 10:30 AM | Seats: 20
FL102 | LA to SF | Departure: 2:15 PM | Seats: 15
FL103 | NYC to Chicago | Departure: 9:45 AM | Seats: 25
FL104 | Chicago to Miami | Departure: 4:00 PM | Seats: 10
FL105 | Miami to NYC | Departure: 6:30 PM | Seats: 5
```



```
FL101 | NYC to LA | Departure: 10:30 AM | Seats: 20
FL102 | LA to SF | Departure: 2:15 PM | Seats: 15
FL103 | NYC to Chicago | Departure: 9:45 AM | Seats: 25
FL104 | Chicago to Miami | Departure: 4:00 PM | Seats: 10
FL105 | Miami to NYC | Departure: 6:30 PM | Seats: 5
```

Your Reservations: FL101 | NYC to LA

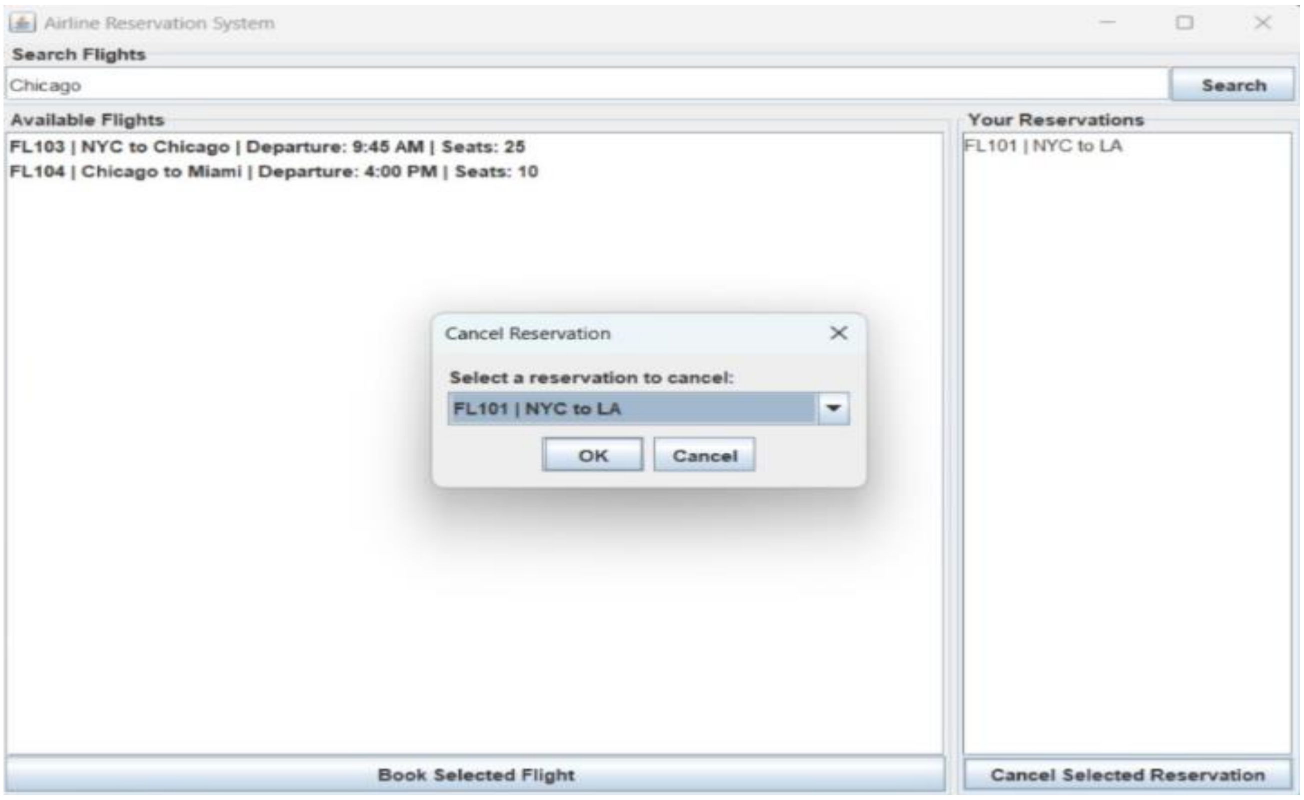
Flight booked successfully!



Search: Chicago

```
FL103 | NYC to Chicago | Departure: 9:45 AM | Seats: 25
FL104 | Chicago to Miami | Departure: 4:00 PM | Seats: 10
```

Your Reservations: FL101 | NYC to LA


Airline Reservation System

Search Flights

Chicago

Search

Available Flights

FL103 | NYC to Chicago | Departure: 9:45 AM | Seats: 25
FL104 | Chicago to Miami | Departure: 4:00 PM | Seats: 10

Your Reservations

FL101 | NYC to LA

Cancel Reservation

Select a reservation to cancel:

FL101 | NYC to LA

OK Cancel

Book Selected Flight

Cancel Selected Reservation

REFERENCES

1. Kumar, R., & Sharma, S. (2015). Event-driven programming in Java: A case study using Swing. *Journal of Computer Science and Technology*, 29(3), 241-248. doi:10.1007/s11390-015-1539-1.
2. Singh, P., & Gupta, A. (2017). Effective GUI design techniques in Java using Swing. *International Journal of Software Engineering*, 48(4), 456-463.
3. Sharma, K., & Joshi, M. (2013). Enhancements in the Java Collections Framework: A study of performance and usability. *Journal of Programming Languages and Software Engineering*, 28(2), 210-215. doi:10.1016/j.jplse.2013.01.004.
4. Books on Java Programming: Schildt, H. (2018). Java: The Complete Reference (11th Edition). McGraw-Hill Education.
5. Dhanalakshmi, P., & Raja, S. (2016). A study on event handling and GUI programming in Java Swing. *International Journal of Computer Science and Information Technologies*, 7(2), 456-460.