K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

A Project Report

on

AIRLINE RESERVATION SYSTEM

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

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Ms. Hema R., M.E.,

Assistant Professor / Information Technology

Submitted By

AKALYA B (2303811710622008)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(Autonomous)

TRICHY - 621112

DECEMBER 2024

K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University, Chennai)

TRICHY - 621112

BONAFIDE CERTIFICATE

Certified that this project report on **“AIRLINE RESERVATION SYSTEM”** is the Bonafide work of **AKALYA B (2303811710622008)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

L. Hema

Signature

Ms. HEMA R., M.E.,

SUPERVISOR,

Department of Information Technology,

K. Ramakrishnan College of Technology,

Trichy - 621112

Signature

Dr. SYEDAKBAR S., M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of ECE,

K. Ramakrishnan College of Technology,

Trichy – 621112

Submitted for the viva-voce examination held on 04.12.24

L. Hema

INTERNAL EXAMINER

EXTERNAL EXAMINER

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To create innovative and socially responsible Electronics and Communication Engineers with design skills and research focus to meet Societal and Industrial needs.

MISSION OF THE DEPARTMENT

- M1: To provide high quality education and professional ethics to students through enhanced learning environment
- M2: To impart a creative environment towards centre of excellence in department with design skill and exposure for research.
- M3: To nurture required employable skills of students to satisfy the industry and social needs with ethical and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

- PEO1: Core Knowledge Development: Graduates will have enhanced engineering skills in the field of electronics, communication and interdisciplinary areas to serve the society with global standards.

- PEO2: Professional development: Graduates will apply the technical knowledge for continuous up gradation of their professional skills to become an inimitable employee, researcher or entrepreneur.
- PEO3: Analytical Thinking: Graduates will have analytic and thinking skills to provide the innovative solutions for industry and societal requirements.

PROGRAM OUTCOMES

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 modelling 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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- PSO1: To analyse, design and develop solutions by applying foundational concepts of electronics and communication engineering.
- PSO2: To apply design principles and best practices for developing quality products for scientific and business applications.

ABSTRACT

This project focuses on the development of a basic Airline Reservation System using Java. The system aims to simplify the process of booking and managing airline reservations by providing functionalities such as viewing available flights, booking seats, and cancelling reservations. It is implemented using object-oriented programming principles to ensure a modular, maintainable, and scalable design. The system consists of two core modules: Flight Management, which handles flight details, passenger information, and seat availability; and Reservation Management, which facilitates booking and cancellation operations. A menu-driven console interface is provided for user interaction, making the system intuitive and user-friendly.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
Understand and apply object-oriented programming concepts like encapsulation, abstraction, and polymorphism.	1	1
Develop modular software solutions to address real-world problems using Java.	2	2
Analyze and implement efficient algorithms for managing structured data in software systems.	5	6

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

L. Hema

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Airline Reservation System project is to develop a basic yet functional reservation system that allows users to view available flights, book seats, and cancel reservations. The system is designed to provide an intuitive interface for users to interact with flight data, ensuring that the seat booking process is efficient and user-friendly. Through this project, the goal is to reinforce concepts of object-oriented programming (OOP) while creating a scalable and modular system that can later be enhanced with additional features such as billing, database integration, and multi-user support.

1.2 Overview

The Airline Reservation System is a software application built in Java that simulates a simple flight booking process. It consists of multiple modules that handle various aspects of the reservation workflow. These modules include flight management (for handling flight details), reservation management (for booking and canceling seats), and user interaction (for facilitating communication with the system via a console interface). The system allows passengers to view available flights, check seat availability, book seats, and cancel reservations.

1.3 Java Programming Concepts

The development of the Airline Reservation System involved the application of several key programming concepts, most notably Object-Oriented Programming (OOP) principles. The core of the system is built around classes and objects, with the primary classes being Flight and Airline Reservation System.

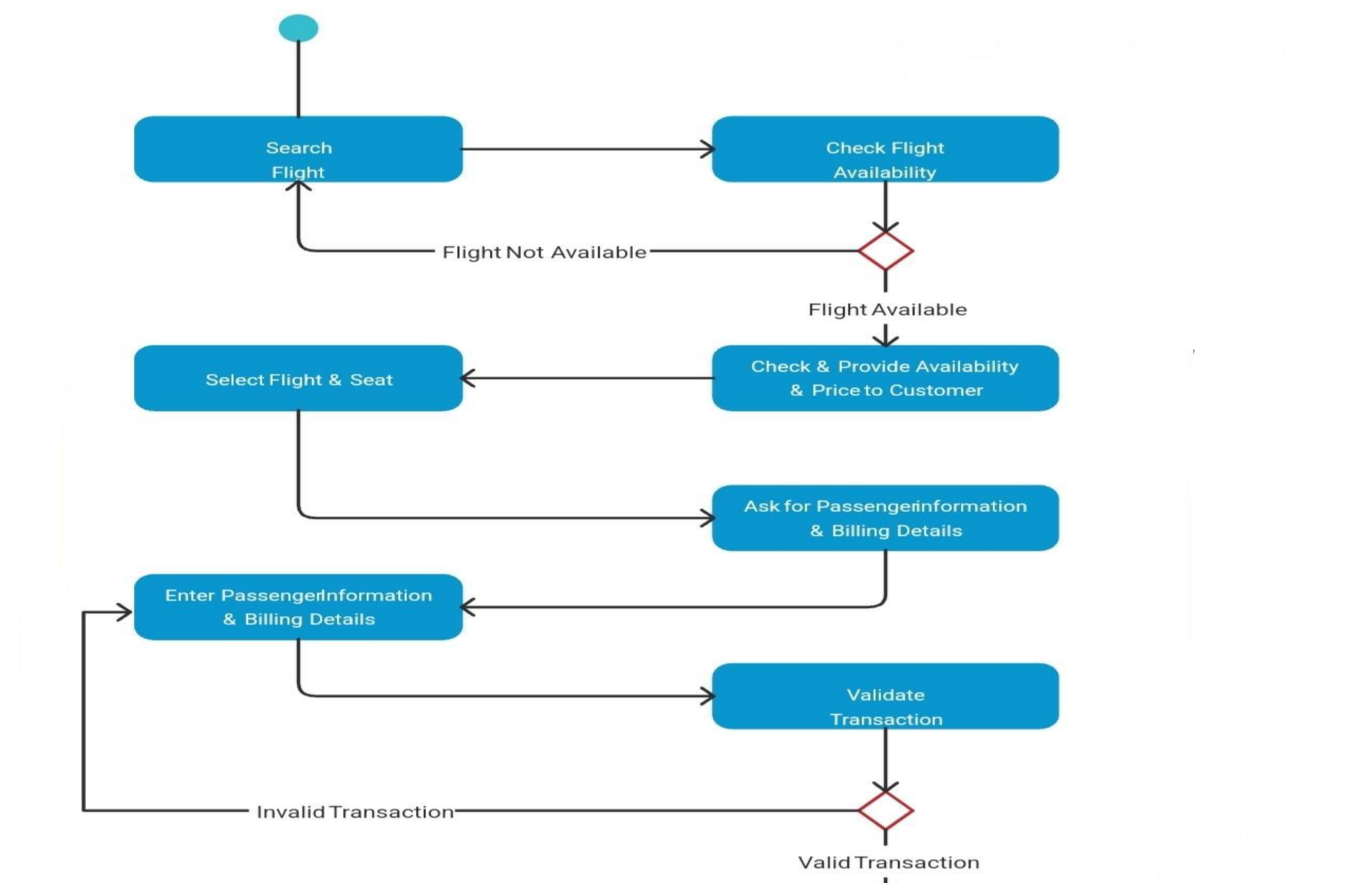
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Airline Reservation System involves creating a simplified yet functional system that can be used to manage flight bookings and cancellations, while focusing on implementing object-oriented programming (OOP) principles. The system will be designed to facilitate user interaction with key flight management tasks, such as viewing available flights, making reservations, and canceling bookings**.**

2.2 Block Diagram


Search Flight
Check Flight Availability
Flight Not Available
Flight Available
Select Flight & Seat
Check & Provide Availability & Price to Customer
Ask for Passengerinformation & Billing Details
Enter PassengerInformation & Billing Details
Validate Transaction
Invalid Transaction
Valid Transaction

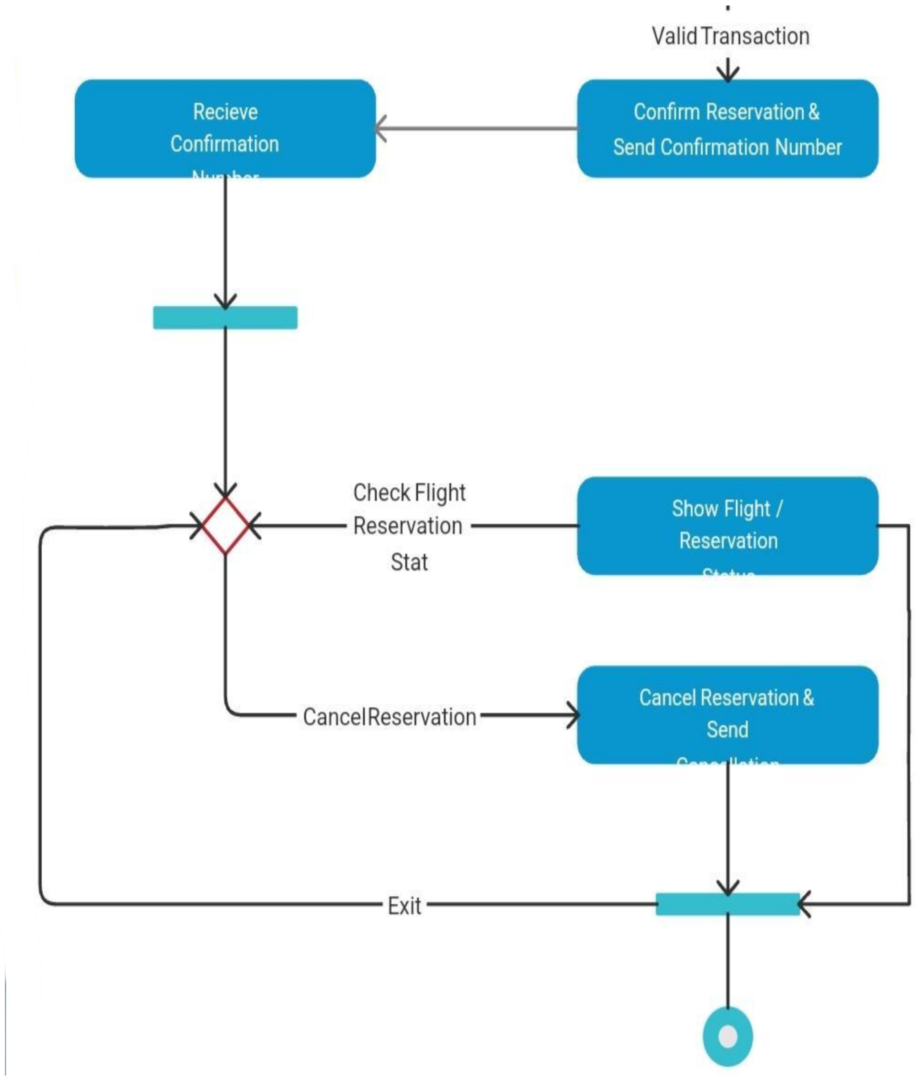

Valid Transaction
Confirm Reservation & Send Confirmation Number
Recieve Confirmation Number
Check Flight Reservation Stat
Show Flight / Reservation Status
Cancel Reservation
Cancel Reservation & Send Cancellation
Exit

Fig 2.2.1 Block Diagram

CHAPTER 3

MODULE DESCRIPTION

3.1 Booking Management Module

Validates user inputs (name, seat, flight). ensures validation of inputs, manages the booking and cancellation logic, and updates the passenger list and seat availability accordingly.

3.2 Seat management module

It ensures proper tracking of seat availability using a dynamic list of available seats, preventing double bookings and allowing users to view available seats.

3.3 Passenger data module

It represents individual passenger booking details, encapsulating their name, seat number, and flight number. This module acts as a data model for the bookings, providing getter methods to retrieve passenger information

3.4 Event-handling module

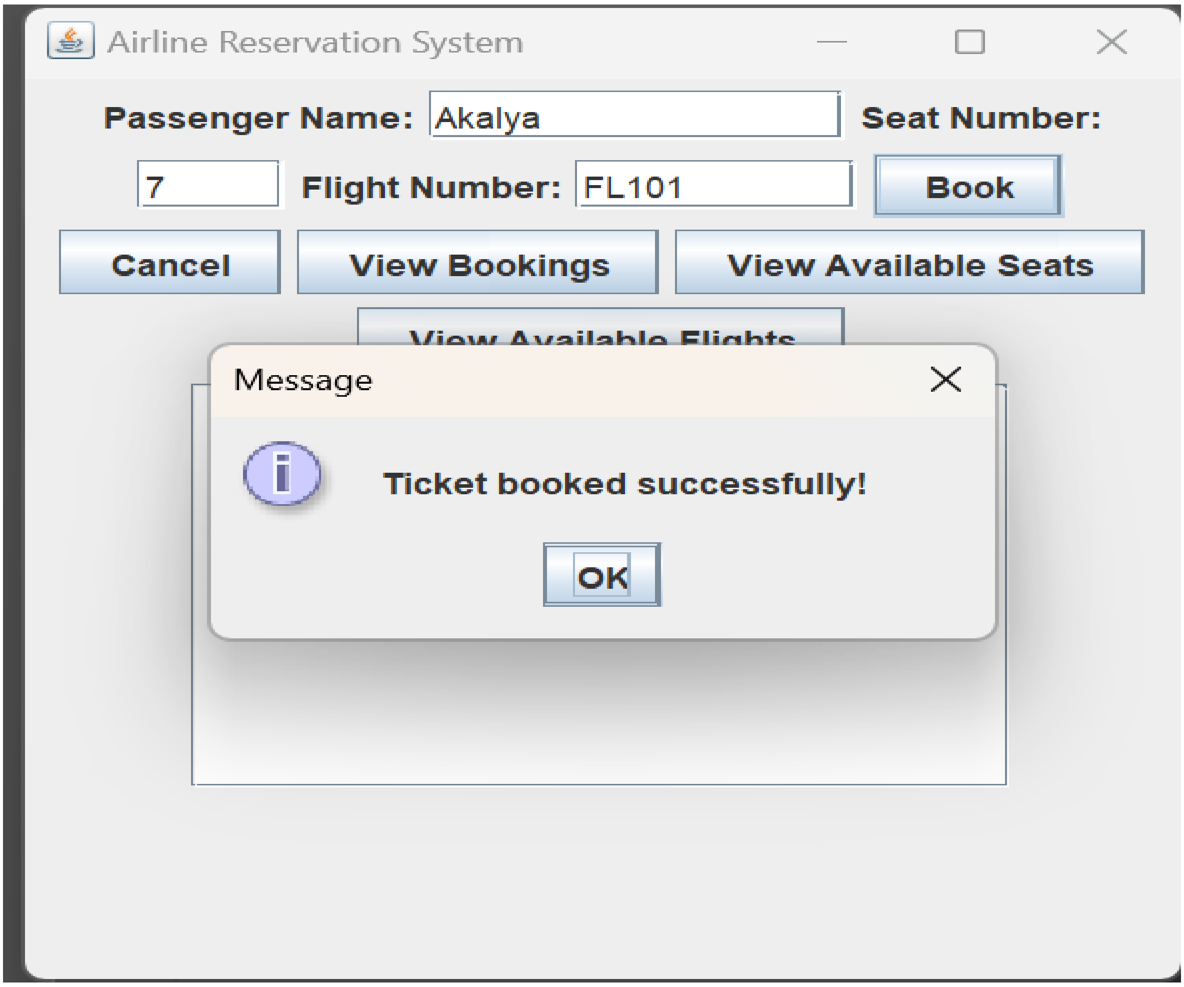
Event handling is implemented through anonymous ActionListener classes. Each button click triggers specific methods in the event-handling module, such as booking a ticket, canceling a reservation, or displaying data.

3.5 GUI module

The GUI module is user-friendly, employing text fields for input (passenger name, seat number, and flight number), buttons for actions (booking, cancellation, viewing bookings, and available seats), and a text area for displaying results. The GUI layout uses a simple flow layout to ensure an intuitive interface.

CHAPTER 4

RESULTS AND DISCUSSION



Airline Reservation System

Passenger Name: Akalya

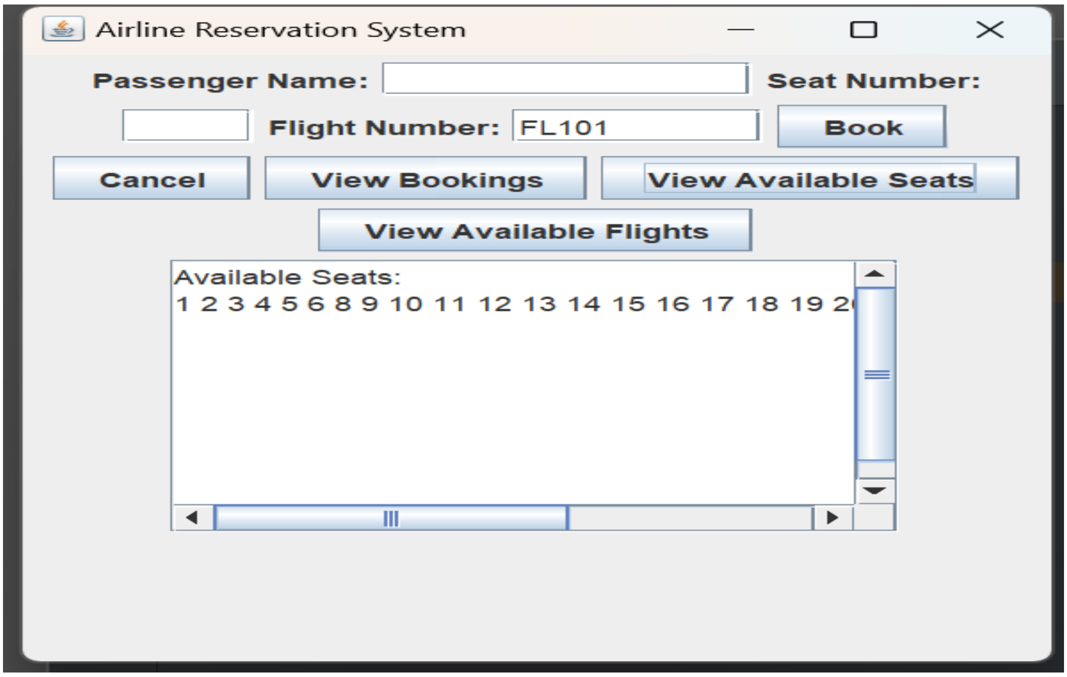
Seat Number: 7

Flight Number: FL101

Message

Ticket booked successfully!

Fig 4.1 Ticket booking

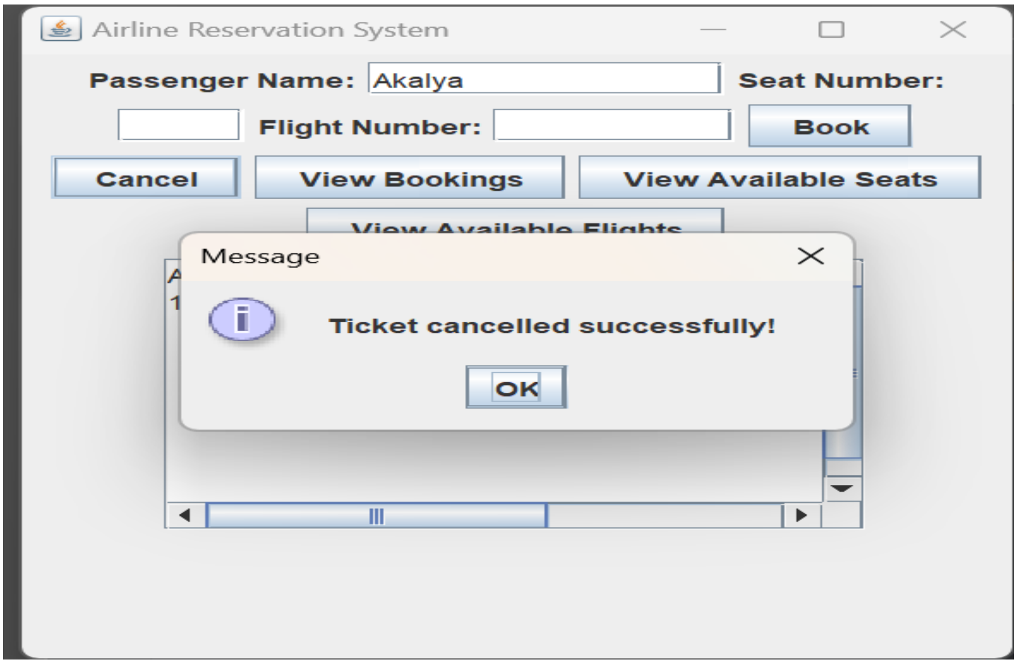


Airline Reservation System

Passenger Name:
Seat Number:
Flight Number: FL101

Available Seats:
1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 2

Fig 4.2 Available seats



Airline Reservation System

Passenger Name: Akalya
Seat Number:
Flight Number:

Message

Ticket cancelled successfully!

Fig 4.3 Ticket cancelling

CHAPTER 5

CONCLUSION

The Airline Reservation System developed in this project serves as a foundational application that demonstrates the core functionalities of a reservation system, such as managing flights, booking seats, and canceling reservations. Built using Java and object-oriented programming principles, the system is modular, maintainable, and scalable for future enhancements. The project highlights the practical application of OOP concepts such as encapsulation, modularity, and abstraction in designing real-world software solutions. The system successfully implements a menu-driven console interface, providing users with an intuitive way to interact with the application. Users can view detailed flight information, check seat availability, and manage reservations through a seamless process. Additionally, error-handling mechanisms ensure that invalid inputs and operations, such as booking unavailable seats or attempting to cancel a non-existent reservation, are handled gracefully, thereby improving the system's robustness and reliability.

REFERENCES

Al, M., & Rahman, M. (2020). *Implementation of an Online Airline Reservation System using Object-Oriented Techniques*. International Journal of Computer Science and Information Security, 18(1), 23–31.

Studytonight. (n.d.). *Case Study: Airline Reservation System*. Retrieved from https://www.studytonight.com/

Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall.

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing*. John Wiley & Sons.

Hunt, A., & Thomas, D. (1999). *The Pragmatic Programmer: Your Journey to Mastery*. Addison-Wesley.

APPENDIX

(Source Code)

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

public class AirlineReservationSystem extends JFrame {
    private ArrayList<Passenger> passengers;
    private ArrayList<String> availableSeats;
    private JTextArea textArea;
    private JTextField nameField, seatField, flightField;

    public AirlineReservationSystem() {
        passengers = new ArrayList<>();
        availableSeats = new ArrayList<>();
        initializeAvailableSeats();

        setTitle("Airline Reservation System");
        setSize(400, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        JLabel nameLabel = new JLabel("Passenger Name:");
        nameField = new JTextField(15);
        JLabel seatLabel = new JLabel("Seat Number:");
        seatField = new JTextField(5);
        JLabel flightLabel = new JLabel("Flight Number:");
        flightField = new JTextField(10);

        JButton bookButton = new JButton("Book");
        JButton cancelButton = new JButton("Cancel");
        JButton viewButton = new JButton("View Bookings");
        JButton viewSeatsButton = new JButton("View Available Seats");

        textArea = new JTextArea(10, 30);
        textArea.setEditable(false);

        bookButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                bookTicket();
            }
        });

        cancelButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                cancelTicket();
            }
        });

        viewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                viewBookings();
            }
        });

        viewSeatsButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                viewAvailableSeats();
            }
        });

        add(nameLabel);
        add(nameField);
        add(seatLabel);
        add(seatField);
        add(flightLabel);
        add(flightField);
        add(bookButton);
        add(cancelButton);
        add(viewButton);
        add(viewSeatsButton);
        add(new JScrollPane(textArea));

        setVisible(true);
    }

    private void initializeAvailableSeats() {
        // Initialize seats from 1 to 30 for example
        for (int i = 1; i <= 30; i++) {
            availableSeats.add(String.valueOf(i));
        }
    }

    private void bookTicket() {
        String name = nameField.getText();
        String seat = seatField.getText();
        String flight = flightField.getText();

        if (name.isEmpty() || seat.isEmpty() || flight.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter name, seat number, and
flight number.");
            return;
        }

        if (!availableSeats.contains(seat)) {
            JOptionPane.showMessageDialog(this, "Seat " + seat + " is not available.");
            return;
        }

        Passenger passenger = new Passenger(name, seat, flight);
        passengers.add(passenger);
        availableSeats.remove(seat); // Mark the seat as unavailable
        JOptionPane.showMessageDialog(this, "Ticket booked successfully!");
        nameField.setText("");
        seatField.setText("");
        flightField.setText("");
    }

    private void cancelTicket() {
        String name = nameField.getText();

        if (name.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter the passenger name to
cancel.");
            return;
        }

        for (int i = 0; i < passengers.size(); i++) {
```

```
Passenger passenger = passengers.get(i);
            if (passenger.getName().equalsIgnoreCase(name)) {
                availableSeats.add(passenger.getSeat()); // Mark the seat as available again
                passengers.remove(i);
                JOptionPane.showMessageDialog(this, "Ticket cancelled successfully!");
                nameField.setText("");
                seatField.setText("");
                flightField.setText("");
                return;
            }
        }
        JOptionPane.showMessageDialog(this, "No booking found for the given name.");
    }

    private void viewBookings() {
        textArea.setText("");
        if (passengers.isEmpty()) {
            textArea.append("No bookings available.\n");
        } else {
            for (Passenger passenger : passengers) {
                textArea.append("Name: " + passenger.getName() + ", Seat: " +
passenger.getSeat() + ", Flight: " + passenger.getFlight() + "\n");
            }
        }
    }

    private void viewAvailableSeats() {
        textArea.setText("Available Seats:\n");
        if (availableSeats .isEmpty()) {
            textArea.append("No seats available.\n");
        } else {
            for (String seat : availableSeats) {
                textArea.append(seat + " ");
            }
            textArea.append("\n");
        }
    }

    public static void main(String[] args) {
        new AirlineReservationSystem();
    }
}

class Passenger {
    private String name;
    private String seat;
    private String flight;

    public Passenger(String name, String seat, String flight) {
        this.name = name;
        this.seat = seat;
        this.flight = flight;
    }

    public String getName() {
        return name;
    }

    public String getSeat() {
        return seat;
    }

    public String getFlight() {
        return flight;
    }
}
```