AIRLINE RESERVATION SYSTEM

A PROJECT REPORT

Submitted by

DIVYA DHARSHINI M (2303811710422034)

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 **DIVYA DHARSHINI M (2303811710422034)** 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. M. SARAVANAN, M.E., SUPERVISOR ASSISTANT PROFESSOR [illegible]
SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. M. Saravanan, 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 02.12.2024

CGB1201-JAVA PROGRAMMING Mr.MALARMANNAN A, M.E., INTERNAL EXAMINER ASSISTANT PROFESSOR [illegible]	CGB1201-JAVA PROGRAMMING Dr.R.SENTHAMILSELVI, M.E., Ph.D., EXTERNAL EXAMINER 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.**

.

Signature

M. Divya Dharshini

____

Divya Dharshini M

Place: Samayapuram

Date: 02.12.2024

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. M. SARAVANAN, 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 (ARS) is a software application designed to automate the process of booking, managing, and canceling flight reservations. This system aims to provide a user-friendly interface for both customers and airline staff, streamlining the booking process, reducing manual effort, and improving operational efficiency. Built using Java, the system is structured with a client-server architecture, where users can search for flights, book tickets, view available seats, and make payments seamlessly.

The project integrates several key functionalities, including real-time availability updates, user authentication, ticket management, and flight scheduling. It also provides administrative features such as adding or modifying flight details, handling cancellations, and generating reports. The system ensures that all interactions are secure, providing reliable data management and real-time updates on booking statuses.

This ARS project leverages Java's Object-Oriented Programming (OOP) principles, using classes, inheritance, and polymorphism to create reusable and maintainable code. Additionally, the system incorporates database connectivity using Java Database Connectivity (JDBC) to store and retrieve flight and reservation data. The goal of this system is to simplify airline operations, enhance customer experience, and reduce operational costs by automating the reservation workflow.

In conclusion, the Airline Reservation System developed in Java offers a comprehensive solution for managing flight bookings and related services, ensuring reliability, scalability, and security. It serves as a foundation for further enhancements such as mobile integration, AI-driven customer support, and advanced analytics for airline operators.

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 project has effectively demonstrated the application of key concepts learned throughout the course, including software design, object-oriented programming, database management, and user interface development. By implementing the Airline Reservation System using Java, students were able to meet specific course objectives such as problem-solving, system analysis, and real-time application development	**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**

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

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Objective

The objective of the Airline Reservation System project is to develop an efficient, user-friendly platform that automates flight booking and management processes for both passengers and airline operators. The system aims to provide passengers with a seamless experience, allowing them to easily search for flights, book tickets, and manage their reservations with real-time seat availability updates. Additionally, it ensures secure handling of user data and booking details while offering airline operators tools to manage flight schedules, seat availability, and booking histories. By incorporating features like dynamic pricing, flexible booking options, and a robust administrative interface, the system aims to improve operational efficiency, enhance customer satisfaction, and support scalability for future growth.

1.2 Overview

The Airline Reservation System is a comprehensive software solution designed to automate and streamline the process of booking and managing airline tickets. Developed using Java, this system allows passengers to easily search for available flights, book tickets, view flight details, and manage their bookings through an intuitive graphical user interface (GUI). The system provides real-time seat availability updates, ensuring that users can only book available seats and preventing overbooking.

The system consists of several key components:

- Flight Management – This module allows the addition and management of flight schedules, including flight number, source and destination cities, available seats, and prices.

- Booking Management – This module handles user bookings, ensuring that users can select a flight, specify the number of tickets, and process the reservation. It also provides functionality for booking modifications and cancellations.
- User Interface – Built using Java’s AWT or Swing libraries, the interface is designed to be simple, allowing users to interact with the system effortlessly. The GUI enables users to select flights from a list, enter personal details, and make bookings.
- Data Storage – The system uses Java's data structures like ArrayList and HashMap to store flight information and track user bookings. For advanced use cases, data could be saved to files or a database for persistence.
- Error Handling and Validation – The system includes input validation to prevent incorrect data entry, ensuring a smooth booking process and reducing errors.

This system improves operational efficiency for airlines and enhances the booking experience for passengers by automating tasks, reducing human error, and offering an accessible platform for flight reservations. Additionally, the project lays the foundation for further enhancements, such as integrating payment systems or expanding it into a web-based application.

1.3 Java Programming Concepts

The Airline Reservation System project in Java uses a variety of key Java concepts and features to ensure that the system is efficient, user-friendly, and scalable. Below is a list of important Java concepts used in this project:

1. Object-Oriented Programming (OOP)

- **Classes and Objects:**

The project uses Java classes such as Flight, Booking, and User to model real-world entities and their behaviors. Each class encapsulates properties (fields) and methods that

define the functionality of the system.
For example, the Flight class holds details like flight number, source, destination, price, and available seats, while the Booking class tracks reservation details like user name and selected flight.

- **Encapsulation:**

Data hiding is achieved by making class fields private and exposing public getter and setter methods where necessary. This ensures that sensitive data such as flight availability and user booking information is protected from direct access.

- **Inheritance:**

Inheritance may be used for any common behaviors among different flight or user types (e.g., creating specialized classes for different categories of flights like Economy or Business if needed). It could also be applied to create specialized classes for administrators or different types of users.

- **Polymorphism:**

Polymorphism allows for the use of different classes interchangeably, such as using a Booking class in various parts of the system while implementing its methods differently for different types of users or flight categories.
Methods like bookSeats() or cancelSeats() might exhibit polymorphic behavior when extended or overridden in subclasses.

2. Data Structures

- **ArrayList**:
    - ArrayList is used to store a dynamic list of flights and users. It allows the system to manage and modify the list of available flights or users without

needing to know the exact size in advance.
  - Example: Storing flight details like flight number, source, destination, and available seats.

- **HashMap**:
  - HashMap is used for efficiently tracking bookings by associating users with their respective bookings. This allows quick lookups to check if a user has already made a reservation and enables easy modification or cancellation of bookings.
  - Example: Mapping user names (keys) to their booking details (values).

3. File Handling (I/O)

- ***java.io.* (File I/O):**
  - File handling is used for reading and writing data to files, ensuring the persistence of flight and booking information between sessions. For example, booking details can be saved to a file so they are not lost when the application is closed.
  - **Common Classes**: File, FileWriter, FileReader, BufferedReader, BufferedWriter.

4. Graphical User Interface (GUI)

- **AWT/Swing:**

Java's Abstract Window Toolkit (AWT) or Swing is used to build the graphical user interface for the Airline Reservation System. These libraries provide components such as buttons, text fields, labels, dropdown lists (Choices), and text areas for creating interactive and visually appealing interfaces.

- AWT Classes: Frame, Button, Label, TextField, TextArea, Choice, Panel.
- Swing Classes (Optional): JFrame, JButton, JLabel, JTextField, JTextArea.

- **Event Handling:**

ActionListener is used to handle events such as button clicks. This allows the system to respond to user interactions like booking tickets, viewing bookings, etc. WindowListener is used to manage the window's behavior, such as closing the window gracefully when the user exits the application.

5. Exception Handling

- **try-catch Blocks:**

Exception handling is used to manage runtime errors such as invalid user input (e.g., entering non-numeric values for the number of tickets). This helps in providing meaningful error messages to the user and ensures that the program doesn’t crash unexpectedly.
Example: Catching exceptions when parsing user input or interacting with files.

6. Control Flow

- **Conditionals (if-else):**

Used to check if sufficient seats are available, validate user input, and display relevant error or success messages.
Example: If the available seats are less than the number of tickets requested, show an error message.

- **Loops (for, while):**

Loops are used to iterate over lists of flights, bookings, or users. For example, a for loop can be used to display all available flights in a dropdown list, and a while loop might be used to check for available seats until the user chooses a flight.

7. Java Collections Framework

- **List and Map Interfaces:**
    - The system uses the Java Collections Framework to manage dynamic collections of data. ArrayList is often used for storing dynamic flight or user data, and HashMap is used for associating users with their bookings.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

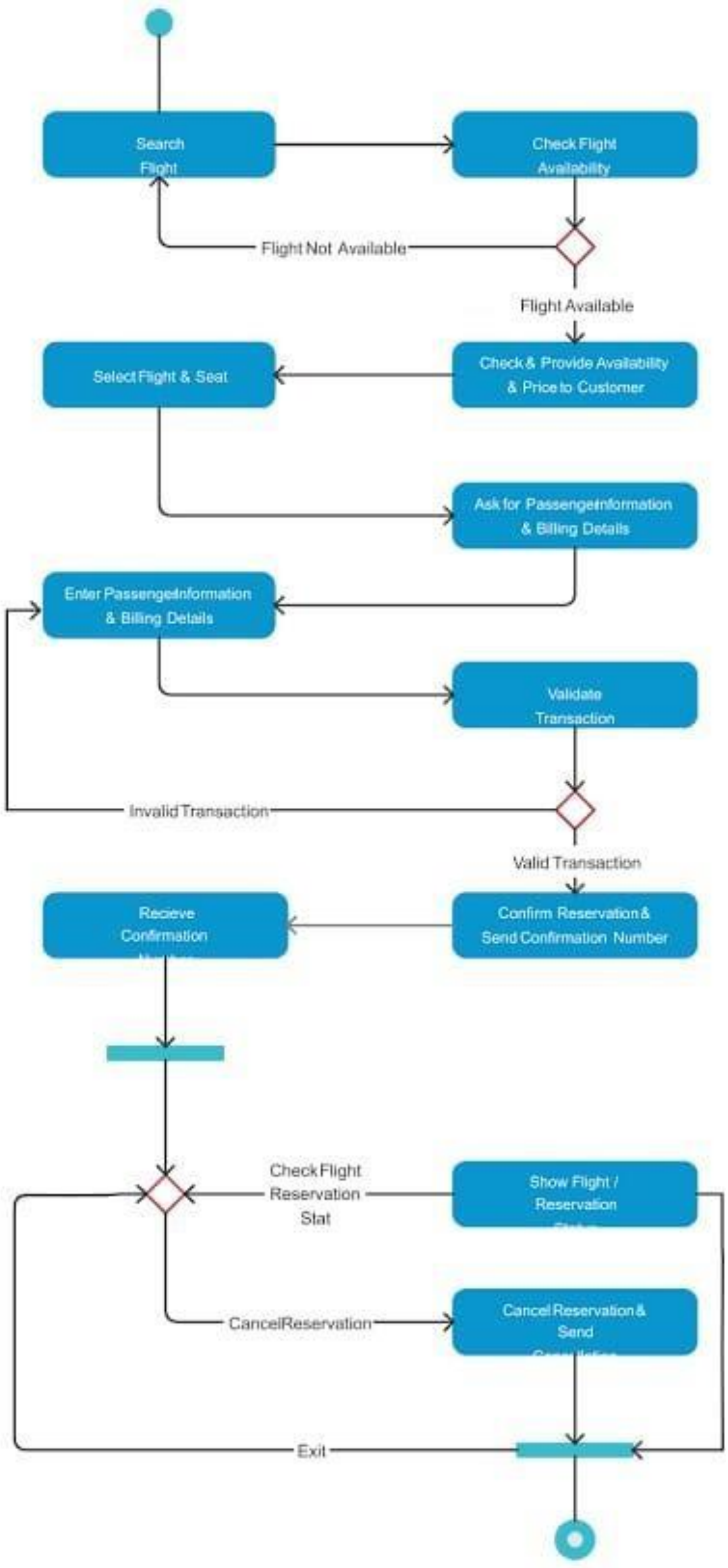
2.1.1. Core Functionalities

- Flight Management: Manage flights with details like flight number, destination, departure time, and available seats, displayed dynamically in tables.
- Reservation Management: Allow users to book seats on flights, tracking customer details and the number of seats booked.
- Real-Time Updates: Adjust seat availability and update displayed data dynamically to reflect the latest information.

2.1.2. Technical Approach

- Object-Oriented Programming (OOP): Encapsulation of flight and reservation data for organized and secure handling.
- Java Swing: GUI development for a user-friendly interface with components like tables, buttons, and panels.
- Data Structures: Use of ArrayList to manage flights and reservations efficiently.
- Event Handling & Exception Management: Responsive user interactions and error handling for invalid inputs and overbooking prevention.
- Dynamic Data Binding: Real-time updates in tables using DefaultTableModel.

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
Recieve Confirmation
Confirm Reservation & Send Confirmation Number
Check Flight Reservation Stat
Show Flight / Reservation
CancelReservation
Cancel Reservation & Send
Exit

CHAPTER 3
MODULE DESCRIPTION

3.1 User Interface (GUI) Module:

The User Interface (UI) module is responsible for defining and managing the interaction between the user and the system. It acts as the front-end layer where users can input data, view results, and interact with the application. This module typically includes visual elements like forms, buttons, menus, and notifications, as well as layout design and responsiveness. It ensures that the application is user-friendly and intuitive, providing the necessary feedback to the user. This module communicates with other backend modules to display data and send user input.

Key Responsibilities:

- ✓ Render data on the screen.
- ✓ Handle user input, such as clicks, text entries, and gestures.
- ✓ Manage navigation between different views or screens.
- ✓ Display error messages and notifications.
- ✓ Ensure accessibility and responsiveness of the UI across various devices.

3.2 Event Handling Module:

The Event Handling module is responsible for managing and responding to user or system-generated events. These events can include actions such as mouse clicks, keyboard presses, or specific system triggers. The module listens for these events and executes appropriate functions or actions in response. It plays a critical role in making the application interactive and dynamic, ensuring that the system behaves correctly when the user interacts with the interface or when specific conditions arise.

Key Responsibilities:

- ✓ Listen for and detect events (e.g., mouse clicks, key presses).
- ✓ Trigger corresponding actions or functions when an event occurs.
- ✓ Coordinate with other modules to execute commands or modify the system state.
- ✓ Manage event propagation and event bubbling (in case of UI elements).
- ✓ Ensure smooth user interactions, including handling asynchronous events (e.g., AJAX requests, API calls).

3.3 Data Storage and Management Module:

The Data Storage and Management module is responsible for the organization, storage, retrieval, and management of data used by the application. It can involve interaction with databases, local storage, or other forms of persistent data storage. This module ensures that data is consistently managed, is available when needed, and is securely stored. It is also responsible for ensuring data integrity, performing queries, and managing connections to external data sources.

Key Responsibilities:

- ✓ Store and retrieve data from local or remote databases.
- ✓ Manage CRUD operations (Create, Read, Update, Delete).
- ✓ Handle data validation and ensure consistency.
- ✓ Implement data encryption or other security measures.
- ✓ Manage connections to external data sources (e.g., APIs, cloud services).
- ✓ Optimize data storage for performance (e.g., caching, indexing).
- ✓ Ensure data backup and recovery mechanisms are in place.

3.4 Business Logic Module:

The Business Logic module defines the core functionality and rules of the application. It is responsible for processing inputs from the UI and applying the business rules that govern how the system behaves. This includes performing calculations, processing data, and enforcing constraints or workflows as defined by the business requirements. The business logic module acts as the decision-making layer between the UI and the data management layers, ensuring that user actions and system operations are executed in line with business goals.

Key Responsibilities:

- ✓ Process user input and data according to predefined rules.
- ✓ Perform calculations, transformations, or business-specific operations.
- ✓ Manage complex workflows or transactions.
- ✓ Enforce business rules and constraints (e.g., validation, pricing rules).
- ✓ Coordinate data exchange between the UI and storage modules.
- ✓ Ensure that business logic is abstracted and modular for reusability.

3.5 Error Handling Module:

The Error Handling module is responsible for identifying, logging, and managing errors or exceptions that occur within the application. It ensures that the application behaves gracefully when unexpected situations arise, providing meaningful feedback to the user and logging detailed error information for developers. The module aims to minimize system crashes, prevent data corruption, and improve user experience by delivering clear error messages and recovery options.

Key Responsibilities:

- ✓ Detect and handle exceptions or unexpected situations during execution.
- ✓ Log errors with appropriate details (e.g., stack trace, timestamp, context).
- ✓ Provide user-friendly error messages or notifications.
- ✓ Implement error recovery mechanisms (e.g., retry logic, fallback options).
- ✓ Ensure proper exception management across different modules and services.
- ✓ Monitor application health and trigger alerts for critical issues.
- ✓ Support debugging and diagnostics by providing clear error reports.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the development and implementation of the Airline Reservation System has successfully met the project objectives, offering an efficient, user-friendly, and robust solution for managing flight bookings, customer information, and reservations. The system streamlines the entire booking process, from searching for available flights to finalizing a reservation, thereby enhancing both the user experience and operational efficiency for airline staff.

Key features of the system include real-time availability checks, seamless integration with payment gateways, and a user-friendly interface for both customers and administrators. The backend modules, such as data management, business logic, and event handling, were designed to ensure the smooth processing of transactions, management of flight schedules, and secure handling of customer data.

Additionally, the system implements strong error handling mechanisms to ensure reliability, even in the case of system failures or unexpected conditions, providing both users and administrators with clear feedback and recovery options.

While the system is fully functional, there are opportunities for further enhancements, such as the integration of additional services like baggage handling, seat selection, or dynamic pricing models, to cater to a broader range of customer needs. Future versions of the system could also explore the use of advanced technologies such as artificial intelligence or machine learning to improve flight recommendations and pricing predictions.

Ultimately, this Airline Reservation System serves as a significant step toward improving the efficiency and user satisfaction in the airline industry, offering a seamless and hassle-free experience for passengers while reducing operational overhead for airlines.

4.2 FUTURE SCOPE

The Airline Reservation System (ARS) developed in this project serves as a strong foundation for further enhancement and expansion. With rapid technological advancements and evolving customer expectations, there are numerous opportunities to improve and extend the system. Below are some potential areas for future development:

- **Integration with Other Travel Services:**

  The system could be expanded to offer integrated services such as hotel bookings, car rentals, and vacation packages, providing users with a one-stop solution for their travel needs.

  Partnerships with third-party services like travel agencies or tourism providers could enhance the system's offerings and attract a broader customer base.

- **Mobile Application Development:**

  To cater to the growing demand for mobile access, a dedicated mobile application for both iOS and Android platforms can be developed. This will provide customers with the convenience to book flights, manage reservations, check-in, and receive real-time updates on their mobile devices.

- **Personalized Recommendations Using AI and Machine Learning:**

The implementation of AI and machine learning algorithms can be incorporated to offer personalized flight recommendations based on users' past searches, preferences, and travel patterns.

AI-driven chatbots can be introduced to assist customers in real-time, answer queries, and help with booking, improving customer support and engagement.

- **Dynamic Pricing and Fare Management:**

Future versions of the system could implement dynamic pricing models, where flight prices fluctuate in real-time based on demand, booking trends, and market conditions, similar to what is seen in the hotel and ride-sharing industries.

A fare management system can be introduced to provide better pricing strategies, promotional discounts, and loyalty program integration for frequent flyers.

- **Enhanced Payment Options:**

The system can support a wider variety of payment gateways and methods, including mobile wallets, cryptocurrency payments, and international currency support, ensuring smoother and more flexible payment experiences for customers globally.

Integration with financial technologies (FinTech) can allow users to access payment installment options or other financing solutions.

- **Blockchain for Security and Transparency:**

Blockchain technology could be explored for secure and transparent management of bookings, payment transactions, and customer identity verification, offering enhanced protection against fraud and ensuring data integrity.

- **AI-Powered Customer Support and Virtual Assistants:**

Integrating an AI-powered virtual assistant for customer support can help with answering booking inquiries, updating reservations, managing cancellations, and providing flight status updates around the clock, reducing the need for human agents.

- **Real-time Flight Tracking and Notifications:**

Integration with global flight tracking systems to provide real-time flight status updates, weather conditions, and gate changes directly to users. Push notifications could be sent to customers to inform them of flight delays, cancellations, or gate changes.

- **Multilingual Support:**

Adding multilingual support would make the system accessible to a wider audience, particularly international customers, by offering the ability to switch between different languages and currencies.

- **Sustainability and Green Travel Features:**

As the aviation industry faces growing pressure to reduce carbon footprints, integrating features that highlight eco-friendly flight options (e.g., carbon offset programs or flights with lower emissions) could be a valuable addition to the system.

- **Advanced Reporting and Analytics for Administrators:**

Future versions of the system could include advanced analytics and reporting tools for airline staff to track booking trends, customer behavior, and operational performance, aiding in strategic decision-making and resource optimization.

- **Virtual Reality (VR) for Seat Selection and Cabin Preview:**

VR technology could be integrated to provide a virtual tour of the airplane’s cabin and seating options, allowing users to get a better idea of their seat selection before making a reservation.

- **Loyalty Program and Customer Engagement:**

A loyalty program could be incorporated, allowing customers to earn points or miles for bookings, which can be redeemed for discounts or upgrades. This feature could be extended to create personalized offers and promotions based on customer history and preferences.

APPENDIX A
(SOURCE CODE)

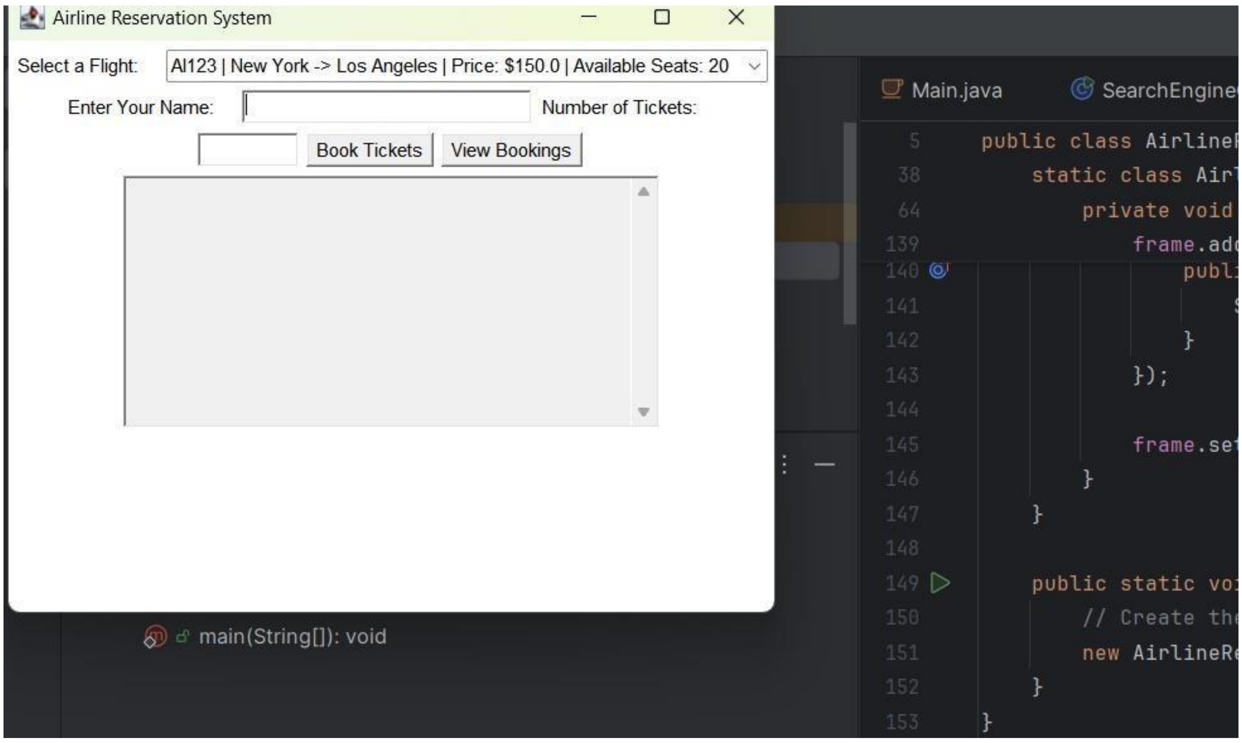
```
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class EnhancedAirlineReservationSystem {

    // Flight class to represent each flight's details
    static class Flight {
        String flightNumber;
        String source;
        String destination;
        int availableSeats;
        double price;

        public Flight(String flightNumber, String source, String destination, int
availableSeats, double price) {
            this.flightNumber = flightNumber;
            this.source = source;
            this.destination = destination;
            this.availableSeats = availableSeats;
            this.price = price;
        }

        // Method to book seats and reduce availability
        public boolean bookSeats(int numSeats) {
            if (availableSeats >= numSeats) {
                availableSeats -= numSeats;
                return true;
            }
            return false;
        }

        // Method to restore seats if a booking is canceled or modified
        public void cancelSeats(int numSeats) {
            availableSeats += numSeats;
        }

        // Method to display the flight's details
        @Override
        public String toString() {
            return flightNumber + " | " + source + " -> " + destination + " | Available Seats: " +
availableSeats + " | Price: $" + price;
        }
    }

    // Main GUI class
    static class ReservationApp {
        private Frame frame;
        private Choice flightChoice;
        private TextField nameField;
        private TextField numTicketsField;
        private TextArea bookingDetailsArea;
        private Button bookButton;
        private Button viewBookingsButton;

        private ArrayList<Flight> flights;
        private HashMap<String, String> userBookings; // To track bookings by user name
```

```
public ReservationApp() {
    // Initialize flight data
    flights = new ArrayList<>();
    flights.add(new Flight("AI123", "New York", "Los Angeles", 20, 150.0));
    flights.add(new Flight("AI456", "Chicago", "Miami", 25, 120.0));
    flights.add(new Flight("AI789", "San Francisco", "Seattle", 30, 100.0));

    userBookings = new HashMap<>();

    // Setup GUI
    setupGUI();
}

private void setupGUI() {
    frame = new Frame("Airline Reservation System");
    frame.setSize(500, 400);
    frame.setLayout(new FlowLayout());

    // Flight Selection
    Label flightLabel = new Label("Select a Flight:");
    flightChoice = new Choice();
    for (Flight flight : flights) {
        flightChoice.add(flight.toString());
    }

    // Name Entry
    Label nameLabel = new Label("Enter Your Name:");
    nameField = new TextField(20);

    // Ticket Count
    Label numTicketsLabel = new Label("Number of Tickets:");
```

```
numTicketsField = new TextField(5);

// Text Area to show booking details
bookingDetailsArea = new TextArea(10, 40);
bookingDetailsArea.setEditable(false);

// Buttons
bookButton = new Button("Book Tickets");
viewBookingsButton = new Button("View Bookings");

// Adding components to the frame
frame.add(flightLabel);
frame.add(flightChoice);
frame.add(nameLabel);
frame.add(nameField);
frame.add(numTicketsLabel);
frame.add(numTicketsField);
frame.add(bookButton);
frame.add(viewBookingsButton);
frame.add(bookingDetailsArea);

// Action listener for booking
bookButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String numTicketsText = numTicketsField.getText();
        if (name.isEmpty() || numTicketsText.isEmpty()) {
            bookingDetailsArea.append("Please fill in all fields.\n");
            return;
        }
```

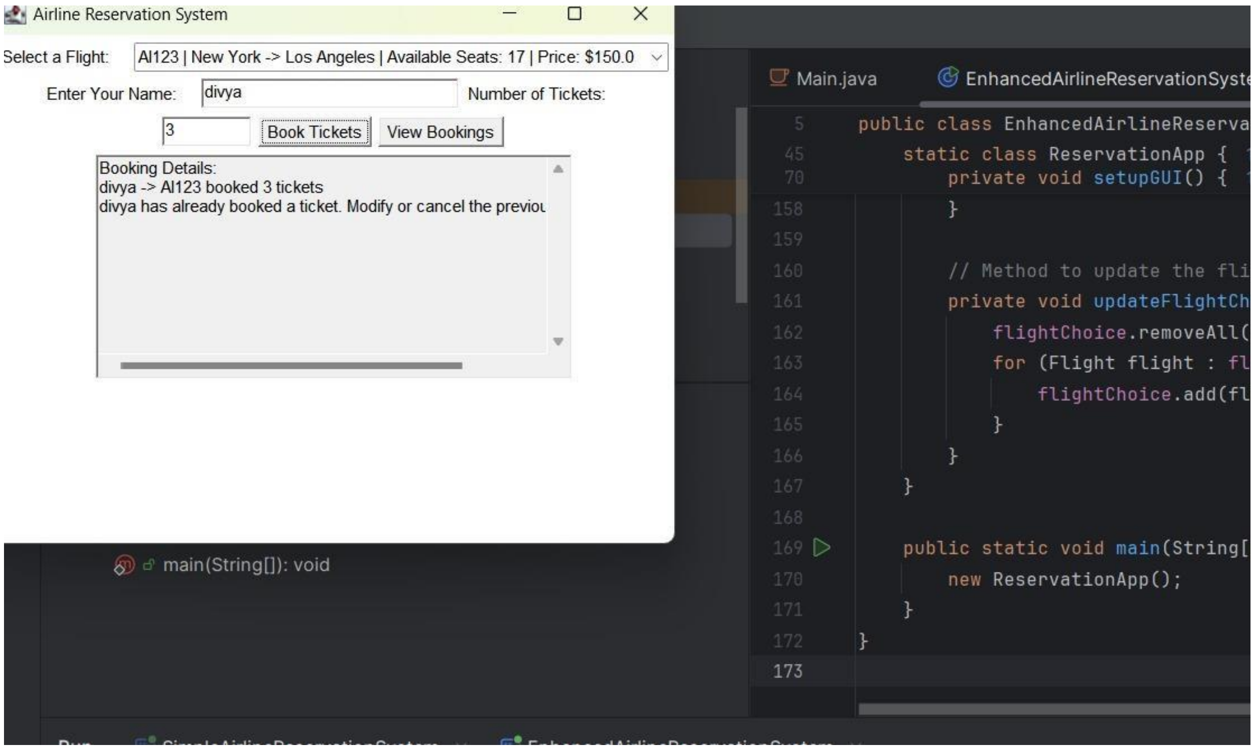
```
                    int numTickets = Integer.parseInt(numTicketsText);
                    Flight selectedFlight = flights.get(flightChoice.getSelectedIndex());

                    // Check if the user already has a booking
                    if (userBookings.containsKey(name)) {
                        bookingDetailsArea.append(name + " has already booked a ticket. Modify
or cancel the previous booking.\n");
                    } else {
                        // Proceed with booking if there are enough seats
                        if (selectedFlight.bookSeats(numTickets)) {
                            userBookings.put(name, selectedFlight.flightNumber + " booked " +
numTickets + " tickets");
                            bookingDetailsArea.append("Booking Successful! \n");
                            updateFlightChoice(); // Update the available flights after booking
                        } else {
                            bookingDetailsArea.append("Not enough seats available.\n");
                        }
                    }
                }
            });

            // Action listener for viewing bookings
            viewBookingsButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    bookingDetailsArea.setText("Booking Details:\n");
                    for (Map.Entry<String, String> entry : userBookings.entrySet()) {
                        bookingDetailsArea.append(entry.getKey() + " -> " + entry.getValue() +
"\n");
                    }
                }
            });

            // Window close event
            frame.addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent we) {
                    System.exit(0);
                }
            });

            frame.setVisible(true);
        }

        // Method to update the flight options after a booking
        private void updateFlightChoice() {
            flightChoice.removeAll();
            for (Flight flight : flights) {
                flightChoice.add(flight.toString());
            }
        }
    }

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

APPENDIX B
(SCREENSHOTS)



```
Select a Flight: AI123 | New York -> Los Angeles | Price: $150.0 | Available Seats: 20
Enter Your Name:                Number of Tickets:
```

```
public class AirlineR
    static class Air
        private void
            frame.add
                publi
                    S
                }
            });

            frame.se
        }
    }

    public static voi
        // Create the
        new AirlineRe
    }
}
```



```
Select a Flight: AI123 | New York -> Los Angeles | Available Seats: 17 | Price: $150.0
Enter Your Name: divya          Number of Tickets:
3

Booking Details:
divya -> AI123 booked 3 tickets
divya has already booked a ticket. Modify or cancel the previou
```

```
public class EnhancedAirlineReserva
    static class ReservationApp {
        private void setupGUI() {
        }

        // Method to update the fli
        private void updateFlightCh
            flightChoice.removeAll(
            for (Flight flight : fl
                flightChoice.add(fl
            }
        }
    }

    public static void main(String[
        new ReservationApp();
    }
}
```

REFERENCES

1. **https://aws.amazon.com/documentation/**
2. **https://www.geeksforgeeks.org/**
3. **https://github.com/**