



FLIGHT RESERVATION SYSTEM

Submitted by

JULIET ROSHAN J (231001080)

MUGUNTH B (231001124)

MAHESH BABU R(231001106)

MINI PROJECT REPORT

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this project report titled “**FLIGHT RESERVATION SYSTEM**” is the bonafide work of “**JULIET ROSHAN J(231001080)**”, “**MUGUNTH B(231001124)**”, “**MAHESH BABU R(231001106)**” who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.P.Valarmathie
Head of The Department
Department of Information Technology
Rajalakshmi Engineering College

SIGNATURE

Mrs.T.Sangeetha
Course Incharge
Department of Information Technology
Rajalakshmi Engineering College

Submitted to Project Viva-Voce Examination held on

Internal Examiner

External Examiner

ABSTRACT

The **Flight reservation system** is a **Java-based interface** designed to facilitate the Reservation of aviation operations, including flight scheduling, ticket booking, and passenger information handling, through an interactive graphical user interface (GUI) connected to a MySQL database using JDBC. This interface provides a seamless connection between the user and the underlying database, allowing for real-time updates and data retrieval. It eliminates manual record-keeping, reduces operational errors, and ensures data consistency across all operations.

The system is built with scalability in mind, ensuring it can handle growing data requirements as the airline expands. Additionally, the interface includes security features, such as role-based access and secure database connections, to protect sensitive information. The proposed system also offers reporting functionality to aid decision-making, with the potential for future integration with other systems, such as online booking platforms or payment gateways. This project serves as an efficient and effective interface for managing airplane operations while providing practical experience in Java programming and database management.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	3
1	INTRODUCTION	
	1.1 Motivation	5
	1.2 Existing System	5
	1.3 Project Objectives	5
	1.4 Proposed System	6
2	SYSTEM DESIGN	
	2.1 Introduction	7
	2.2 System Architecture	8
	2.3 System Requirements	10
3	PROJECT DESCRIPTION 3.1 Methodologies 3.2 Module Description	12
4	Results and Discussion	15
	CONCLUSION	

CHAPTER - 1

INTRODUCTION

1.1 MOTIVATION

The aviation industry involves complex operations such as flight scheduling, ticket booking, and passenger management. Creating an efficient, computerized system helps streamline these processes, reducing manual errors and improving overall efficiency. This project serves as a practical implementation of Java's JDBC framework and GUI design principles. It provides an opportunity to learn database connectivity and query execution in real-world applications. Developing a database-backed application provides a scalable solution for managing increasing data volumes in aviation. It also sets the foundation for future enhancements, such as online ticket booking or integration with third-party APIs. The project addresses common challenges in aviation management, such as data retrieval, updating schedules, and ensuring data consistency, showcasing how technology can solve industry-specific problems.

1.2 EXISTING SYSTEM

In traditional systems, flight schedules, passenger information, and ticketing are recorded manually on paper or spreadsheets. This is prone to errors, duplication, and inefficiency. Critical tasks like flight rescheduling, passenger updates, and ticket cancellations require significant human intervention, leading to delays and inaccuracies. Some systems use rudimentary desktop applications with limited functionality, lacking centralized databases and often requiring redundant data entry. Basic digital systems may not support real-time updates, making it hard to manage dynamic changes, such as delays or rescheduling.

1.3 PROJECT OBJECTIVES

The objective of the Flight reservation system project is to develop a centralized, efficient, and user-friendly platform for managing aviation operations such as flight scheduling, ticket booking, and passenger data. By utilizing Java's JDBC framework and a MySQL database,

the system ensures real-time data access, robust security, and minimized human errors. Additionally, it offers reporting features for better decision-making and serves as a foundation for future integration with online booking systems and other advanced functionalities, combining practical application with hands-on learning.

1.4 PROPOSED SYSTEM

The proposed Flight reservation system is a Java-based application integrated with a MySQL database, designed to streamline and automate critical aviation operations. This system introduces a user-friendly graphical interface (GUI) for managing flight schedules, ticket bookings, and passenger data efficiently. Unlike traditional manual or basic digital systems, it provides a centralized database to ensure consistency, eliminate redundancy, and facilitate real-time data updates. The system automates tasks such as flight scheduling, passenger management, and ticket issuance, significantly reducing manual errors and operational delays. It incorporates robust security features, including role-based access and secure database connections, to safeguard sensitive data. Scalability is a core feature, allowing the system to accommodate increased data loads as operations grow. Additionally, the proposed system generates detailed reports for decision-making and supports dynamic adaptability for future integrations, such as online booking portals or payment gateways. By leveraging modern technologies and automation, the proposed system aims to deliver a reliable, efficient, and user-centric solution for aviation management.

CHAPTER - 2

SYSTEM DESIGN

2.1 INTRODUCTION

The **Flight reservation system** is a comprehensive software solution designed to streamline and automate the operational, administrative, and customer-facing processes of an airline. With the rapid growth of the aviation industry and the increasing complexity of managing large-scale operations, such systems play a pivotal role in ensuring efficiency, accuracy, and customer satisfaction.

Airline operations involve a diverse range of activities, including flight scheduling, ticket reservation, passenger and crew management, aircraft maintenance, and financial transactions. Traditionally, these tasks were handled manually or through disparate systems, leading to inefficiencies, delays, and potential errors. The Flight reservation system addresses these challenges by integrating all essential functionalities into a unified platform, enabling real-time data processing, operational transparency, and seamless coordination across departments.

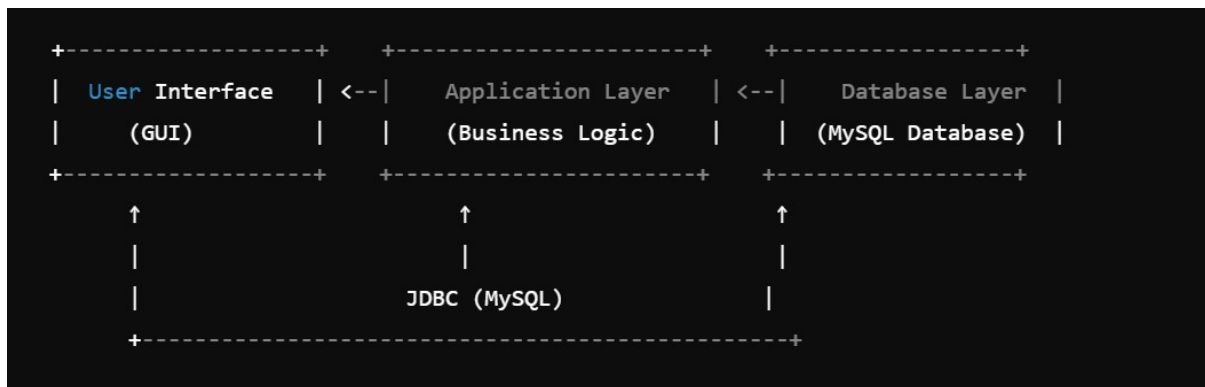
Key features of such systems include automated ticket booking, real-time flight status updates, secure payment processing, and advanced analytics for decision-making. They also incorporate robust security measures to protect sensitive customer data and comply with regulatory requirements. Modern Airline leverage cutting-edge technologies such as cloud computing, artificial intelligence, and big data analytics to deliver scalable and adaptive solutions. In addition to operational benefits, these systems enhance the passenger experience by providing intuitive interfaces for booking and managing travel, personalized recommendations, and prompt communication regarding flight changes or delays. For airlines, the system reduces operational costs, improves resource utilization, and ensures compliance with industry standards.

The Flight reservation system thus serves as the backbone of efficient airline operations, supporting growth in a competitive and fast-evolving industry.

In addition to enhancing daily operations, the system is scalable, adaptable, and secure. It is designed to accommodate growing data requirements as the airline expands, while ensuring the safety of sensitive information through role-based access control and secure database connections. With the potential for future enhancements and integration with other systems, this Airplane Management System serves as a practical, modern solution to the evolving needs of the aviation industry.

Traditionally, these operations were handled manually or through outdated systems, leading to inefficiencies, errors, and a lack of real-time data access. This system introduces an innovative approach by offering a centralized and automated solution that connects a user-friendly interface with a MySQL database, using Java's JDBC framework. The system's primary objective is to simplify and optimize aviation Reservation tasks, such as flight schedule updates, ticket booking, and passenger data management, through **an intuitive graphical user interface (GUI)**

2.2 SYSTEM ARCHITECTURE



1. User Interface Layer (Client-side)

- **Description:**
The Graphical User Interface (GUI) is the front-end component that allows users (e.g., airport staff, booking agents) to interact with the system. It is developed using Java Swing or JavaFX for a seamless user experience.
- **Components:**
 - Login/Authentication Window: Secure login for authorized users.
 - Dashboard: Displays key information such as flight schedules, passenger details, and booking status.
 - Flight and Passenger Management Forms: Allows users to add, update, and view flight details and passenger records.
 - Reports Section: Allows users to generate and view reports on flight schedules, bookings, and passenger lists.
- **Purpose:**
The GUI simplifies user interaction, allowing easy navigation and data management without technical expertise.

2. Application Layer (Server-side)

- **Description:**
The Server-side Logic processes user requests from the GUI, communicates with the database, and sends responses back to the client. It handles the business logic and performs actions like adding flight records, booking tickets, and generating reports.
- **Components:**
 - Controller Classes: Handle user requests and implement actions such as adding flights, updating schedules, and booking tickets.

- Validation Logic: Ensures that data entered by users is correct (e.g., checking for valid dates, available seats).
- Data Processing: Processes the data for report generation or updates to the database.
- Purpose:
The application layer encapsulates the business logic and ensures data integrity and validation before interacting with the database.

3. Database Layer (Backend)

- Description:
The Database layer stores all the persistent data related to flights, passengers, bookings, and schedules. The system uses MySQL as the database management system (DBMS), which is connected via JDBC.
- Components:
 - Flight Table: Stores information related to flights, including flight ID, flight number, departure/arrival details, and status.
 - Passenger Table: Stores passenger details such as name, contact information, and booking ID.
 - Booking Table: Stores information about ticket bookings, including booking ID, passenger ID, and flight ID.
 - Reports Data: The database also maintains historical data, which is used for generating reports on bookings, flight schedules, etc.
- Purpose:
The database layer ensures data consistency, allows for efficient querying, and serves as the main repository for the system's information.

4. Communication Layer

- Description:
The JDBC (Java Database Connectivity) API facilitates communication between the application layer and the database layer. It ensures that the application can execute SQL queries and interact with the database.
- Components:
 - JDBC Driver: Used to connect the Java application to the MySQL database.
 - SQL Queries: Used for retrieving, inserting, updating, and deleting data in the database (e.g., adding a new flight, booking a ticket).

2.3 SYSTEM REQUIREMENTS

1. Functional Requirements

The functional requirements for the Flight reservation system include various key features. **User Authentication and Authorization** ensures secure login with role-based access for

passengers, administrators, and staff. **Flight Reservation** enables the addition, updating, and management of flight schedules, routes, and availability. The **Ticket Reservation and Booking** feature allows users to search for flights, book tickets, select seats, and cancel reservations. **Real-Time Updates** keep users informed about flight delays, cancellations, and gate changes.

The system includes **Payment Processing** that supports multiple payment methods like credit/debit cards, net banking, and wallets. **Customer Management** tracks user profiles, preferences, and booking history to enhance service. The **Admin Dashboard** allows administrators to generate reports, analyze performance metrics, and manage staff and flight operations. Lastly, **Feedback and Support** enables users to submit feedback and access customer support.

2. Non-Functional Requirements

The system should be capable of handling high transaction volumes during peak hours without delays, ensuring **performance**. It must also be **scalable** to accommodate increasing user traffic and future feature expansions. **Availability** is crucial, with a target uptime of 99.9% for critical functions like ticket booking. The system should implement robust **security** measures, including encryption, secure authentication, and regular audits to protect user data. **Usability** is important, with a user-friendly interface that is accessible across both web and mobile platforms. Lastly, the system must ensure **compliance** with regulatory requirements such as GDPR for data protection and relevant industry safety standards.

HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

Server Hardware:

1. **Processor:** Multi-core processor (Intel i7 or higher / AMD Ryzen 7 or higher)
2. **RAM:** Minimum of 16 GB (32 GB or more for high traffic systems)
3. **Storage:** At least 500 GB SSD (scalable storage as needed for database and logs)
4. **Network:** High-speed internet connection with backup for continuous operations
5. **Backup:** Regular and redundant backup systems for data security

Client Hardware:

1. **User Devices:** Desktop or laptop computers with at least 4 GB RAM and modern browsers
2. **Mobile Devices:** Smartphones or tablets (Android/iOS) for mobile app usage
3. **Printers:** For ticket printing, boarding passes, and other physical documentation

Network Infrastructure:

1. High-bandwidth network connections for handling online reservations and real-time updates
2. Load balancing and firewall to ensure smooth traffic management and security

Software Requirements:

Operating System

1. **Server Side:** Linux (Ubuntu, CentOS, or Debian) or Windows Server (depending on chosen tech stack)
2. **Client Side:** Windows, macOS, or Linux for desktop users; Android and iOS for mobile applications

Database Management System (DBMS):

1. **Relational Databases:** MySQL, PostgreSQL, or Oracle for managing

Web and Application Development Frameworks:

1. **Backend Frameworks:** Java (Spring Boot), Python (Django, Flask), or Node.js (Express)
2. **Frontend Frameworks:** React.js, Java FX for web interfaces

Security Software:

1. **SSL/TLS** for encryption of data transmission
2. **Firewall** and **IDS/IPS** systems for network security
3. **Multi-factor Authentication (MFA)** for secure user logins

Development and Testing Tools:

1. IDEs like Visual Studio Code, IntelliJ IDEA, or Eclipse for development
2. Version control with Git and GitHub/GitLab for collaboration
3. Automated testing tools like Selenium, JUnit, or Mocha for quality assurance

CHAPTER - 3

PROJECT DESCRIPTION

3.1 METHODOLOGIES

The development of the **Airplane Reservation System** follows a structured approach to ensure efficient project management, quality, and functionality. The methodologies used are based on **Object-Oriented Programming (OOP)** principles, **Agile Development**, and **Database-Driven Development**, incorporating a combination of systematic planning, iterative progress, and user feedback.

1. Object-Oriented Programming (OOP) Methodology

- **Overview:**
OOP is the foundation of the system's design. The application is developed using object-oriented principles to model real-world entities like flights, passengers, and bookings. Each of these entities is represented by classes, with attributes and methods that define their properties and behaviors.
- **Key Features:**
 - **Encapsulation:** Data (attributes) and operations (methods) are bundled into classes, ensuring data security and easy maintenance.
 - **Inheritance:** Allows the creation of reusable code through the inheritance of common features, promoting code reusability.
 - **Polymorphism:** The system supports polymorphism, allowing methods to take many forms and adapt to different use cases, such as updating different flight information.
 - **Abstraction:** The complex underlying details of the system are hidden from users, simplifying interactions through a clean, understandable interface.
- **Purpose:**
OOP methodology ensures that the system is modular, maintainable, and easy to scale. It also helps in creating a system that models real-world scenarios accurately.

2. Agile Development Methodology

- **Overview:**
Agile focuses on iterative development, flexibility, and continuous feedback. The development of the Airplane Management System follows an Agile approach, where the project is broken down into smaller tasks or "sprints," each addressing a specific feature or functionality.
- **Key Features:**

- **Iterative Progress:** The system is developed incrementally, with each sprint building upon the previous one, allowing for quick adjustments based on user feedback.
- **Collaboration and Communication:** Regular communication between developers, stakeholders, and end-users ensures that the system meets the needs of all involved parties.
- **Adaptability:** Agile allows for flexibility in changing requirements, enabling developers to respond to new insights or technological advancements.
- **Frequent Releases:** At the end of each sprint, a working version of the system is released, ensuring that progress is continuously evaluated.
- **Purpose:**
Agile methodology ensures that the system is developed in a flexible, adaptive environment, where changes and improvements can be made based on real-time feedback, leading to a product that better serves the end users' needs.

3. Database-Driven Development

- **Overview:**
The Airplane Reservation System heavily relies on a MySQL database to store, retrieve, and manage data. The database-driven development approach focuses on designing and structuring the database to ensure efficient data handling.
- **Key Features:**
 - **Database Design:** A normalized database schema is created to store flight details, passenger information, and booking records. The relationships between entities are carefully modeled to ensure data integrity.
 - **SQL Queries:** The system uses SQL queries for CRUD operations (Create, Read, Update, Delete) to interact with the database.
 - **JDBC Integration:** The Java application communicates with the MySQL database using JDBC to fetch, update, and delete data in real-time.
 - **Data Integrity and Security:** Constraints, primary keys, and foreign keys are implemented to ensure consistency and integrity of the stored data. Role-based access control is used to protect sensitive information.
- **Purpose:**
A database-driven approach ensures that the system is capable of handling large volumes of structured data, with efficient storage, retrieval, and reporting capabilities.

4. Prototype Model (for User Interface Design)

- **Overview:**
In this methodology, a **prototype** of the GUI is developed early in the project to demonstrate the functionality and gather user feedback. The interface evolves based on continuous interaction and testing.
- **Key Features:**
 - **Rapid Development:** A working prototype is quickly built, allowing stakeholders to interact with the system early in the development process.
 - **User Feedback:** Users provide feedback on the interface's usability, which is used to refine the design.
 - **Iterative Refinement:** The prototype is iteratively refined to enhance user experience and meet requirements.
- **Purpose:**
The prototype model ensures that the GUI is user-centric, intuitive, and meets the practical needs of users by involving them early and continuously throughout the development process.

3.2 MODULE DESCRIPTION

The Flight reservation system consists of several interrelated modules designed to ensure efficient operations and enhance the user experience. The **User Management Module** handles passenger and staff registration, login, and profile management, with role-based access control for different users, including administrators and staff. The **Flight Reservation Module** manages flight scheduling, routes, and availability while handling updates such as delays, cancellations, and rescheduling.

The **Reservation and Ticketing Module** facilitates booking, cancellation, and rebooking, generating electronic tickets and allowing for seat selection. The **Payment Processing Module** integrates secure payment gateways for transactions, manages refunds, and maintains billing records. The **Customer Support Module** provides assistance via FAQs, complaint management, and live chat or automated responses to passenger queries.

The **Crew and Staff Management Module** schedules pilots, flight attendants, and ground crew, and tracks their availability, shifts, and performance. The **Aircraft Maintenance and Operations Module** ensures proper maintenance scheduling and compliance with safety regulations. The **Real-Time Notification Module** sends updates on flight statuses, gate changes, and promotional offers via email or SMS.

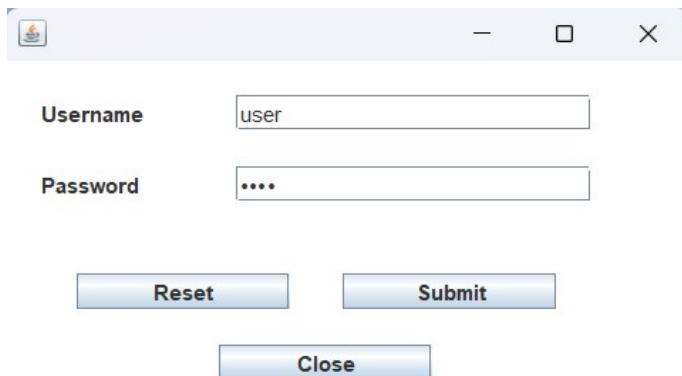
The **Admin Dashboard Module** provides tools for managing operations, analytics, and user data, while also generating reports for decision-making. The **Loyalty and Reward Program Module** tracks frequent flyer miles and reward points, offering redemption options and membership benefits. Lastly, the **Security Module** ensures robust user authentication, encryption, data protection, and fraud monitoring.

CHAPTER - 4

RESULTS AND DISCUSSION

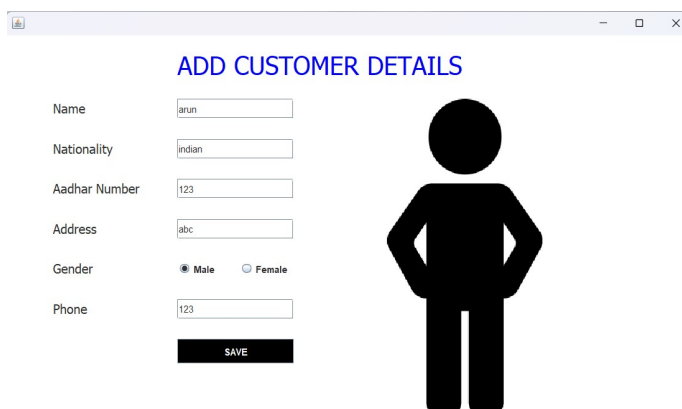
DESIGN

1. Login page




A screenshot of a web browser window displaying a login form. The window has a title bar with a minimize, maximize, and close button. The form contains two input fields: 'Username' with the text 'user' and 'Password' with four dots. Below the fields are three buttons: 'Reset', 'Submit', and 'Close'.

2. Home page



A screenshot of a web browser window displaying a form titled 'ADD CUSTOMER DETAILS'. The form includes input fields for 'Name' (arun), 'Nationality' (indian), 'Aadhar Number' (123), 'Address' (abc), 'Gender' (Male selected), and 'Phone' (123). A 'SAVE' button is at the bottom. To the right of the form is a black silhouette of a person standing with hands on hips.

3. Customer Details


— □ ×

PNR Details

Show Details

PNR	TICKET	aadhar	name	nationality	flightname	flightcode	src	des	ddate
PNR-958640	TIC-3261	364646251089	Audi	INDIAN	AI-1212	1001	Delhi	Mumbai	22 Nov 202.

4.PNR Details

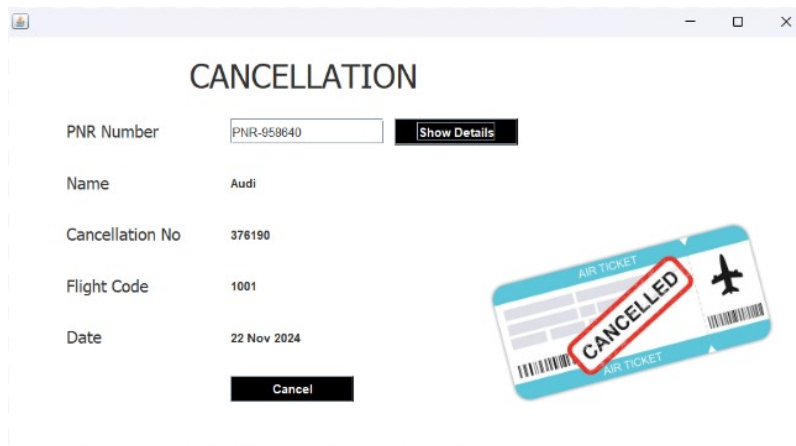
[Home](#)
[Dashboard](#)
[Logout](#)

LIST OF BOOKED TICKETS FOR THE FLIGHT

PNR	Date of Reservation	Class	No. of Passengers	Payment ID	Customer ID
9497836	2019-08-16	business	1	470360387	Marry
6845456	2019-08-16	economy	2	303632303	Marry

5.

Boarding pass



6. Cancellation

IMPLEMENTATION (CODE)

```
package flightreservationsystem;
```

```
import javax.swing.*.*;
```

```
import java.awt.*.*;
```

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

```
public class AddCustomer extends JFrame implements ActionListener{
```

```
    JTextField tfname, tfphone, tfaadhar, tfnationality, tfaddress;
```

```
    JRadioButton rbmale, rbfemale;
```

```
    public AddCustomer() {
```

```
        getContentPane().setBackground(Color.WHITE);
```

```
        setLayout(null);
```

```
        JLabel heading = new JLabel("ADD CUSTOMER DETAILS");
```

```
        heading.setBounds(220, 20, 500, 35);
```

```
        heading.setFont(new Font("Tahoma", Font.PLAIN, 32));
```

```
heading.setForeground(Color.BLUE);  
add(heading);
```

```
JLabel lblname = new JLabel("Name");  
lblname.setBounds(60, 80, 150, 25);  
lblname.setFont(new Font("Tahoma", Font.PLAIN, 16));  
add(lblname);
```

```
tfname = new JTextField();  
tfname.setBounds(220, 80, 150, 25);  
add(tfname);
```

```
JLabel lblnationality = new JLabel("Nationality");  
lblnationality.setBounds(60, 130, 150, 25);  
lblnationality.setFont(new Font("Tahoma", Font.PLAIN, 16));  
add(lblnationality);
```

```
tfnationality = new JTextField();  
tfnationality.setBounds(220, 130, 150, 25);  
add(tfnationality);
```

```
JLabel lblaadhar = new JLabel("Aadhar Number");  
lblaadhar.setBounds(60, 180, 150, 25);  
lblaadhar.setFont(new Font("Tahoma", Font.PLAIN, 16));  
add(lblaadhar);
```

```
tfaadhar = new JTextField();  
tfaadhar.setBounds(220, 180, 150, 25);  
add(tfaadhar);
```

```
JLabel lbladdress = new JLabel("Address");
```

```
lbladdress.setBounds(60, 230, 150, 25);  
lbladdress.setFont(new Font("Tahoma", Font.PLAIN, 16));  
add(lbladdress);
```

```
tfaddress = new JTextField();  
tfaddress.setBounds(220, 230, 150, 25);  
add(tfaddress);
```

```
JLabel lblgender = new JLabel("Gender");  
lblgender.setBounds(60, 280, 150, 25);  
lblgender.setFont(new Font("Tahoma", Font.PLAIN, 16));  
add(lblgender);
```

```
ButtonGroup gendergroup = new ButtonGroup();
```

```
rbmale = new JRadioButton("Male");  
rbmale.setBounds(220, 280, 70, 25);  
rbmale.setBackground(Color.WHITE);  
add(rbmale);
```

```
rbfemale = new JRadioButton("Female");  
rbfemale.setBounds(300, 280, 70, 25);  
rbfemale.setBackground(Color.WHITE);  
add(rbfemale);
```

```
gendergroup.add(rbmale);  
gendergroup.add(rbfemale);
```

```
JLabel lblphone = new JLabel("Phone");  
lblphone.setBounds(60, 330, 150, 25);  
lblphone.setFont(new Font("Tahoma", Font.PLAIN, 16));
```

```

add(lblphone);

tfphone = new JTextField();
tfphone.setBounds(220, 330, 150, 25);
add(tfphone);

JButton save = new JButton("SAVE");
save.setBackground(Color.BLACK);
save.setForeground(Color.WHITE);
save.setBounds(220, 380, 150, 30);
save.addActionListener(this);
add(save);

ImageIcon image = new
ImageIcon(ClassLoader.getResource("flightReservationsystem/icons/emp.png"));
JLabel lblimage = new JLabel(image);
lblimage.setBounds(450, 80, 280, 400);
add(lblimage);

setSize(900, 600);
setLocation(300, 150);
setVisible(true);
}

public void actionPerformed(ActionEvent ae) {
    String name = tfname.getText();
    String nationality = tfnationality.getText();
    String phone = tfphone.getText();
    String address = tfaddress.getText();
    String aadhar = tfaadhar.getText();
    String gender = null;

```

```

        if (rbmale.isSelected()) {
            gender = "Male";
        } else {
            gender = "Female";
        }

        try {
            Conn conn = new Conn();

            String query = "insert into passenger values('"+name+"', '"+nationality+"', '"+phone+"',
            '"+address+"', '"+aadhar+"', '"+gender+"')";

            conn.s.executeUpdate(query);

            JOptionPane.showMessageDialog(null, "Customer Details Added Successfully");

            setVisible(false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

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

2.boarding pass
package flightreservationsystem;

import javax.swing.*;
import java.awt.*;

```

```

import java.awt.event.*;
import java.sql.*;

public class BoardingPass extends JFrame implements ActionListener {

    JTextField tfpnr;
    JLabel tfname, tfnationality, lblsrc, lbldest, labelfname, labelfcode, labeldate;
    JButton fetchButton;

    public BoardingPass() {
        getContentPane().setBackground(Color.WHITE);
        setLayout(null);

        JLabel heading = new JLabel("AIR INDIA");
        heading.setBounds(380, 10, 450, 35);
        heading.setFont(new Font("Tahoma", Font.PLAIN, 32));
        add(heading);

        JLabel subheading = new JLabel("Boarding Pass");
        subheading.setBounds(360, 50, 300, 30);
        subheading.setFont(new Font("Tahoma", Font.PLAIN, 24));
        subheading.setForeground(Color.BLUE);
        add(subheading);

        JLabel lblaadhar = new JLabel("PNR DETAILS");
        lblaadhar.setBounds(60, 100, 150, 25);
        lblaadhar.setFont(new Font("Tahoma", Font.PLAIN, 16));
        add(lblaadhar);

        tfpnr = new JTextField();
        tfpnr.setBounds(220, 100, 150, 25);

```

```
add(tfpnr);
```

```
fetchButton = new JButton("Enter");  
fetchButton.setBackground(Color.BLACK);  
fetchButton.setForeground(Color.WHITE);  
fetchButton.setBounds(380, 100, 120, 25);  
fetchButton.addActionListener(this);  
add(fetchButton);
```

```
        e.printStackTrace();  
    }  
} else {  
    Random random = new Random();  
  
    String aadhar = tfaadhar.getText();  
    String name = tfname.getText();  
    String nationality = tfnationality.getText();  
    String flightname = labelfname.getText();  
    String flightcode = labelfcode.getText();  
    String src = source.getSelectedItem();  
    String des = destination.getSelectedItem();  
    String ddate = ((JTextField) dcdate.getDateEditor().getUiComponent()).getText();  
  
    try {  
        Conn conn = new Conn();  
  
        String query = "insert into reservation values('PNR-"+random.nextInt(1000000)+"",  
'TIC-"+random.nextInt(10000)+"', '"+aadhar+"', '"+name+"', '"+nationality+"',  
        '"+flightname+"', '"+flightcode+"', '"+src+"', '"+des+"', '"+ddate+"')";
```

```

        conn.s.executeUpdate(query);

        JOptionPane.showMessageDialog(null, "Ticket Booked Successfully");

        setVisible(false);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}

}

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

}

5.login
package flightreservationsystem;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class Login extends JFrame implements ActionListener{
    JButton submit, reset, close;
    JTextField tfusername;
    JPasswordField tfpassword;

    public Login() {
        getContentPane().setBackground(Color.WHITE);
        setLayout(null);
    }
}

```



```

JLabel lblusername = new JLabel("Username");
lblusername.setBounds(20, 20, 100, 20);
add(lblusername);

tfusername = new JTextField();
tfusername.setBounds(130, 20, 200, 20);
add(tfusername);

boardingPass.addActionListener(this); // Add ActionListener to Boarding Pass
ticket.add(boardingPass);

// Set Frame Properties
setExtendedState(JFrame.MAXIMIZED_BOTH);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent ae) {
    String text = ae.getActionCommand();

    if (text.equals("Add Customer Details")) {
        new AddCustomer();
    } else if (text.equals("Flight Details")) {
        new FlightInfo();
    } else if (text.equals("Book Flight")) {
        new BookFlight();
    } else if (text.equals("Journey Details")) {
        new JourneyDetails();
    } else if (text.equals("Cancel Ticket")) {
        new Cancel();
    }
}

```

```

    } else if (text.equals("Boarding Pass")) {
        new BoardingPass(); // Create and display the Boarding Pass window
    }
}

```

```

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

```

6.Flight details:

```

package flightreservationsystem;

```

```

import javax.swing.*;
import java.awt.*;
import java.sql.*;
import net.proteanit.sql.DbUtils;

```

```

public class FlightInfo extends JFrame{

```

```

    public FlightInfo() {

```

```

        getContentPane().setBackground(Color.WHITE);
        setLayout(null);

```

```

        JTable table = new JTable();

```

```

        try {
            Conn conn = new Conn();

```

```

            ResultSet rs = conn.s.executeQuery("select * from flight");
            table.setModel(DbUtils.resultSetToTableModel(rs));

```

```
    } catch(Exception e) {  
        e.printStackTrace();  
    }  
}
```

```
JScrollPane jsp = new JScrollPane(table);  
jsp.setBounds(0, 0, 800, 500);  
add(jsp);
```

```
setSize(800, 500);  
setLocation(400, 200);  
setVisible(true);  
}
```

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

7.Connection:

```
package flightreservationsystem;
```

```
import java.sql.*;
```

```
public class Conn {
```

```
    Connection c;
```

```
    Statement s;
```

```
    public Conn() {
```

```
        try {
```

```
            Class.forName("com.mysql.cj.jdbc.Driver");
```

```
        c = DriverManager.getConnection("jdbc:mysql:///flightreservationsystem", "root",
"0409");
        s = c.createStatement();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

CONCLUSION

Conclusion

The Flight reservation system project successfully demonstrates the integration of modern software development practices to address the complexities of managing aviation operations. By leveraging Java, JDBC, and MySQL, the system provides an efficient, automated solution for managing flight schedules, passenger information, and ticketing processes, all through a user-friendly GUI interface. The use of Object-Oriented Programming (OOP) ensures a

modular and scalable design, while the Agile methodology allows for continuous feedback and iterative development, ensuring the system meets both user needs and technical requirements.

The project effectively eliminates the inefficiencies and errors associated with manual Reservation, providing real-time data access and ensuring data consistency. The system's design is scalable, enabling it to handle growing amounts of data as the airline expands. Additionally, the implementation of robust security features ensures that sensitive passenger and flight data is protected.

Through the creation of this system, valuable hands-on experience in Java programming, database Reservation, and system design was gained, providing a strong foundation for future development projects. Overall, the Flight reservation system stands as a reliable, adaptable, and efficient solution for modernizing aviation operations, offering both operational benefits and technical insights.