

**AIRLINE MANAGEMENT SYSTEM**

**PROJECT REPORT**

*Submitted by*

**VIGNESH ANAND [RegNo: RA2111003010674]**

*Under the Guidance of*

**Dr. R. Subash**

**Assistant Professor, Computing Technologies**

*In partial satisfaction of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**

**with specialization in Computer Science and Engineering**



**SCHOOL OF COMPUTING**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**April 2024**



**SRM**  
INSTITUTE OF SCIENCE & TECHNOLOGY  
Deemed to be University u/s 3 of UGC Act, 1956

**SRM INSTITUTION OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR-603203**

**BONAFIDE CERTIFICATE**

Certified that this lab report titled Airline Management System is the bonafide work done by Vignesh Anand (RA2111003010674) who carried out the lab exercises under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

*R. Subash*  
7/5/24

**SIGNATURE**

Dr. R. Subash  
Assistant Professor  
Computing Technologies



*M. Pushpalatha*

**SIGNATURE**

Dr. Pushpalatha M.  
Head of the Department  
Computing Technologies

## **TABLE OF CONTENTS**

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>3</b>
<b>2</b>	<b>PROJECT FEATURES AND OBJECTIVES</b>	<b>6</b>
<b>3</b>	<b>FRONTEND DEVELOPMENT</b>	<b>9</b>
<b>4</b>	<b>BACKEND DEVELOPMENT</b>	<b>14</b>
<b>5</b>	<b>MODULE AND APPLICATION</b>	<b>18</b>
	<b>CONCLUSION</b>	<b>20</b>
	<b>REFERENCES</b>	<b>21</b>

# **Chapter 1**

## **Introduction**

### **1.1 Background**

An airline management system (AMS) is a software application that helps airlines manage various aspects of their operations. It streamlines processes, improves efficiency, and enhances customer service.

### **1.2 Purpose of the Project**

The primary objective of this project has shifted to develop a robust Airline Management System using Java. By leveraging the capabilities of JDBC connector for database management and a suitable UI framework (e.g., Swing, JavaFX) for creating a user-friendly graphical interface, the aim is to streamline various administrative tasks associated with airline operations.

### **1.3 Project Scope**

The scope of this project encompasses the design and implementation of a comprehensive airline management system capable of handling key aspects of airline operations, including but not limited to:

- **Flight Management:** Create, modify, and view flight schedules, routes, aircraft details, and crew assignments.
- **Reservation & Ticketing:** Manage passenger reservations, ticketing process, seat selection, and fare calculations.
- **Inventory Control:** Track available seats on flights, manage waitlists, and handle overbooking scenarios.
- **Customer Relationship Management (CRM):** Manage passenger information, preferences, loyalty programs, and communication.

## 1.4 Project Overview

The project will consist of two core components: a relational database and a user-friendly graphical user interface (GUI).

- **Backend Database:** The backend will leverage Java Database Connectivity (JDBC) to interact with a relational database management system (RDBMS) of your choice (e.g., MySQL, PostgreSQL, Oracle). JDBC provides a standard interface for connecting to various databases from Java applications.
- **Frontend GUI:** The GUI will be developed using NetBeans. NetBeans is a free and open-source Integrated Development Environment (IDE) that provides a robust platform for building Java applications with graphical interfaces. NetBeans offers visual tools and features to simplify GUI development, including drag-and-drop components and visual form editing.

This combination of JDBC for database interaction and NetBeans for GUI development provides a powerful and user-friendly approach for building the airline management system.

## 1.5 Report Structure

This report is structured to provide a comprehensive overview of the Airline Management System project. Following this introduction, subsequent sections will delve into the detailed design and implementation of both the backend and frontend components. Additionally, the report will discuss the functionalities, features, and user interactions of the system, along with insights into the development process and potential areas for future enhancement.



## **Chapter 2**

### **Project Features and Objectives**

#### **2.1 Features**

##### **2.1.1 Flight Management**

- Create, modify, and view flight schedules, including departure and arrival times, routes, and aircraft details.
- Manage flight crew assignments, including pilots, cabin crew, and other personnel.
- Define and manage different fare classes (e.g., economy, business, first class) for each flight.

##### **2.1.2 Reservation & Ticketing**

- Allow passengers to search for flights based on various criteria (destination, date, price, etc.).
- Facilitate online booking of tickets, including seat selection and add-on services (e.g., baggage allowance, meals).
- Manage reservation changes and cancellations.
- Generate e-tickets and boarding passes for confirmed reservations.

##### **2.1.3 Inventory Control**

- Track real-time availability of seats on each flight.
- Manage waitlists for oversold flights.
- Implement automated notifications for waitlisted passengers when seats become available.

##### **2.1.4 Customer Relationship Management (CRM)**

- Manage passenger profiles, including contact information, travel preferences, and loyalty program details.

- Send personalized communication to passengers about flight confirmations, special offers, and loyalty program updates.
- Track passenger feedback and complaints for continuous improvement.

### **2.1.5 Additional Functionalities**

- Baggage handling: Manage checked and carry-on baggage allowance, weight restrictions, and fees.
- Cargo management: Track and manage cargo shipments on passenger and dedicated cargo flights.
- Reporting and analytics: Generate reports on flight occupancy, revenue, passenger demographics, and other key metrics to inform business decisions.

## **2.2 Objectives**

### **2.2.1 Efficiency and Streamlining**

The primary objective of the Airline Management System is to enhance efficiency by streamlining administrative tasks related to airline management. By automating processes such as data entry, passenger tracking, and performance evaluation, the system reduces manual workload and improves productivity.

### **2.2.2 Accuracy and Data Integrity**

Another key objective is to ensure accuracy and data integrity in passenger records and their information. By centralizing data storage and implementing validation mechanisms, the system minimizes errors and discrepancies, providing reliable information for decision-making purposes.

### **2.2.3 Accessibility and User-Friendliness**

The system aims to be accessible and user-friendly, catering to the diverse needs of administrators, crew, and passengers. Intuitive interfaces, simplified workflows, and clear navigation enhance usability, promoting adoption and acceptance among users.

### **2.2.4 Security and Confidentiality**

Ensuring the security and confidentiality of student data is paramount. The system employs robust authentication mechanisms, encryption techniques, and access controls to safeguard sensitive information, complying with relevant privacy regulations and standards.

### **2.2.5 Scalability and Flexibility**

As airlines evolve and grow, the Airline Management System should be scalable and adaptable to accommodate changing requirements. The system architecture is designed to support scalability, allowing for seamless integration of additional features and modules as needed.



## **Chapter 3**

### **Frontend Development**

The frontend of the Airline Management System (AMS) serves as the interface through which users interact with the system, facilitating intuitive navigation and seamless access to various functionalities. In this section, we explore the design and development of the frontend GUI, highlighting key features and functionalities, including buttons for adding, displaying, clearing, deleting, searching, updating, and exiting.

#### **3.1 Swing Overview**

Swing: A built-in Java library that provides a wide range of GUI components like buttons, text fields, menus, and tables. Swing offers a mature and versatile toolkit for creating desktop applications.

#### **3.2 GUI Design Principles**

The GUI design for the AMS adheres to principles of simplicity, clarity, and usability, ensuring an intuitive user experience. Key design elements include:

##### **3.2.1 Layout and Organization**

The interface is organized into logically grouped sections, with clear delineation between different functionalities. Layouts are designed to be visually appealing and easy to navigate.

##### **3.2.2 Visual Feedback**

Interactive elements such as buttons provide immediate visual feedback upon user interaction, enhancing responsiveness and usability. Changes in button states (e.g., hover, click) are reflected visually to indicate user actions.

##### **3.2.3 Error Handling**

Error messages and notifications are displayed prominently to alert users of any issues or invalid inputs. Clear instructions and guidance are provided to assist users in resolving errors and completing tasks successfully.

### **3.3 Key GUI Components and Functionalities**

#### **3.3.1 Add New**

The "Add New" button allows users to add new student records to the system. Clicking this button prompts a form or dialog box where users can input the necessary details for creating a new student profile.

#### **3.3.2 Display**

The "Display" button retrieves and displays existing student records from the database, presenting them in a tabular format within the GUI. Users can view and navigate through the displayed records to access specific student information.

#### **3.3.3 Clear**

The "Clear" button resets or clears the input fields and form elements, allowing users to start afresh or remove any entered data. This functionality aids in maintaining a clean and clutter-free interface.

#### **3.3.4 Delete**

The "Delete" button enables users to delete selected student records from the system. Users may select one or multiple records from the displayed list and initiate the deletion process by clicking this button.

#### **3.3.5 Search**

The "Search" button initiates a search operation, allowing users to search for specific student records based on specified criteria (e.g., name, ID). Users can enter search keywords or criteria, and the system retrieves relevant records matching the search query.

#### **3.3.6 Update**

The "Update" button facilitates the modification or updating of existing student records. Users can select a record from the displayed list, make necessary changes to the record details, and initiate the update process by clicking this button.


### **3.3.7 Exit**

The "Exit" button allows users to exit or close the AMS application. Clicking this button terminates the program and closes the GUI window, providing a convenient way for users to end their session.

### **3.4. User Interaction and Feedback**

The GUI promotes user interaction through intuitive controls, informative feedback, and contextual guidance. Interactive elements such as buttons respond to user actions with visual cues, providing immediate feedback and guidance.

## Insert Function

 — □ ×

### Manage Flight

<-- BACK

Flight Code

Source

Destination

Take Off

No of seats

cdf

vfcvfc

cdf

vcfc

dc

INSERT

UPDATE

SEARCH

DELETE

FlightCode	Source	Destination	TakeOff	NoofSeats
FI - 102				80
126				80
fvkfnk				20
behx	dekj	nen	fnjd	45
125	hd	we	cjv	ahd
cdf	vfcvfc	cdf	vcfc	dc

## Update Function

—

□

×

Manage Passenger

<-- BACK

Passenger Name:

Gender

Nationality

Passport Number

Phone

dedi

dne

neid

000

nnie

INSERT

UPDATE

SEARCH

DELETE

PassengerName	Gender	Nationality	PassportNumber	Phone
dedi	dne	neid	000	nnie

## Search Function

TICKET BOOKING

<-- BACK

Passengerid

Passenger name

Flight Code

Gender

Passportnumber

Amount

Nationality

SAVE

RESET

SEARCH

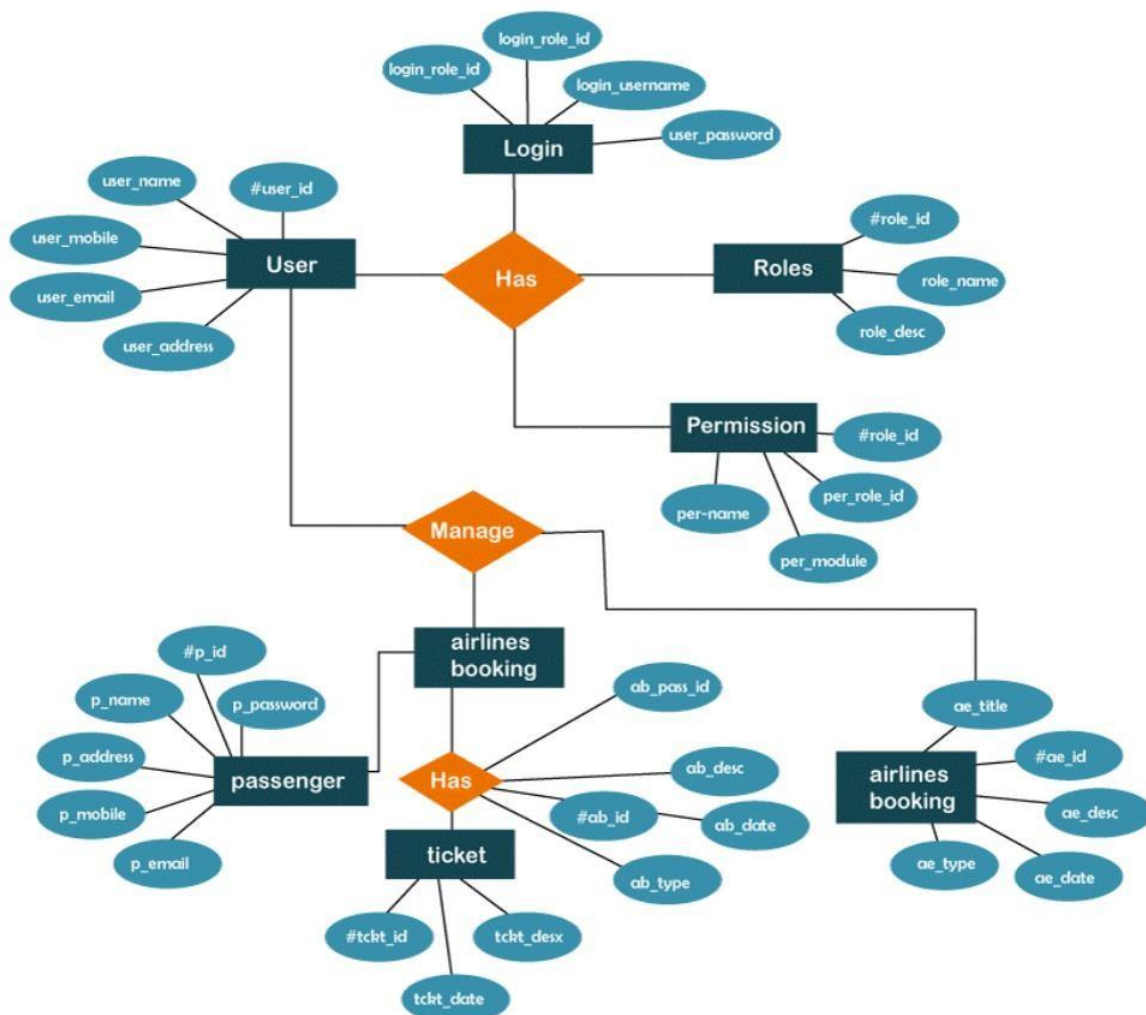
Passenger ID	PassengerName	Flight Code	Gender	Passport Number	Amount	Nationality
123	dnejd	djnednej	djeisdn	desn	wdko	dne
123	dnejd	djnednej	djeisdn	desn	wdko	dne
jdkd	xksa	njsdennd	dekD	DKn	dein	eijl
dmc	nwsj	sjsx	mxs	xjscjd	1234	cmdk
FMkmf	kenkF	EN	DKE	DKM	EFKNFE	DNn

## Chapter 4

### Backend Development

The backend of the Student Management System (AMS) serves as the foundation of the application, handling data storage, retrieval, and processing tasks. In this section, we explore the design and implementation of the backend using JDBC connector and Swing library for database management.

#### ER DIAGRAM FOR AIRLINE MANAGEMENT SYSTEM





## **4.1 Swing Overview**

Swing is a built-in Java library for crafting graphical user interfaces (GUIs). It offers a toolbox of familiar components like buttons, menus, and text fields. While not the most modern option, Swing remains popular for its ease of use and wide range of features, making it a solid choice for building desktop applications.

## **4.2 Database Design**

The database design for the AMS is structured to efficiently store and manage student information, course details, attendance records, and other relevant data. Key considerations in the database design include:

### **4.2.1 Entity-Relationship Model**

The database schema follows an entity-relationship model, defining entities such as Passengers, Flight code, Destination, and Fare, along with their respective attributes and relationships.

### **4.2.2 Normalization**

Normalization techniques are employed to minimize data redundancy and ensure data integrity. The database schema is normalized to reduce the risk of anomalies and inconsistencies.

### **4.2.3 Indexing and Optimization**

Indexing is utilized to improve query performance and accelerate data retrieval operations. Indexes are strategically applied to columns frequently used in search and filter queries, optimizing database performance.

## **4.3 Data Access Layer**

The data access layer serves as an intermediary between the application logic and the database, encapsulating data access operations and ensuring separation of concerns. Key components of the data access layer include:

### **4.3.1 Database Connection**

A database connection is established using Swing's built-in functionality, enabling communication with the underlying database file.

### **4.3.2 Data Manipulation**

CRUD (Create, Read, Update, Delete) operations are implemented to manipulate data stored in the database. Python functions are defined to execute SQL queries and handle data manipulation tasks.

### **4.3.3 Error Handling**

Error handling mechanisms are incorporated to gracefully handle exceptions and errors that may arise during database operations. Exception handling ensures robustness and reliability of the backend functionality.

## **4.4 Security and Authentication**

Security measures are implemented to safeguard the integrity and confidentiality of the data stored in the database. Key security features include:

### **4.4.1 User Authentication**

Authentication mechanisms are implemented to verify the identity of users accessing the system. Secure login procedures prevent unauthorized access to sensitive data and functionalities.

### **4.4.2 Access Control**

Role-based access control (RBAC) mechanisms are employed to enforce access permissions based on user roles and privileges. Access control ensures that users are granted appropriate levels of access to system resources.

## **4.5 Scalability and Performance**

The backend architecture is designed to be scalable and performant, capable of accommodating growing data volumes and user loads. Optimization techniques such as query tuning, connection pooling, and caching are applied to enhance scalability and performance.

## Add Function

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    try{  
        Class.forName("com.mysql.cj.jdbc.Driver");  
  
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ams", "root", "");  
        String sql = "insert into manageflight values (?, ?, ?, ?, ?)";  
        PreparedStatement ptst= con.prepareStatement(sql);  
        ptst.setString(1,fc.getText());  
        ptst.setString(2,s.getText());  
        ptst.setString(3,d.getText());  
        ptst.setString(4,toff.getText());  
        ptst.setString(5,nos.getText());  
        ptst.executeUpdate();  
        JOptionPane.showMessageDialog(this, "Data inserted Successfully!");  
        con.close();  
    }  
    catch(Exception e){  
    }  
}
```

## Delete Function

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    try{  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ams", "root", "");  
        Statement st = con.createStatement();  
  
        String id= fc.getText();  
        String sql = "DELETE FROM `manageflight` WHERE flightcode='"+id+"'";  
        PreparedStatement ptst = con.prepareStatement(sql);  
        ptst.executeUpdate();  
        JOptionPane.showMessageDialog(this, "Data deleted successfully");  
        con.close();  
  
    }  
    catch(Exception e){  
        JOptionPane.showMessageDialog(this, e);  
    }  
}
```

## Search Function

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    try{  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ams", "root", "");  
        Statement st = con.createStatement();  
        String sql = "SELECT * from manageflight";  
        PreparedStatement ptst= con.prepareStatement(sql);  
        ResultSet rs= ptst.executeQuery();  
        DefaultTableModel dt= (DefaultTableModel) jTable1.getModel();  
        dt.setRowCount(0);  
        while(rs.next()){  
            Object o[] = {rs.getString("flightcode"), rs.getString("source"), rs.getString("destination"), rs.getString("takeoff"), rs.getString("noofseat")};  
            dt.addRow(o);  
        }  
    }  
    catch(Exception e){  
        JOptionPane.showMessageDialog(this, e);  
    }  
}
```

## Update Function

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    try{  
        String id=fc.getText();  
        String source=s.getText();  
        String des=d.getText();  
        String tak=toff.getText();  
        String noofseat=nos.getText();  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ams", "root", "");  
  
        String sql= "UPDATE `manageflight` SET `source`='"+source+"', `destination`='"+des+"', `takeoff`='"+tak+"', `noofseat`='"+noofseat+"';"  
        PreparedStatement ptst = con.prepareStatement(sql);  
        ptst.execute();  
  
        JOptionPane.showMessageDialog(this, "Record Updated!");  
    }  
    catch(Exception e){  
        JOptionPane.showMessageDialog(this, e);  
    }  
}
```

## **Chapter 5**

### **Modules and Applications**

#### **5.1 Modules**

##### **5.1.1 Staff**

- **Responsibilities:** Staff module is responsible for managing Flight activities related to boarding, departure, and passenger satisfaction.
- **Functions:**
  - Passenger Management: Staff can create, update, and delete list materials such as id, number, and code.
  - Grading: Staff can input and manage passenger record for departure, arrivals.
  - Communication: Staff can communicate with Passengers through announcements, messages, and feedback on journey.

##### **5.1.2 Passenger**

- **Responsibilities:** Passenger module is responsible for managing their flight journey, accessing booking ticket, and communicating with the crew.
- **Functions:**
  - Flight Booking: Passenger can view available Flights, book flights, and manage their flights.
  - Flight Records: Passenger can access their flight records, including price, discounts, and flight history.
  - Communication: Passenger can interact with crew through messages, discussions, and requesting assistance.

## 5.2 Applications

### 5.2.1 Crew Panel

- **Usage:** The Crew Panel is utilized by instructors and crew members to manage flights, passengers, and communication with passengers.
- **Features:**
  - **Flight Management:** Crew can create and update flight details, upload notes, communicate.
  - **Grading Dashboard:** Crew have access to a grading dashboard to input and manage passenger details, calculate count, and generate performance reports.
  - **Communication Tools:** Crew can communicate with passengers through announcements, messages, and discussion forums.

### 5.2.2 Passenger Panel

- **Usage:** The Passenger Panel is used by passenger to access flight details, view flight, and communicate with crew.
- **Features:**
  - **Flight Enrollment:** Passenger can browse available flights, book desired flight, and manage their flight schedule.
  - **Passenger Dashboard:** Passenger have access to an passenger dashboard to view their records, flight, flight track.
  - **Communication Tools:** Passenger can interact with crew through messages, discussions, and requesting assistance.

## **Conclusion**

In conclusion, the expansion of the Airline Management System with the inclusion of additional modules and applications, such as the Flight crew and Passengers modules along with their corresponding panels, significantly enhances its functionality and utility. With the Crew module, crew gain the ability to manage flight detail, records, and communication with client effectively, streamlining flight activities and improving the overall airline experience. Similarly, the Passenger module empowers passenger to take control of their travel journey by providing access to flight crew, records, and communication channels with crew members. These new features not only cater to the specific needs of crew and passenger but also contribute to the overall efficiency and effectiveness of the airline process. The continued development and refinement of the Airline Management System exemplify a commitment to excellence in flight administration and support the mission of providing a seamless and enriching journey experience for all stakeholders involved.



## References

1. Java Database Connectivity (JDBC) Tutorial: <https://docs.oracle.com/en/java/javase/17/books.html> (Official Oracle documentation on JDBC)
2. Building a Desktop Application with Swing: <https://docs.oracle.com/javase/tutorial/uiswing/> (Official Oracle documentation on Swing)
3. Airline Reservation System Project in Java with MySQL: <https://copyassignment.com/airline-reservation-system-project-in-java/> (Step-by-step guide with code snippets)
4. AAdewunmi/Airline-Management-System: <https://github.com/topics/airline-management-system?l=java> (GitHub repository for a basic Airline Management System with JDBC and MySQL)
5. SchoolSpice/ScotiaAirlines: <https://github.com/topics/airline?l=java> (Another GitHub repository for an Airline Management System using JDBC)
6. Java GUI Frameworks Comparison (Swing vs JavaFX): <https://www.baeldung.com/javafx> (Helps you choose the right GUI framework)
7. NetBeans IDE: <https://netbeans.apache.org/front/main/index.html> (Free and open-source IDE for Java development)