

# **AIRPORT MANAGEMENT SYSTEM**

## **PROJECT REPORT**

By

**EESHANK M (RA2211056010043)**  
**ARMAN CHAUDHURY (RA2211056010060)**  
**ANSH KOUL (RA2211056010053)**  
**P PRAKASHITA RAJKUMARI (RA2211056010067)**

Under the guidance of

**Dr.V.Vijayalakshmi**

*In partial fulfilment for the Course*

of

**21CSC206P – ADVANCED OBJECT ORIENTED AND PROGRAMMING**

in Data Science And Business Systems



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

# **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

## **BONAFIDE CERTIFICATE**

**Certified that this project report for the course 21CSC206P ADVANCED OBJECT ORIENTED AND PROGRAMMING entitled in " AIRPORT MANAGEMENT SYSTEM " is the bonafide work of EESHANK M (RA2211056010043), ANSH KOUL (RA2211056010053) ARMAN CHAUDHURY (RA2211056010060) and P PRAKASHITA RAJKUMARI (RA2211056010067) who carried out the work under my supervision.**

### **SIGNATURE**

Dr.V.Vijayalakshmi

**Assist Professor**

**Data Science and Business Systems**

SRM Institute of Science and Technology

Kattankulathur

## **ABSTRACT**

Airport management is a complex and critical task that involves the coordination of various activities and resources to ensure the smooth operation of an airport. This abstract introduces an Airport Management System developed using Java programming, which aims to streamline and optimize airport operations, enhance passenger experience, and improve overall safety and efficiency.

The Airport Management System is designed to provide a comprehensive set of features to address the diverse needs of airport staff, including administrators, airline operators, security personnel, and air traffic controllers. It leverages Java programming to develop a robust and flexible system that can adapt to evolving airport requirements.

The Airport Management System's use of Java programming ensures cross-platform compatibility, robust security, and scalability. It can be extended with additional modules or integrated with other systems, such as air traffic control and weather forecasting.

## ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. Muthamizhchelvan**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal**, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G, Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to my Course project Faculty **Dr.V.Vijayalakshmi**, Assistant Professor, Data Science And Business Systems, for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. M Lakshmi, Professor, Department of Data Science and Business Systems** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

## TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
<b>1</b>	<b>INTRODUCTION</b>	<b>8</b>
	1.1 Motivation	<b>8</b>
	1.2 Objective	<b>10</b>
	1.3 Problem Statement	<b>11</b>
	1.4 Challenges	<b>12</b>
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>15</b>
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>16</b>
<b>4</b>	<b>ARCHITECTURE &amp; DESIGN</b>	<b>18</b>
<b>5</b>	<b>IMPLEMENTATION</b>	<b>20</b>
<b>6</b>	<b>EXPERIMENT RESULTS &amp; ANALYSIS</b>	<b>37</b>
<b>7</b>	<b>CONCLUSION</b>	<b>55</b>
<b>8</b>	<b>REFERENCES</b>	<b>56</b>

# 1. INTRODUCTION

Airport management plays a crucial role in ensuring the safe and efficient operation of airports, which are complex hubs of transportation, logistics, and passenger services. Java programming has become an indispensable tool in the aviation industry, enabling the development of software systems that streamline various airport management functions. These systems can help manage everything from air traffic control and passenger check-in to baggage handling and security protocols.

In the realm of air traffic control, Java-based applications are instrumental in tracking and guiding aircraft, ensuring their safe and timely arrival and departure. Java's platform independence and strong networking capabilities make it an ideal choice for creating real-time systems that manage flight schedules, monitor weather conditions, and facilitate communication among air traffic controllers, pilots, and ground personnel.

Furthermore, airport management extends beyond the skies to the ground, encompassing passenger services such as ticketing, security, and baggage handling. Java programming allows for the development of user-friendly, web-based applications that help streamline these processes. Passengers can check in for their flights, access flight information, and even track their baggage using Java-powered airport management systems, resulting in a smoother and more convenient travel experience. Overall, Java programming plays a pivotal role in enhancing the operational efficiency and safety of airports while enhancing the passenger experience.

## 1.1 MOTIVATION

Creating an Airport Management System using Java programming offers various benefits and serves as a valuable project for developers. Here are several motivations for developing such a system:

1. **Real-world Application:** An Airport Management System is a practical and real-world application that can be used in the aviation industry. It provides an opportunity to apply Java programming skills to solve real-world problems related to airport operations.
2. **Enhanced Efficiency:** The system can help in streamlining and automating various

airport processes, such as passenger check-in, baggage handling, flight scheduling, and resource allocation. This can lead to improved efficiency in airport operations.

3. Learning Object-Oriented Concepts: Java is an object-oriented programming language, and developing an Airport Management System allows developers to apply and reinforce object-oriented concepts like encapsulation, inheritance, and polymorphism.

4. Database Management: An Airport Management System typically involves the storage and retrieval of a large amount of data. Implementing a database for the system provides an opportunity to work with database management systems (DBMS) and understand how to handle data effectively.

5. User Interface Design: Developing the user interface for the system offers a chance to work on the graphical aspects of programming. You can create a user-friendly interface for various stakeholders, including airport staff, airlines, and passengers.

6. Multi-threading: Airport systems often require handling multiple concurrent tasks, such as flight status updates, passenger check-ins, and baggage handling. Implementing multi-threading in Java can help manage these concurrent tasks efficiently.

7. Security Considerations: Airport systems deal with sensitive information, including passenger details and flight schedules. Implementing security measures in the system allows developers to understand and address security concerns.

8. Integration of External Services: Airport Management Systems often need to integrate with external services, such as weather APIs, airline databases, and security systems. This provides an opportunity to work on integration challenges and understand how different systems can communicate.

9. Error Handling and Logging: Developing a robust error-handling mechanism and logging features is crucial for any system. This project can help developers learn how to handle exceptions, log errors, and troubleshoot issues effectively.

10. Project Management Skills: Developing a comprehensive system involves project planning, task delegation, and coordination among team members. This project can provide insights into project management skills and collaborative development.

11. Documentation: Creating documentation for the Airport Management System is essential for future maintenance and enhancements. It helps developers practice the skill of documenting code and system architecture.

## **1.2 OBJECTIVE**

The objective of developing an Airport Management System using Java programming can be multifaceted, encompassing various goals and functionalities. Below is a comprehensive list of potential objectives for an Airport Management System:

1. **Automated Flight Scheduling:** Develop a system that automates the scheduling of flights, considering factors such as availability of runways, airline preferences, and time constraints.
2. **Passenger Management:** Create a module for managing passenger information, including booking, check-in, and boarding processes.
3. **Baggage Handling:** Implement a system for tracking and managing baggage, ensuring accurate handling and delivery to the correct flights and destinations.
4. **Resource Allocation:** Efficiently allocate airport resources such as gates, check-in counters, and baggage claim areas based on real-time requirements.
5. **Security Management:** Develop security features to monitor and manage airport security, including access control, surveillance, and threat detection.
6. **Flight Information Display System (FIDS):** Implement a display system that provides real-time information about flight schedules, delays, and gate assignments for passengers and staff.
7. **Financial Management:** Integrate a financial module for tracking revenue, expenses, and financial transactions related to airport operations.
8. **Aircraft Maintenance Tracking:** Create a system for monitoring and scheduling maintenance activities for aircraft, ensuring compliance with safety regulations.
9. **Staff Management:** Develop modules for managing airport staff, including roles such as ground crew, security personnel, and administrative staff.
10. **Fuel Management:** Implement a system to monitor and manage fueling operations for aircraft, ensuring an adequate and efficient fuel supply.
11. **Weather Integration:** Integrate weather data to provide real-time information to pilots, airport operations, and passengers, helping to manage and anticipate weather-



related disruptions.

12. Passenger Services: Develop features to enhance passenger experience, such as self-check-in kiosks, mobile boarding passes, and information kiosks.

13. Environmental Impact Monitoring: Implement tools to monitor and assess the environmental impact of airport operations, promoting sustainability and compliance with environmental regulations.

14. Emergency Response System: Develop protocols and systems for emergency response and crisis management, ensuring the safety and well-being of passengers and staff during emergencies.

15. Reporting and Analytics: Implement a reporting system that provides insights into key performance indicators, enabling airport management to make informed decisions and optimize operations

### **1.3 PROBLEM STATEMENT**

The Airport Management System aims to streamline and automate the operations of an airport to enhance efficiency and provide a seamless experience for both staff and passengers. The system will cover various aspects such as passenger handling, flight scheduling, staff management, and resource allocation.

Features:

#### **1. Flight Management:**

- Schedule flights with details such as departure and arrival times, airline information, and gate assignments.
- Allow for the addition, modification, and cancellation of flights.
- Display real-time flight status.

#### **2. Passenger Management:**

- Register passengers for flights.
- Issue boarding passes and manage passenger information.
- Handle passenger check-in and boarding procedures.

#### **3. Staff Management:**

- Maintain information about airport staff, including roles, schedules, and contact

details.

- Assign staff to specific tasks such as security, check-in, and baggage handling.

#### 4. Resource Allocation:

- Manage airport resources such as gates, check-in counters, and baggage claim areas.
- Optimize resource allocation based on flight schedules and passenger traffic.

#### 5. Security Measures:

- Implement security checks for passengers, luggage, and airport premises.
- Monitor and control access to restricted areas.

#### 6. Reporting and Analytics:

- Generate reports on flight performance, passenger statistics, and resource utilization.
- Provide analytics to optimize operations and improve overall efficiency.

#### 7. User Authentication and Authorization:

- Implement secure login mechanisms for airport staff with role-based access control.
- Ensure data privacy and protection.

#### 8. Notification System:

- Send notifications to passengers about flight status, delays, or other relevant information.
- Notify staff about their assigned tasks and schedules.

## 1.4 CHALLENGES

Developing an Airport Management System using Java programming can be a complex task, and there are several challenges that developers might face. Here are some challenges and considerations:

1. **Concurrency and Multithreading:** Airports are busy places with numerous activities happening simultaneously. Managing concurrent access to shared resources, such as runways, gates, and flight schedules, is crucial. Java provides multithreading support, but improper synchronization can lead to data inconsistencies and application instability.

2. **Real-time Data Handling:** Airports deal with real-time data, such as flight schedules, passenger information, and baggage tracking. Ensuring the system can handle and process this data in real-time is a significant challenge. Java offers tools like threads and timers, but designing a responsive and efficient system requires careful consideration.
3. **Security:** Airports handle sensitive information, including passenger details, flight plans, and security protocols. Implementing robust security measures, such as encryption, authentication, and authorization, is essential. Java provides security features, but their proper implementation is critical.
4. **Integration with External Systems:** The Airport Management System needs to integrate with various external systems, such as airline databases, customs systems, and security databases. Ensuring smooth communication and data exchange between these systems can be challenging. Java offers technologies like Java Message Service (JMS) and APIs for integration.
5. **User Interface Design:** Designing an intuitive and user-friendly interface for airport staff, air traffic controllers, and other stakeholders is crucial. Java provides Swing and JavaFX for GUI development, but creating a visually appealing and efficient interface requires careful design and consideration of user experience.
6. **Scalability:** Airports can experience varying levels of traffic and activities. Designing the system to scale efficiently to handle increased loads is essential. Java's platform independence and scalability features can be leveraged, but proper architecture and optimization are crucial.
7. **Fault Tolerance and Recovery:** Airports must operate 24/7, and system failures can have severe consequences. Implementing fault tolerance mechanisms and data recovery strategies is essential. Java provides tools for exception handling, but a comprehensive approach to error recovery is needed.
8. **Compliance with Aviation Standards:** The aviation industry has specific standards and regulations that the Airport Management System must comply with. Ensuring that the system adheres to these standards, such as IATA (International Air Transport Association) regulations, is important for interoperability and compliance.
9. **Testing and Quality Assurance:** Thorough testing is crucial to ensure the reliability

and security of the system. Developing comprehensive test cases and conducting various testing phases, including unit testing, integration testing, and performance testing, is a significant challenge.

10. Documentation: Creating thorough documentation for the Airport Management System, including code documentation, user manuals, and system architecture documentation, is essential for maintenance and future development.

## 2. LITERATURE SURVEY

The two major objectives of this java project are:

1. To create a user interface for the user to look for available flights and then book his tickets accordingly.
2. To implement the tables that contain flight information and attach the functionality of booking tickets on the selected flight & get a boarding pass.

We have implemented a Java project that helps to manage customer information & help to book a flight with Swing (used for GUI). This Airline Reservation System in Java program is a very efficient & easy way to reserve tickets & have all the details of the customer & flight in the boarding pass attached to it.

The implemented Java project distinguishes itself from the referenced Airline Reservation System by virtue of its refined user interface and enhanced functionality. Notably, the project features a meticulously crafted login page and an intelligently designed dashboard, fostering a seamless user experience. The incorporation of dedicated modules for 'Manage Passengers' and 'Manage Flight' empowers users with comprehensive control over customer information and flight details, exemplifying a heightened level of system sophistication. The streamlined integration of ticket booking, and cancellation processes further exemplifies the project's commitment to user convenience. Leveraging the Swing framework for GUI, the project not only prioritizes functionality but also boasts an aesthetically pleasing interface. In sum, the project stands out through its superior user experience, expanded functionality, and refined design, positioning it as a distinguished solution for flight reservation management.

### **3. REQUIREMENTS**

#### **3.1 Requirement Analysis**

##### **1. Development Environment:**

###### **Hardware:**

- Modern computer with sufficient RAM and processing power for development.
- Adequate storage space for the IDE, libraries, and project files.

##### **2. Runtime Environment:**

###### **Hardware:**

- Depending on the nature and complexity of the Java Swing application, modern desktop or laptop configurations are typically sufficient.

##### **3. External Resources:**

###### **- Database Server:**

- Users might need access to the database server, so ensure network connectivity and necessary credentials.

#### **Software Requirements:**

##### **1. Development Environment:**

###### **- DE:**

- NetBeans or any other Java IDE that supports Java Swing development.

###### **- Java Development Kit (JDK):**

- Install the latest version of the JDK for compiling and running Java programs.

###### **- Database :**

- If the application interacts with a database, ensure the necessary database software is installed. Here we Install and use XAMPP for our backend Database structures which can help us view the Tables in our database on the website:

##### **2. Runtime Environment:**

###### **- Java Runtime Environment (JRE):**

- Users need to have the JRE installed on their machines to run Java applications.

###### **- Operating System:**

- Ensure compatibility with the operating systems you intend to support (e.g., Windows, macOS, Linux).

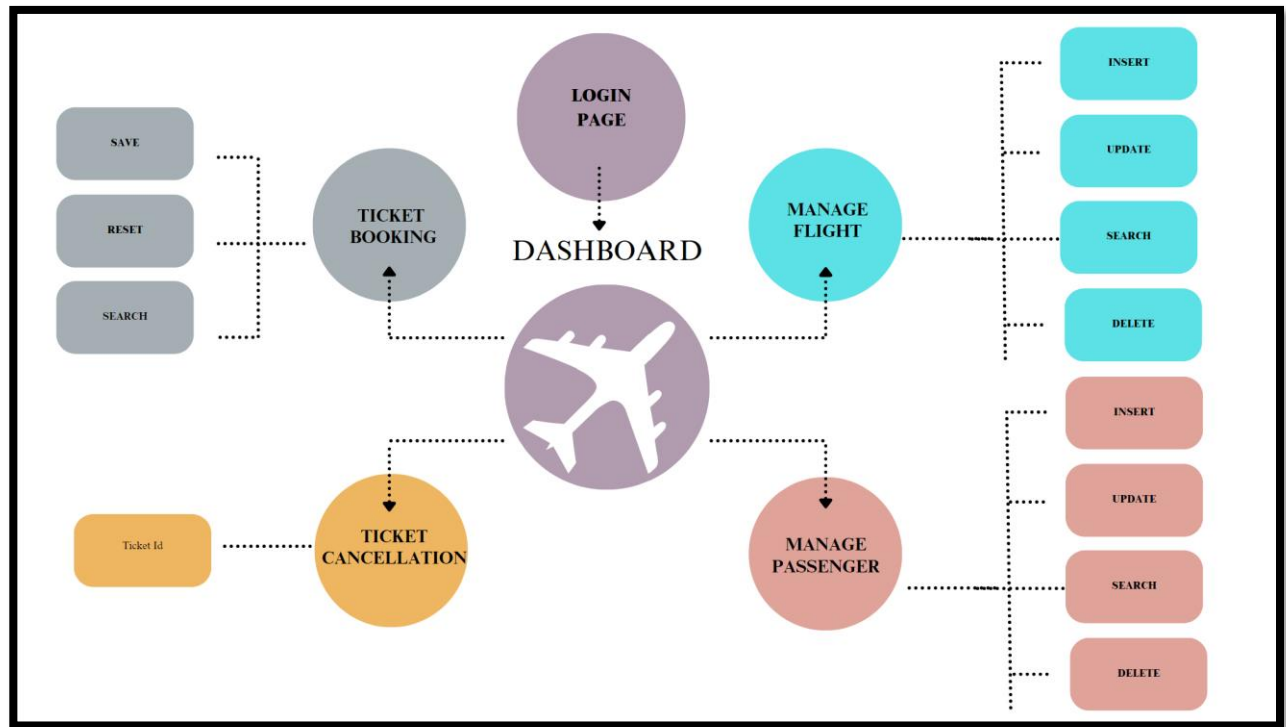
### **3. External Resources:**

#### **- External Libraries:**

- Using JDBC connector by installing and using it in the NetBeans Project Library.
- Using Images in the login page and Intermediate animations when switching from one page to another

## 4. ARCHITECTURE AND DESIGN

### 4.1 Network Architecture



### KEY POINTS

Certainly, Let's dive into more details about the key points mentioned in the context of a ticket booking system for airport management:

#### 1. Flight Code:

- The flight code is a unique identifier assigned to each flight.
- It helps distinguish one flight from another, making it easier for passengers and the system to identify and reference specific flights.
- Typically, flight codes follow a specific format, often including a combination of letters and numbers.

#### 2. Source:

- The source refers to the departure location or the airport from which the flight originates.
- It is essential for passengers to know the source of the flight for planning their travel and logistics.

#### 3. Destination:



- The destination is the arrival location or the airport to which the flight is heading.
- Passengers need to be aware of the destination to ensure they are booking a flight to the correct location.

#### 4. Take-off Time:

- The take-off time indicates when the flight is scheduled to depart from the source airport.
- It is a critical piece of information for passengers to plan their arrival at the airport and manage their travel itinerary.

#### 5. Number of Seats:

- The total number of seats represents the overall capacity of the flight.
- It helps passengers and the system understand the available seating options.
- Airlines often have different classes (e.g., economy, business, first class), each with its own seat capacity.

#### 6. Available Seats:

- The available seats represent the number of seats that are currently unoccupied and can be booked.
- It is a dynamic value that changes as passengers book tickets for the flight.
- Ensuring that the available seats are updated accurately is crucial for preventing overbooking and managing capacity.

These key points collectively form the basic information required for a ticket booking system in airport management. A more comprehensive system would include additional features such as user authentication, payment processing, ticket confirmation, and integration with a database for data persistence. Additionally, considerations for security, scalability, and user experience are vital for the success of an airport ticket booking system.

## 5. IMPLEMENTATION

### CODE:-

#### Loginpage:

// Pseudocode for Login Page

// Initialize the login page

function initializeLoginPage():

    create new instance of loginpage

    set the title of the page to "LOGIN PAGE"

    set the background color to white

    display the login page

// Event handler for the submit button

function onButtonClick():

    // Get the entered username and password

    username = get entered username

    password = get entered password

    // Validate the username and password

    if validateCredentials(username, password):

        // If credentials are valid, show a success message

        showSuccessMessage("Login successful!")

    else:

        // If credentials are invalid, show an error message

```

        showErrorMessage("Invalid username or password")

// Validate the entered credentials

function validateCredentials(username, password):

    // Database connection details

    databaseURL = "jdbc:mysql://your_database_url";

    databaseUser = "your_database_user";

    databasePassword = "your_database_password";


    // Attempt to connect to the database

    try:

        connection = DriverManager.getConnection(databaseURL, databaseUser,
databasePassword)

        statement = connection.createStatement()


        // Execute a query to check if the username and password match a record in the database

        query = "SELECT * FROM users WHERE username = " + username + " AND password =
" + password + ""

        resultSet = statement.executeQuery(query)


        // If a record is found, credentials are valid

        if resultSet.next():

            return true

        else:

            return false

```

```
catch (SQLException e):
```

```
    // Handle database connection errors
```

```
    showMessage("Error connecting to the database")
```

```
// Display a success message
```

```
function showSuccessMessage(message):
```

```
    show a JOptionPane dialog with a success message
```

```
// Display an error message
```

```
function showErrorMessage(message):
```

```
    show a JOptionPane dialog with an error message
```

```
// Main function
```

```
function main():
```

```
    // Set the look and feel for the GUI
```

```
    setLookAndFeel("Nimbus")
```

```
    // Run the GUI application
```

```
    runGUIApplication(initializeLoginPage)
```

### **Dashboard:**

```
// Pseudocode for Dashboard
```

```
// Initialize the dashboard
```

```
function initializeDashboard():
```

```
    create new instance of dashboard
```

set the title of the dashboard to "DASHBOARD"

set the background color to light gray

display the dashboard

// Event handler for the "Manage Flight" button

function onManageFlightButtonClick():

    // Open the Manage Flight page or perform related actions

    openManageFlightPage()

// Event handler for the "Manage Passenger" button

function onManagePassengerButtonClick():

    // Open the Manage Passenger page or perform related actions

    openManagePassengerPage()

// Display a success message

function displayDashboard():

    show the dashboard with buttons and labels

// Main function

function main():

    // Run the GUI application

    runGUIApplication(initializeDashboard)

// Pseudocode for Manage Flight Page

function openManageFlightPage():

```
// Perform actions related to managing flights
```

```
// You can open a new window, update the current window, etc.
```

```
// Pseudocode for Manage Passenger Page
```

```
function openManagePassengerPage():
```

```
    // Perform actions related to managing passengers
```

```
    // You can open a new window, update the current window, etc.
```

### **Manageflight:**

```
// Pseudocode for ManageFlight Class
```

```
// Initialize the ManageFlight page
```

```
function initializeManageFlightPage():
```

```
    create new instance of manageflight
```

```
    set the title of the page to "MANAGE FLIGHT"
```

```
    display the manageflight page
```

```
// Event handler for the "INSERT" button
```

```
function onInsertButtonClick():
```

```
    // Get data from the input fields
```

```
    flightCode = get value from fc field
```

```
    source = get value from s field
```

```
    destination = get value from d field
```

```
    takeOff = get value from tof field
```

```
    numberOfSeats = get value from nos field
```

```

// Insert data into the database

insertDataIntoDatabase(flightCode, source, destination, takeOff, numberOfSeats)


// Update the table to reflect the changes

updateTable()


// Event handler for the "UPDATE" button

function onUpdateButtonClick():

    // Get data from the input fields

    flightCode = get value from fc field

    source = get value from s field

    destination = get value from d field

    takeOff = get value from tof field

    numberOfSeats = get value from nos field


// Update data in the database

updateDataInDatabase(flightCode, source, destination, takeOff, numberOfSeats)


// Update the table to reflect the changes

updateTable()


// Event handler for the "SEARCH" button

function onSearchButtonClick():

    // Get the flight code to search

    flightCode = get value from fc field

```

```

// Search for the flight in the database

searchDataInDatabase(flightCode)


// Event handler for the "DELETE" button

function onDeleteButtonClick():

    // Get the flight code to delete

    flightCode = get value from fc field


// Delete the flight from the database

deleteDataFromDatabase(flightCode)


// Update the table to reflect the changes

updateTable()


// Event handler for the "BACK" label

function onBackLabelClick():

    // Navigate back to the dashboard page

    openDashboardPage()


// Insert data into the database

function insertDataIntoDatabase(flightCode, source, destination, takeOff, numberOfSeats):

    // Use JDBC to connect to the database

    // Execute an SQL INSERT query to add the new flight data

```



```

// Update data in the database

function updateDataInDatabase(flightCode, source, destination, takeOff, numberOfSeats):

    // Use JDBC to connect to the database

    // Execute an SQL UPDATE query to modify the flight data


// Search for the flight in the database

function searchDataInDatabase(flightCode):

    // Use JDBC to connect to the database

    // Execute an SQL SELECT query to retrieve the flight data

    // Display the data in the appropriate fields


// Delete the flight from the database

function deleteDataFromDatabase(flightCode):

    // Use JDBC to connect to the database

    // Execute an SQL DELETE query to remove the flight data


// Update the table with the latest data

function updateTable():

    // Use JDBC to connect to the database

    // Execute an SQL SELECT query to retrieve all flight data

    // Update the jTable1 with the latest data


// Open the Dashboard page

function openDashboardPage():

    create new instance of dashboard

```

display the dashboard page

dispose the current manageflight page

### **Managepassenger:**

// Pseudocode for ManagePassenger Class

// Initialize the ManagePassenger page

function initializeManagePassengerPage():

    create new instance of managepassenger

    set the title of the page to "MANAGE PASSENGER"

    display the managepassenger page

// Event handler for the "INSERT" button

function onInsertButtonClick():

    // Get data from the input fields

    passengerName = get value from pn field

    gender = get value from g field

    nationality = get value from n field

    passportNumber = get value from pass field

    phone = get value from ph field

    flightCode = get value from fc field

    // Insert data into the database

    insertDataIntoDatabase(passengerName, gender, nationality, passportNumber, phone,  
    flightCode)

```
// Update the table to reflect the changes

updateTable()


// Event handler for the "UPDATE" button
function onUpdateButtonClick():

    // Get data from the input fields

    passengerName = get value from pn field

    gender = get value from g field

    nationality = get value from n field

    passportNumber = get value from pass field

    phone = get value from ph field

    flightCode = get value from fc field


    // Update data in the database

    updateDataInDatabase(passengerName, gender, nationality, passportNumber, phone,
flightCode)


    // Update the table to reflect the changes

    updateTable()


// Event handler for the "SEARCH" button
function onSearchButtonClick():

    // Get the passport number to search

    passportNumber = get value from pass field
```

```

// Search for the passenger in the database

searchDataInDatabase(passportNumber)


// Event handler for the "DELETE" button

function onDeleteButtonClick():

    // Get the passport number to delete

    passportNumber = get value from pass field


// Delete the passenger from the database

deleteDataFromDatabase(passportNumber)


// Update the table to reflect the changes

updateTable()


// Event handler for the "BACK" label

function onBackLabelClick():

    // Navigate back to the dashboard page

    openDashboardPage()


// Insert data into the database

function insertDataIntoDatabase(passengerName, gender, nationality, passportNumber, phone,
flightCode):

    // Use JDBC to connect to the database

    // Execute an SQL INSERT query to add the new passenger data

```

```

// Update data in the database

function updateDataInDatabase(passengerName, gender, nationality, passportNumber, phone,
flightCode):

    // Use JDBC to connect to the database

    // Execute an SQL UPDATE query to modify the passenger data


// Search for the passenger in the database

function searchDataInDatabase(passportNumber):

    // Use JDBC to connect to the database

    // Execute an SQL SELECT query to retrieve the passenger data

    // Display the data in the appropriate fields


// Delete the passenger from the database

function deleteDataFromDatabase(passportNumber):

    // Use JDBC to connect to the database

    // Execute an SQL DELETE query to remove the passenger data


// Update the table with the latest data

function updateTable():

    // Use JDBC to connect to the database

    // Execute an SQL SELECT query to retrieve all passenger data

    // Update the jTable1 with the latest data


// Open the Dashboard page

function openDashboardPage():

```

create new instance of dashboard

display the dashboard page

dispose the current managepassenger page

### **Ticketbooking :**

// Pseudocode for TicketBooking Class

// Initialize the TicketBooking page

function initializeTicketBookingPage():

create new instance of ticketbooking

set the title of the page to "TICKET BOOKING"

display the ticketbooking page

// Event handler for the "SAVE" button

function onSaveButtonClick():

// Get data from the input fields

passengerId = get value from pid field

passengerName = get value from pn field

flightCode = get value from fc field

gender = get value from g field

passportNumber = get value from pass field

amount = get value from amt field

nationality = get value from n field

// Save data into the database

saveDataIntoDatabase(passengerId, passengerName, flightCode, gender, passportNumber,

amount, nationality)

// Update the table to reflect the changes

updateTable()

// Event handler for the "RESET" button

function onResetButtonClick():

// Clear all input fields

clearInputFields()

// Event handler for the "SEARCH" button

function onSearchButtonClick():

// Get the passenger ID to search

passengerId = get value from pid field

// Search for the passenger in the database

searchDataInDatabase(passengerId)

// Event handler for the "BACK" label

function onBackLabelClick():

// Navigate back to the dashboard page

openDashboardPage()

// Save data into the database

function saveDataIntoDatabase(passengerId, passengerName, flightCode, gender,

passportNumber, amount, nationality):

// Use JDBC to connect to the database

// Execute an SQL INSERT query to add the new ticket booking data

// Search for the passenger in the database

function searchDataInDatabase(passengerId):

// Use JDBC to connect to the database

// Execute an SQL SELECT query to retrieve the ticket booking data

// Display the data in the appropriate fields

// Update the table with the latest data

function updateTable():

// Use JDBC to connect to the database

// Execute an SQL SELECT query to retrieve all ticket booking data

// Update the jTable1 with the latest data

// Clear all input fields

function clearInputFields():

// Set the values of all input fields to empty strings

set value of pid field to ""

set value of pn field to ""

set value of fc field to ""

set value of g field to ""

set value of pass field to ""

set value of amt field to ""



set value of n field to ""

// Open the Dashboard page

function openDashboardPage():

create new instance of dashboard

display the dashboard page

dispose the current ticketbooking page

**Intermediatepage(redirecting page):**

// Pseudocode for IntermediatePage1 Class

// Initialize the IntermediatePage1

function initializeIntermediatePage1():

create new instance of IntermediatePage1

set the title of the page to "REDIRECTING"

createTimerAndRedirect()

display the IntermediatePage1

// Create a timer and redirect to the dashboard

function createTimerAndRedirect():

// Create a timer to delay the redirection to the dashboard form

create new Timer with delay of 3000 milliseconds (3 seconds)

add ActionListener to the timer:

// This will be executed when the timer elapses

close the current IntermediatePage1 frame

open the dashboard frame

// Start the timer

set the timer to run only once

start the timer

// Main method

function main():

set the look and feel for the UI

// Create and display the IntermediatePage1 form

create new instance of IntermediatePage1

set it to be visible

// Variables declaration

private JLabel jLabel1, jLabel2;

private JPanel jPanel1;

## **6. EXPERIMENTAL RESULTS AND ANALYSIS:**

A Flight Management System with features like flight booking and cancellation can offer several benefits to the aviation industry. Here are some potential advantages:

### **1. Efficient Operations:**

- Automation of flight booking and cancellation processes can significantly improve efficiency. This reduces the workload on staff and minimizes the chances of errors associated with manual data entry.

### **2. Cost Savings:**

- Automation often leads to cost savings in terms of reduced manual labor and increased accuracy. With a well-designed system, resources can be allocated more effectively, resulting in overall cost reduction.

### **3. Improved Customer Experience:**

- Streamlined booking and cancellation processes contribute to a better customer experience. Passengers can easily book or cancel flights through a user-friendly interface, which can lead to increased customer satisfaction and loyalty.

### **4. Real-time Information:**

- Having a centralized database allows for real-time updates on flight availability, bookings, and cancellations. This ensures that both staff and customers have access to the most up-to-date information.

### **5. Data Analysis and Reporting:**

- The system can generate reports and analytics based on the data collected. This information can be invaluable for airlines to make informed decisions, such as optimizing routes, adjusting pricing strategies, and identifying trends in passenger behavior.

### **6. Enhanced Security and Compliance:**

- A robust Flight Management System can contribute to enhanced security measures by ensuring that passenger data is handled securely. It can also assist in compliance with industry regulations and standards.

## **7. Resource Optimization:**

- Efficient management of flights, bookings, and cancellations allows for better resource optimization. Airlines can allocate resources such as aircraft, crew, and ground staff more effectively, leading to improved operational efficiency.

## **8. Integration with Other Systems:**

- The system can be integrated with other relevant systems, such as payment gateways, airport systems, and travel agencies. This seamless integration enhances overall connectivity and workflow efficiency.

## **9. Adaptability and Scalability:**

- A well-designed system should be adaptable to changes in the industry and scalable to accommodate growth. This ensures that the system remains relevant and effective over time.

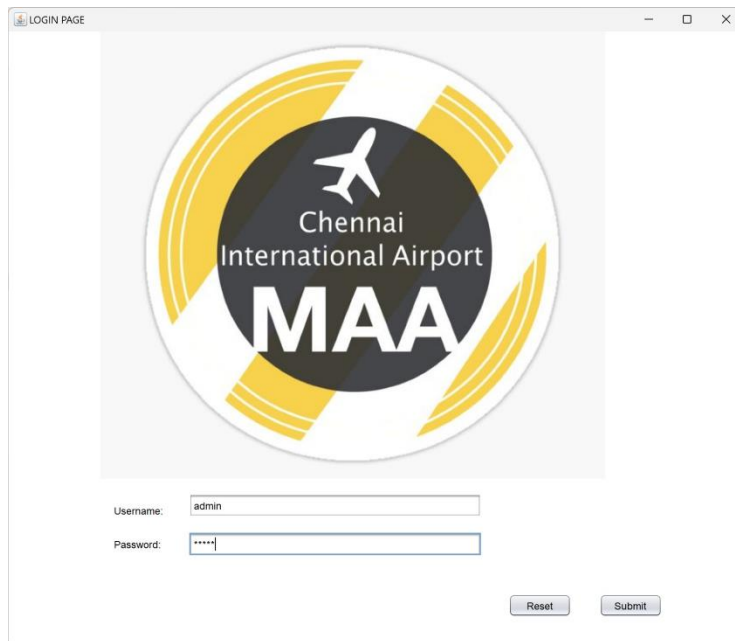
## **10. Competitive Advantage:**

- Implementing a sophisticated Flight Management System can give an airline a competitive advantage. Offering a seamless and efficient booking and cancellation experience can attract more customers and help the airline stand out in the market.

In summary, a comprehensive Flight Management System with booking and cancellation features can contribute to increased efficiency, cost savings, improved customer satisfaction, and a competitive edge in the aviation industry.

## LOGIN PAGE:

### BEFORE RESET :-



LOGIN PAGE

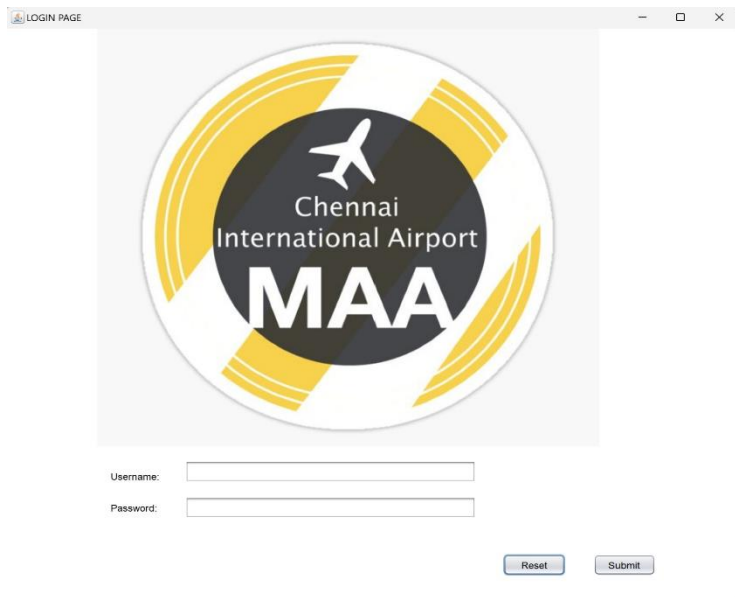
Chennai International Airport  
MAA

Username:

Password:

Reset Submit

### AFTER RESET:-



LOGIN PAGE

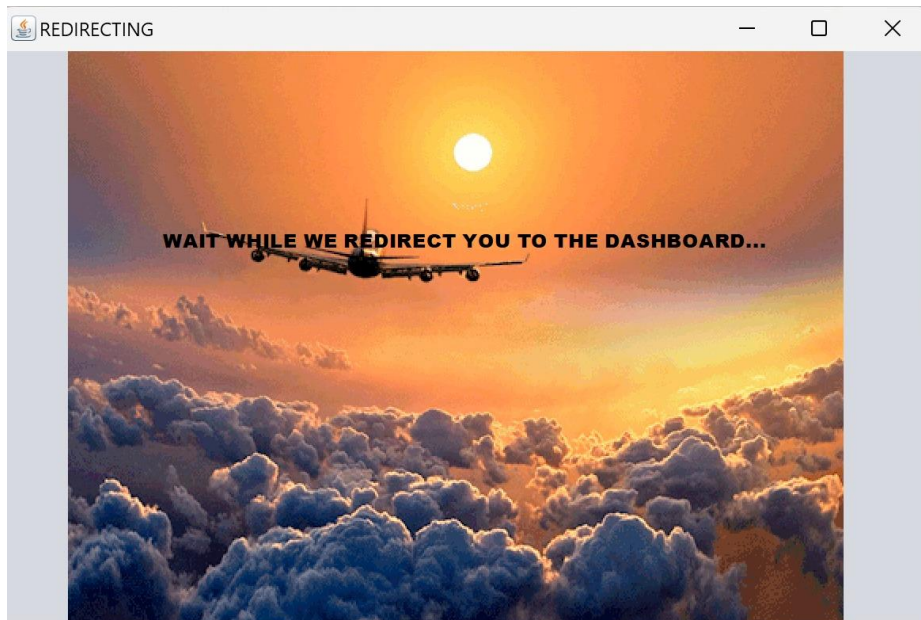
Chennai International Airport  
MAA

Username:

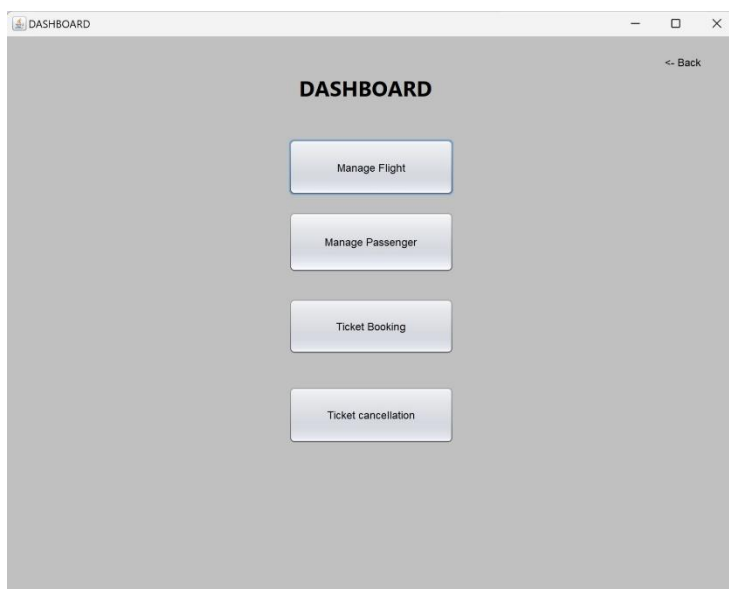
Password:

Reset Submit

## ON LOGIN: -



**WE CAN SELECT ON ANY OF THE BUTTONS TO MOVE TO THE PARTICULAR PAGE , ALSO CLICKING ON BACK TAKES US BACK TO THE LOGIN PAGE**



**MANAGE FLIGHT FORM PROVIDES THE ADMIN WITH FOUR OPTIONS, THAT IS EITHER ADD PASSENGERS USING INSERT, UPDATE PASSENGERS DETAILS, SEARCH FOR A PARTICULAR PASSENGER/SET OF PASSENGERS, AND DELETE PASSENGERS**

The screenshot shows a web application window titled "MANAGE FLIGHT". Inside, there's a form titled "Manage Flight" with a "<- BACK" link. The form has five input fields: "Flight Code", "Source", "Destination", "Take off", and "No of Seats". Below these fields are four buttons: "INSERT", "UPDATE", "SEARCH", and "DELETE". Below the buttons is a table with the following headers: "FlightCode", "Source", "Destination", "Take off", and "NoofSeats". The table is currently empty.

**AFTER ENTERING FLIGHT DETAILS AND CLICKING ON INSERT**

The screenshot shows the same "MANAGE FLIGHT" window. The input fields are now filled with: "Flight Code" (C1-702), "Source" (CHENNAI), "Destination" (SRINAGAR), "Take off" (30/11/2023), and "No of Seats" (225). The "INSERT" button has been clicked, and a message dialog box is displayed in the center. The dialog box has a title bar "Message" and a close button (X). It contains an information icon (i) and the text "Data inserted Successfully!". There is an "OK" button at the bottom of the dialog box. The table below the form now contains one row with the data: "C1-702", "CHENNAI", "SRINAGAR", "30/11/2023", and "225".

**USING UPDATE WE CAN CHANGE THE FLIGHT DETAILS LIKE HERE WE CHANGE THE DATE FROM 30TH TO 29TH**

MANAGE FLIGHT

Manage Flight

<- BACK

Flight Code

Source

Destination

Take of

No of Seats

C1-702

CHENNAI

SRINAGAR

29/11/2023

225

INSERT

UPDATE

SEARCH

DELETE

FlightCode	Source	Destination	Take off	NoofSeats
C1-702	CHENNAI			225

Message

i

Record Updated!

OK



**FOR SEARCHING WE JUST NEED TO ENTER THE SOURCE AND DESTINATION FOR THE FLIGHTS WE ARE LOOKING FOR AND CLICK ON THE SEARCH BUTTON**

The screenshot shows a window titled "MANAGE FLIGHT" with a sub-header "Manage Flight" and a "<- BACK" button. Below the header are five input fields: "Flight Code", "Source", "Destination", "Take of", and "No of Seats". The "Source" field contains "CHENNAI" and the "Destination" field contains "MUMBAI". Below these fields are four buttons: "INSERT", "UPDATE", "SEARCH", and "DELETE". The "SEARCH" button is highlighted with a blue border. Below the buttons is a table with the following data:

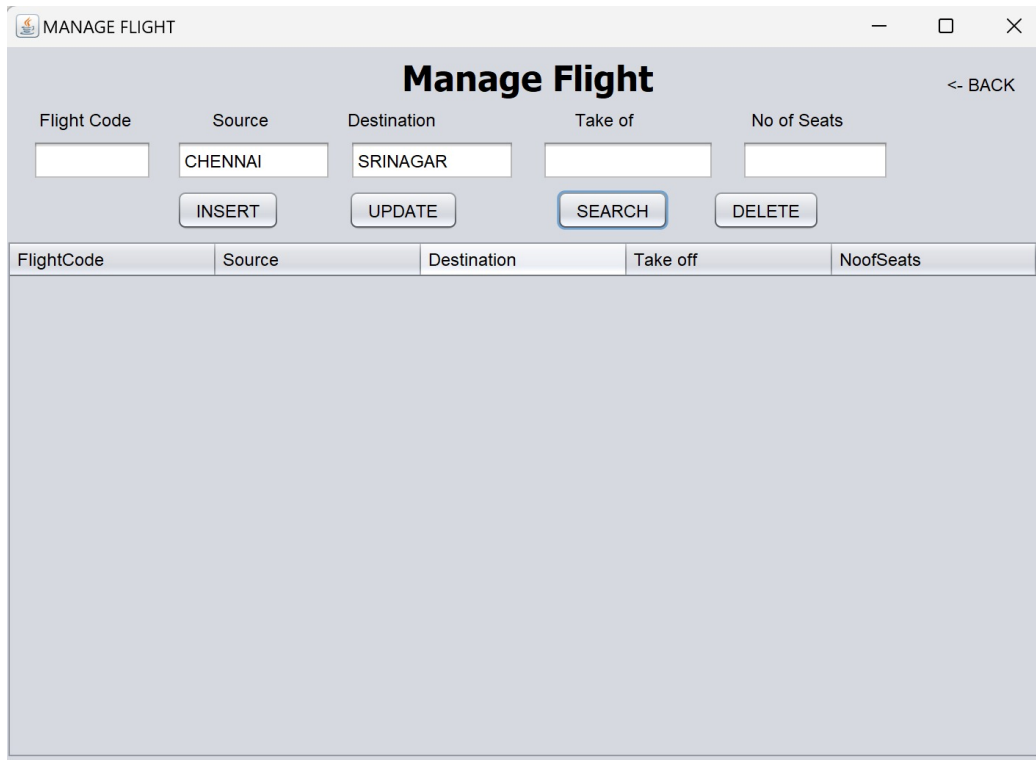
FlightCode	Source	Destination	Take off	NoofSeats
C1-301	CHENNAI	MUMBAI	20/10/2023	200
C1-502	CHENNAI	MUMBAI	30/10/2023	180

**FOR DELETION AGAIN WE USE THE FLIGHTCODE CORRESPONDING TO THE PARTICULAR FLIGHT AND CLICK ON DELETE**

The screenshot shows the same "MANAGE FLIGHT" window, but now the "Flight Code" field contains "C1-702". The "SEARCH" button is still highlighted. A modal dialog box is open in the center of the screen with the title "Message" and a close button (X). The dialog contains an information icon (i) and the text "Data deleted successfully". Below the text is an "OK" button. The table in the background is partially obscured by the dialog box, but the visible rows are:

FlightCode	Source	Destination	Take off	NoofSeats
C1-301	CHENNAI			200
C1-502	CHENNAI			180

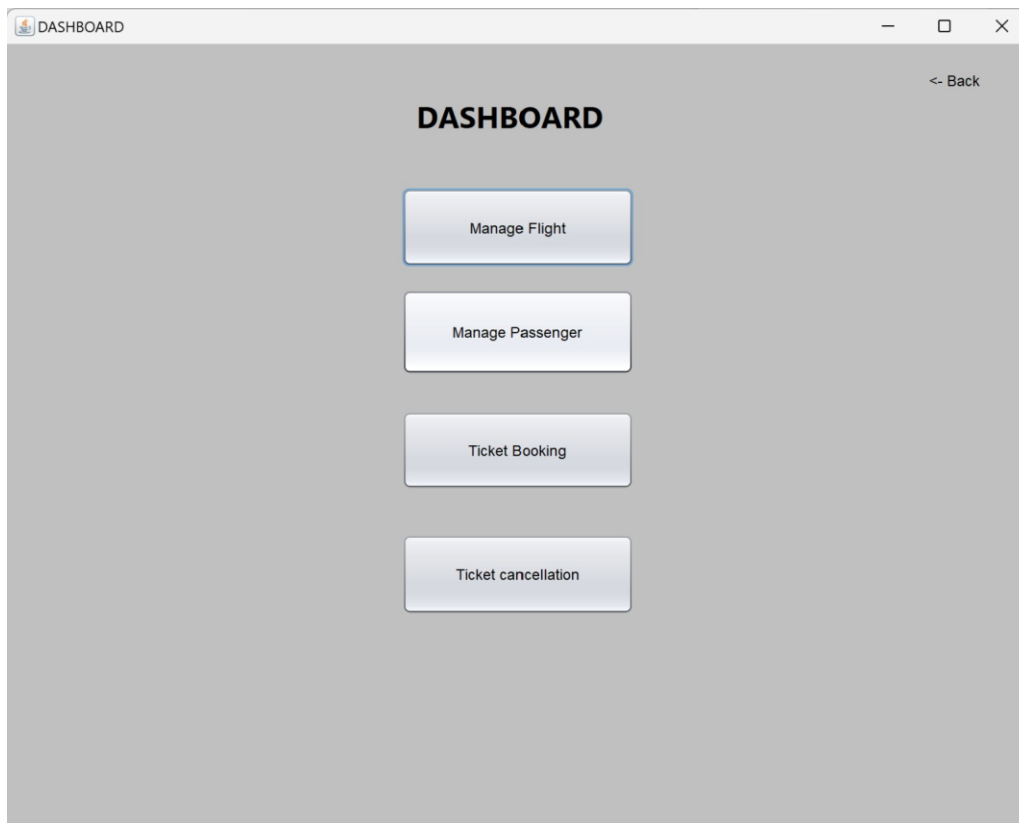
**THEN AGAIN ON SEARCHING WE FIND THAT THE FLIGHT DETAILS HAVE BEEN DELETED**



The screenshot shows a web application window titled "MANAGE FLIGHT". The main heading is "Manage Flight". There is a "<- BACK" link in the top right corner. Below the heading, there are five input fields: "Flight Code", "Source" (containing "CHENNAI"), "Destination" (containing "SRINAGAR"), "Take of", and "No of Seats". Below these fields are four buttons: "INSERT", "UPDATE", "SEARCH" (which is highlighted with a blue border), and "DELETE". Below the buttons is a table with the following headers: "FlightCode", "Source", "Destination", "Take off", and "NoofSeats". The table body is empty.

FlightCode	Source	Destination	Take off	NoofSeats
------------	--------	-------------	----------	-----------

**CLICKING ON THE BACK BUTTON TAKES US BACK TO THE DASHBOARD**



**FROM HERE AGAIN WE CLICK ON MANAGE PASSENGER TO MANAGE PASSENGER DETAILS**

### **THE MANAGEPASSENGER FORM**

A screenshot of a web application window titled "MANAGE PASSENGER". The window has a light gray background and a dark gray header bar. In the top right corner of the header bar, there are window control icons (minimize, maximize, close) and a "<BACK" link. The main content area is titled "Manage Passenger" in bold black text. Below the title, there are six input fields arranged horizontally, each with a label above it: "Passenger Name:", "Gender", "Nationality", "Passport Number", "Phone", and "FlightCode". The "Passenger Name" field is currently active, showing a cursor. Below the input fields, there are four buttons: "INSERT", "UPDATE", "SEARCH", and "DELETE". Below the buttons, there is a table with six columns: "PassengerName", "Gender", "Nationality", "Passportnumber", "Phone", and "FlightCode". The table has a light gray header and a large empty body area.

**ON ENTERING THE PASSENGER DETAILS AND CLICKING ON INSERT**

MANAGE PASSENGER

## Manage Passenger

<BACK

Passenger Name: 
 Gender: 
 Nationality: 
 Passport Number: 
 Phone: 
 FlightCode:

PassengerName	Gender	Nationality	Passport Number	Phone	FlightCode

Message

Data inserted Successfully!

**WHEN WE CLICK ON SEARCH WE GET THE LIST OF ALL THE PASSENGERS THAT HAVE CURRENT BOOKING (STORED ON OUR DATABASE)**

MANAGE PASSENGER

## Manage Passenger

<BACK

Passenger Name: 
 Gender: 
 Nationality: 
 Passport Number: 
 Phone: 
 FlightCode:

PassengerName	Gender	Nationality	Passportnumber	Phone	FlightCode
YASHASVI	MALE	INDIAN	IND-123	7279999969	C1-101
LOPAMUDRA	FEMALE	INDIAN	IND-321	1234567898	C1-101
EESHANK	MALE	INDIAN	IND-654	7278888869	C1-102
ARMAN	MALE	INDIAN	IND-911	9841186778	C1-102
MEDHA	FEMALE	INDIAN	IND-119	9941430341	C1-201
ANSH	MALE	INDIAN	IND-789	7010319797	C1-201
ADARSH	MALE	INDIAN	IND-987	7550161230	C1-202
VANSH	MALE	INDIAN	IND-111	8939176750	C1-202
KEITH	MALE	INDIAN	IND-456	9841411113	C1-301
ARPITH	MALE	INDIAN	IND-222	8939176750	C1-301
ANUJ	MALE	INDIAN	IND-333	9087654321	C1-302
CHARAN	MALE	INDIAN	IND-444	9878675432	C1-302
AASHISH	MALE	INDIAN	IND-555	9182706354	C1-401
NANDHA	MALE	INDIAN	IND-696	9043245531	C1-401
AARAV	MALE	INDIAN	IND-123	9876543210	C1-501
VISHAKA	FEMALE	INDIAN	IND-309	7890987654	C1-701

**WE MAKE THE UPDATION REQUIRED(HERE THE PASSPORT NUMBER CORRESPONDING TO THE NAME VISHAKA)**

**MANAGE PASSENGER**

## Manage Passenger

<BACK

Passenger Name: 
 Gender: 
 Nationality: 
 Passport Number: 
 Phone: 
 FlightCode:

Message

Record Updated!

PassengerName	Gender	Nationality	Passport Number	Phone	FlightCode
ISHA	FEMALE	INDIAN	IND-456	8765432109	C1-501
VIHAAN	MALE	INDIAN	IND-789	7654321098	C1-501
ANAYA	FEMALE	INDIAN	IND-234	6543210987	C1-502
ARJUN	MALE	INDIAN	IND-567	5432109876	C1-502
SARA	FEMALE	INDIAN	IND-890	4321098765	C1-502
AYAN	MALE	INDIAN	IND-345	3210987654	C1-601
ZARA	FEMALE	INDIAN	IND-678	2109876543	C1-601
ARNAV	MALE	INDIAN	IND-901	1098765432	C1-601
AISHA	FEMALE	INDIAN	IND-456	9876543210	C1-602
VIVAAN	MALE	INDIAN	IND-789	8765432109	C1-602
SOFIA	FEMALE	INDIAN	IND-012	7654321098	C1-602
REYAN	MALE	INDIAN	IND-567	6543210987	C1-701
ANVI	FEMALE	INDIAN	IND-890	5432109876	C1-701
KABIR	MALE	INDIAN	IND-123	4321098765	C1-701
VISHAKA	FEMALE	INDIAN	IND-309	7890987654	C1-701

**THENY CLICKING ON SEARCH WE SEE THE CHANGES REFLECTING**

**MANAGE PASSENGER**

## Manage Passenger

<BACK

Passenger Name: 
 Gender: 
 Nationality: 
 Passport Number: 
 Phone: 
 FlightCode:

PassengerName	Gender	Nationality	Passportnumber	Phone	FlightCode
ISHA	FEMALE	INDIAN	IND-456	8765432109	C1-501
VIHAAN	MALE	INDIAN	IND-789	7654321098	C1-501
ANAYA	FEMALE	INDIAN	IND-234	6543210987	C1-502
ARJUN	MALE	INDIAN	IND-567	5432109876	C1-502
SARA	FEMALE	INDIAN	IND-890	4321098765	C1-502
AYAN	MALE	INDIAN	IND-345	3210987654	C1-601
ZARA	FEMALE	INDIAN	IND-678	2109876543	C1-601
ARNAV	MALE	INDIAN	IND-901	1098765432	C1-601
AISHA	FEMALE	INDIAN	IND-456	9876543210	C1-602
VIVAAN	MALE	INDIAN	IND-789	8765432109	C1-602
SOFIA	FEMALE	INDIAN	IND-012	7654321098	C1-602
REYAN	MALE	INDIAN	IND-567	6543210987	C1-701
ANVI	FEMALE	INDIAN	IND-890	5432109876	C1-701
KABIR	MALE	INDIAN	IND-123	4321098765	C1-701
VISHAKA	FEMALE	INDIAN	IND-311	7890987654	C1-701

**FOR DELETING AGAIN WE USE THE PASSENGERNAME ALONE**

MANAGE PASSENGER

Manage Passenger

<BACK

Passenger Name:
 Gender
 Nationality
 Passport Number
 Phone
 FlightCode

VISHAKA

INSERT
 UPDATE
 SEARCH
 DELETE

Message
 X

Data deleted successfully
 OK

PassengerName	Gender				FlightCode
ISHA	FEMALE				C1-501
VIHAAN	MALE				C1-501
ANAYA	FEMALE				C1-502
ARJUN	MALE				C1-502
SARA	FEMALE				C1-502
AYAN	MALE				C1-601
ZARA	FEMALE	INDIAN	IND-678	2109876543	C1-601
ARNAV	MALE	INDIAN	IND-901	1098765432	C1-601
AISHA	FEMALE	INDIAN	IND-456	9876543210	C1-602
VIVAAN	MALE	INDIAN	IND-789	8765432109	C1-602
SOFIA	FEMALE	INDIAN	IND-012	7654321098	C1-602
REYAN	MALE	INDIAN	IND-567	6543210987	C1-701
ANVI	FEMALE	INDIAN	IND-890	5432109876	C1-701
KABIR	MALE	INDIAN	IND-123	4321098765	C1-701
VISHAKA	FEMALE	INDIAN	IND-311	7890987654	C1-701

HERE NOW WE CAN CLICK ON TICKETBOOKING

DASHBOARD

<- Back

DASHBOARD

Manage Flight
 Manage Passenger
 Ticket Booking
 Ticket cancellation

## THE FIRST THING WE CAN DO ON TICKETBOOKING IS ENTER THE PASSENGER BOOKING DETAILS ALONG WITH THE PRICE

The screenshot shows a web application window titled "TICKET BOOKING". At the top right is a "<BACK" link. Below the title is a form with seven input fields: "Passengerid", "Passenger name", "Flight Code", "Gender", "Passportnumber", "Amount", and "nationality". Below these fields are three buttons: "SAVE", "RESET", and "SEARCH". At the bottom of the form is a table with seven columns: "Passenger ID", "Passenger...", "Flight Code", "Gender", "Passport N...", "Amout", and "Nationality". The table is currently empty.

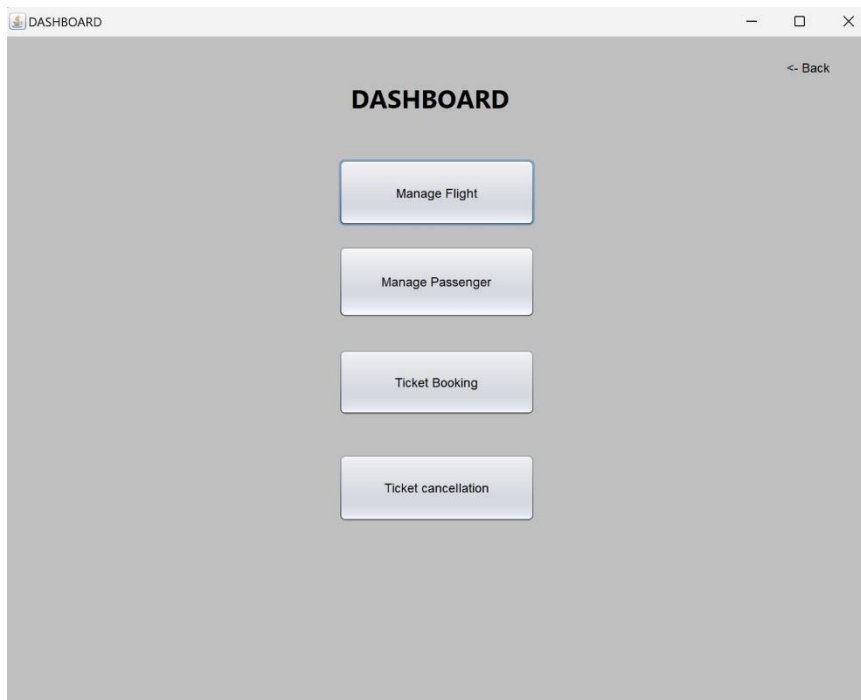
## AGAIN SEARCHING RETURNS THE ENTIRE LOT OF PASSENGERS WE HAVE BOOKED RECENTLY

The screenshot shows the same "TICKET BOOKING" application window, but now the "SEARCH" button is highlighted with a blue border. The table below the form is populated with 16 rows of passenger data. The columns are: "Passenger...", "Passenger...", "Flight Code", "Gender", "Passport ...", "Amout", and "Nationality".

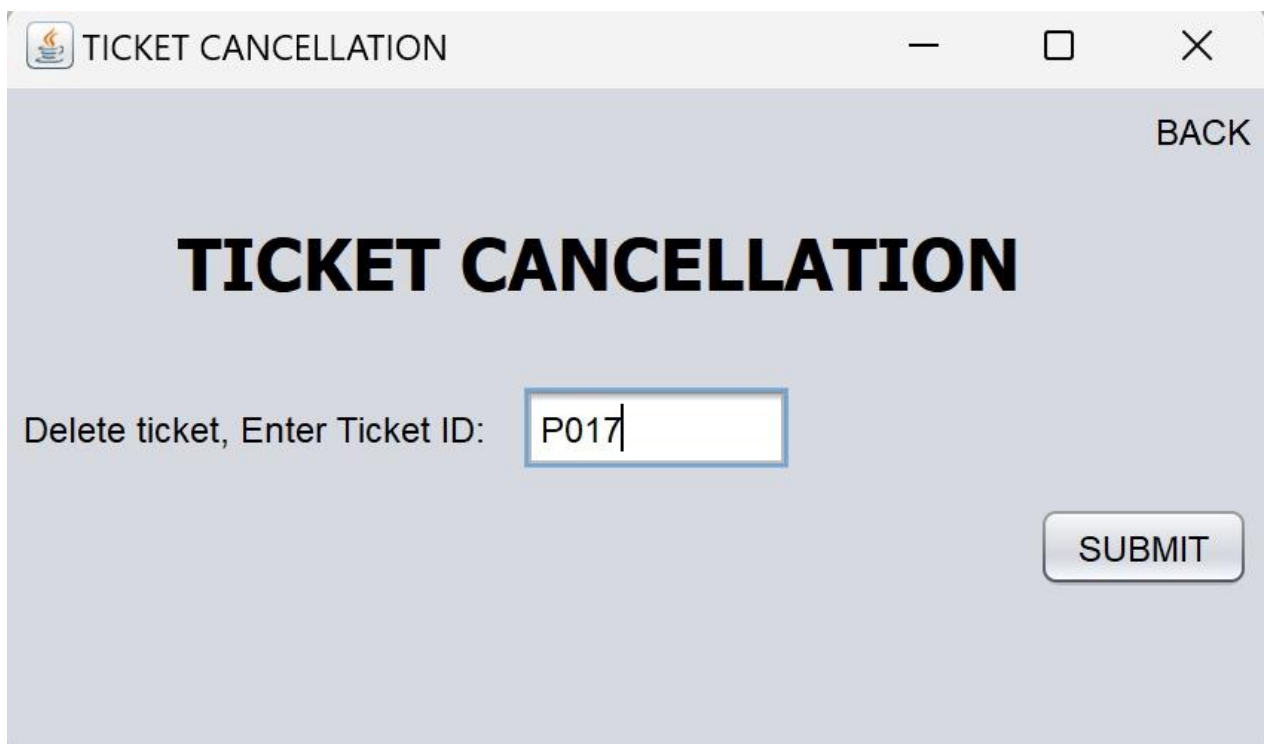
Passenger...	Passenger...	Flight Code	Gender	Passport ...	Amout	Nationality
P001	YASHASVI	C1-101	MALE	IND-123	3500	INDIAN
P002	LOPAMUD...	C1-101	FEMALE	IND-321	3500	INDIAN
P003	EESHANK	C1-102	MALE	IND-654	3800	INDIAN
P004	ARMAN	C1-102	MALE	IND-911	3800	INDIAN
P005	MEDHA	C1-201	FEMALE	IND-119	3800	INDIAN
P006	ANSH	C1-201	MALE	IND-789	3800	INDIAN
P007	ADARSH	C1-202	MALE	IND-987	4000	INDIAN
P008	VANSH	C1-202	MALE	IND-111	4000	INDIAN
P009	KEITH	C1-301	MALE	IND-456	3600	INDIAN
P010	ARPIYH	C1-301	MALE	IND-222	3600	INDIAN
P011	ANUJ	C1-302	MALE	IND-333	3200	INDIAN
P012	CHARAN	C1-302	MALE	IND-444	3200	INDIAN
P013	AASHISH	C1-401	MALE	IND-555	4200	INDIAN
P014	NANDHA	C1-401	MALE	IND-696	4200	INDIAN
P015	YASH	C1-402	MALE	IND-777	4500	INDIAN
P016	RAHIMA	C1-402	FEMALE	IND-888	4500	INDIAN



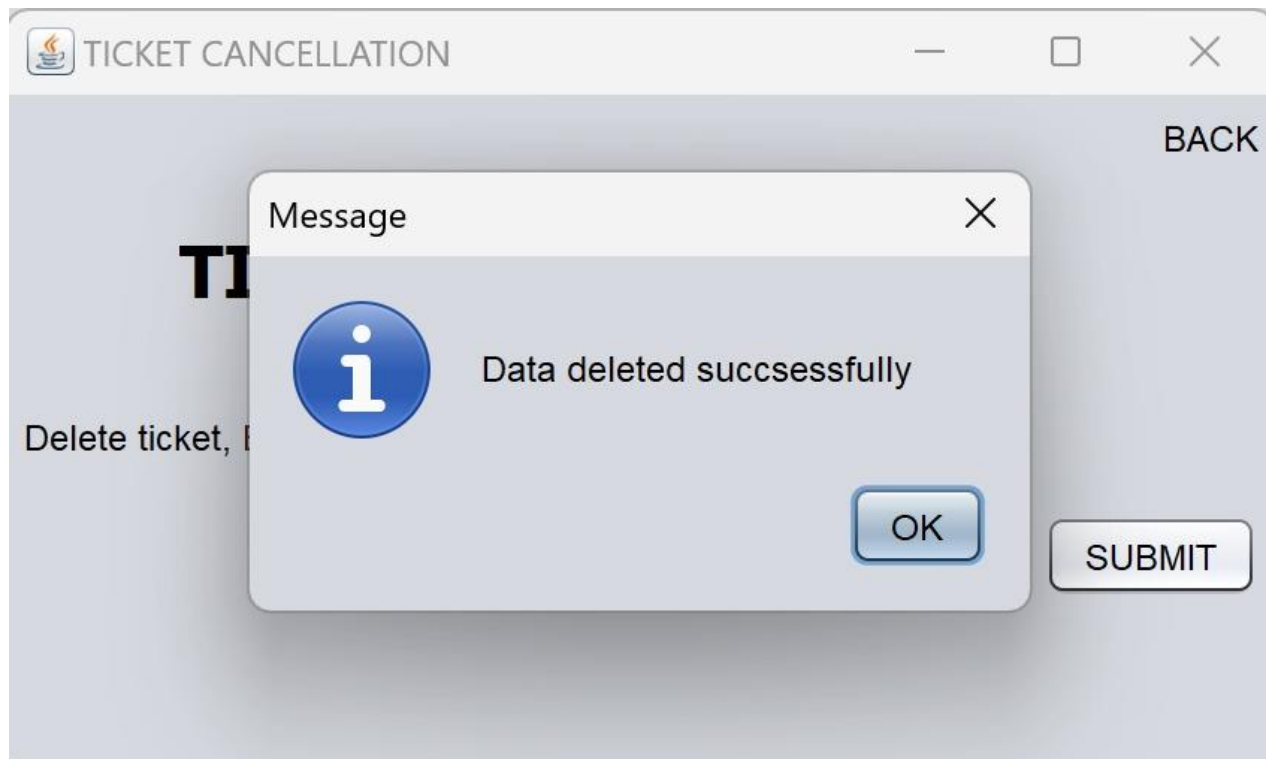
## BACK BRINGS US BACK TO THE DASHBOARD



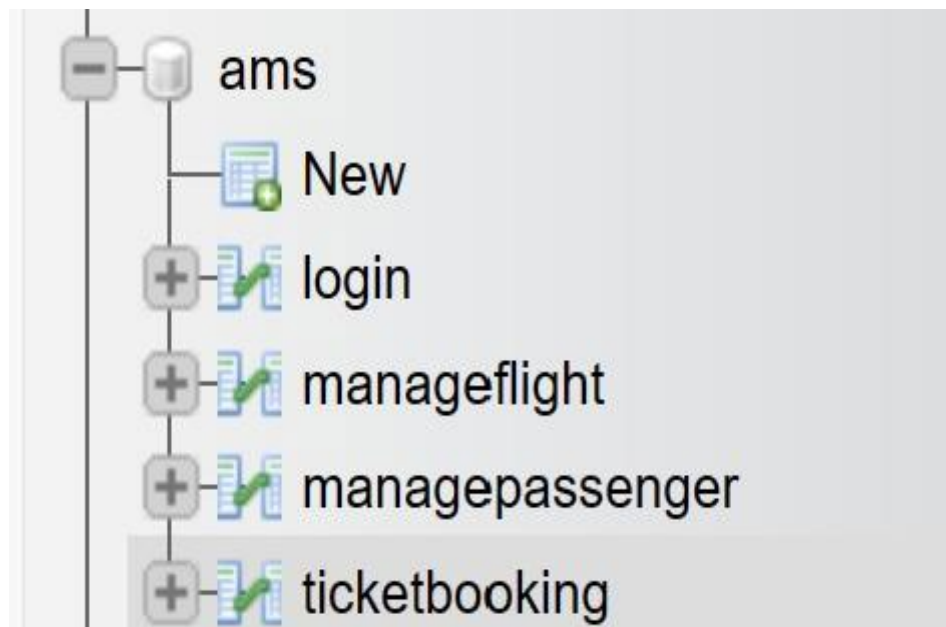
## TICKET CANCELLATION:-







## OUR DATABASE 'ams' AND THE DIFFERENT TABLES WE HAVE CREATED



## LOGIN TABLE

**username**

**password**

admin

admin

## MANAGE FLIGHTS TABLE

flightcode	source	destination	takeoff	noofseat
C1-101	CHENNAI	BENGALURU	19/10/2023	150
C1-102	BENGALURU	CHENNAI	24/10/2023	150
C1-201	CHENNAI	DELHI	18/10/2023	180
C1-202	DELHI	CHENNAI	26/10/2023	180
C1-301	CHENNAI	MUMBAI	20/10/2023	200
C1-302	MUMBAI	CHENNAI	29/10/2023	200
C1-401	chennai	kolkata	23/10/2023	65
C1-402	GUWAHATI	CHENNAI	23/10/23	160
C1-501	CHENNAI	BHOPAL	29/10/2023	200
C1-502	CHENNAI	MUMBAI	30/10/2023	180
C1-601	DELHI	KOLKATA	31/10/2023	220
C1-602	MUMBAI	DELHI	01/11/2023	190
C1-701	BANGALORE	HYDERABAD	02/11/2023	210

## THE MANAGEPASSENGER TABLE

passengername	gender	nationality	passportnumber	phone	flightcode
YASHASVI	MALE	INDIAN	IND-123	7279999969	C1-101
LOPAMUDRA	FEMALE	INDIAN	IND-321	1234567898	C1-101
EESHANK	MALE	INDIAN	IND-654	7278888869	C1-102
ARMAN	MALE	INDIAN	IND-911	9841186778	C1-102
MEDHA	FEMALE	INDIAN	IND-119	9941430341	C1-201
ANSH	MALE	INDIAN	IND-789	7010319797	C1-201
ADARSH	MALE	INDIAN	IND-987	7550161230	C1-202
VANSH	MALE	INDIAN	IND-111	8939176750	C1-202
KEITH	MALE	INDIAN	IND-456	9841411113	C1-301
ARPITH	MALE	INDIAN	IND-222	8939176750	C1-301
ANUJ	MALE	INDIAN	IND-333	9087654321	C1-302
CHARAN	MALE	INDIAN	IND-444	9878675432	C1-302
AASHISH	MALE	INDIAN	IND-555	9182706354	C1-401
NANDHA	MALE	INDIAN	IND-696	9043245531	C1-401
AARAV	MALE	INDIAN	IND-123	9876543210	C1-501
ISHA	FEMALE	INDIAN	IND-456	8765432109	C1-501
VIHAAN	MALE	INDIAN	IND-789	7654321098	C1-501
ANAYA	FEMALE	INDIAN	IND-234	6543210987	C1-502
ARJUN	MALE	INDIAN	IND-567	5432109876	C1-502
AARAV	MALE	INDIAN	IND-123	9876543210	C1-501
ISHA	FEMALE	INDIAN	IND-456	8765432109	C1-501
VIHAAN	MALE	INDIAN	IND-789	7654321098	C1-501
ANAYA	FEMALE	INDIAN	IND-234	6543210987	C1-502
ARJUN	MALE	INDIAN	IND-567	5432109876	C1-502
SARA	FEMALE	INDIAN	IND-890	4321098765	C1-502
AYAN	MALE	INDIAN	IND-345	3210987654	C1-601
ZARA	FEMALE	INDIAN	IND-678	2109876543	C1-601
ARNAV	MALE	INDIAN	IND-901	1098765432	C1-601
AISHA	FEMALE	INDIAN	IND-456	9876543210	C1-602
VIVAAN	MALE	INDIAN	IND-789	8765432109	C1-602
SOFIA	FEMALE	INDIAN	IND-012	7654321098	C1-602
REYAN	MALE	INDIAN	IND-567	6543210987	C1-701
ANVI	FEMALE	INDIAN	IND-890	5432109876	C1-701
KABIR	MALE	INDIAN	IND-123	4321098765	C1-701

## THE TICKET BOOKING TABLE

id	name	flightcode	gender	passportnumber	amount	nationality
P001	YASHASVI	C1-101	MALE	IND-123	3500	INDIAN
P002	LOPAMUDRA	C1-101	FEMALE	IND-321	3500	INDIAN
P003	EESHANK	C1-102	MALE	IND-654	3800	INDIAN
P004	ARMAN	C1-102	MALE	IND-911	3800	INDIAN
P005	MEDHA	C1-201	FEMALE	IND-119	3800	INDIAN
P006	ANSH	C1-201	MALE	IND-789	3800	INDIAN
P007	ADARSH	C1-202	MALE	IND-987	4000	INDIAN
P008	VANSH	C1-202	MALE	IND-111	4000	INDIAN
P009	KEITH	C1-301	MALE	IND-456	3600	INDIAN
P010	ARPIYH	C1-301	MALE	IND-222	3600	INDIAN
P011	ANUJ	C1-302	MALE	IND-333	3200	INDIAN
P012	CHARAN	C1-302	MALE	IND-444	3200	INDIAN
P013	AASHISH	C1-401	MALE	IND-555	4200	INDIAN
P014	NANDHA	C1-401	MALE	IND-696	4200	INDIAN
P015	YASH	C1-402	MALE	IND-777	4500	INDIAN
P016	RAHIMA	C1-402	FEMALE	IND-888	4500	INDIAN
P018	VISHAL	C1-502	MALE	IND-312	2590	INDIAN

## 7. CONCLUSION

The development and implementation of an effective airport management system are paramount for enhancing overall operational efficiency, ensuring passenger satisfaction, and addressing the complex challenges faced by modern airports. Through the review of existing literature and considering key features such as flight code, source, destination, take-off time, and the number of seats, it is evident that a well-designed and technologically advanced airport management system can bring about significant improvements.

The incorporation of advanced technologies, such as data analytics, artificial intelligence, and smart infrastructure, offers the potential to streamline various processes within the airport ecosystem. This includes optimizing air traffic management, improving passenger experience through streamlined ticket booking systems, and ensuring the seamless coordination of ground operations. Moreover, the integration of sustainable practices and adherence to international standards contribute to the long-term viability and success of airport facilities.

While our discussion has highlighted the importance of certain features in a ticket booking system, it is crucial to recognize that a comprehensive airport management system goes beyond ticketing. It encompasses diverse aspects such as security, safety, regulatory compliance, and financial sustainability. The cooperation of various stakeholders, including airlines, airport authorities, regulatory bodies, and technology providers, is vital for the successful implementation and continuous improvement of such systems.

As airports continue to evolve in response to technological advancements and the changing landscape of global travel, the implementation of innovative solutions becomes a strategic imperative. The ongoing commitment to research, development, and collaboration will play a pivotal role in shaping the future of airport management, contributing to enhanced efficiency, safety, and the overall passenger experience.

In essence, a forward-looking and adaptive approach to airport management systems is essential for addressing the evolving needs of the aviation industry and ensuring that airports remain key facilitators of global connectivity.

## **REFERENCES**

1. <https://www.youtube.com/watch?v=gI4AYeqV3Xk>
2. <https://github.com/Samyak005/Central-Airport-Management-System-Database>
3. <https://github.com/Ishan-sinha/Airport-Management-System>
4. <https://myprojectideas.com/airline-reservation-system-in-java-java-project/>