# Airplane Management System
# (Algorithm Based System)

**B.Sc. Level 2 Semester II PROJECT REPORT**
**Course Code: CSE 254 Course Title: Object Oriented Programming(JAVA) Sessional**

## Prepared By

Amit Hasan Sikder(ID:2102043)

## Course Teacher

Professor Dr. Md. Abdullah Al Mamun


**Department of Computer Science and Engineering**
**Hajee Mohammad Danesh Science and Technology University,**
**Dinajpur-5200**

# Table of Contents



# List of Figures



# List of Tables

# 1. Introduction

The Airplane Management Application is a graphical user interface (GUI)-based desktop application developed in Java, aimed at managing and visualizing airport routes, calculating shortest paths between airports using the Floyd-Warshall algorithm, and analyzing product capacities via a knapsack problem solver. The application is designed to assist airport managers and logistics teams in efficiently handling flight routes and optimizing cargo capacity management.

# 2. Objectives and Goals

**The primary objectives of this project are:**

1. To create a user-friendly application for managing flight routes between airports.
2. To provide an efficient way of calculating the shortest paths between multiple airports using advanced algorithms.
3. To incorporate a dynamic visualization of airport locations and flight routes on a map.
4. To allow users to manage and solve the 0/1 knapsack problem related to airport cargo capacities.
5. To offer a time-based feature to display a real-time world clock within the application.
6. To enhance the visual experience with a slideshow of airplane-themed background images.

**Project Goals:**

The goals for the Airplane Management App include:

1. Simplifying the management of flight routes by offering intuitive input mechanisms (dropdowns, buttons, etc.).
2. Implementing efficient algorithms like Floyd-Warshall for route optimization and shortest path calculation.
3. Providing a dynamic, interactive map showing airport connections and their geographical locations.

4. Creating a flexible solution for product capacity management using the knapsack algorithm.

5. Enriching the user experience through real-time features (e.g., the world clock) and aesthetic elements (e.g., background image slideshows).

# 3.Project Overview

The Airplane Management App provides a clean and intuitive interface for managing flight routes, calculating shortest paths, and visualizing airports and routes on a map. The application allows users to input airports, set distances between them, and visually display connections. Furthermore, the app has additional features such as product capacity management using the knapsack algorithm and a real-time world clock. The user interface is built using Java Swing, while the back-end algorithms handle complex route optimization and capacity management tasks.

# 4. Functionalists

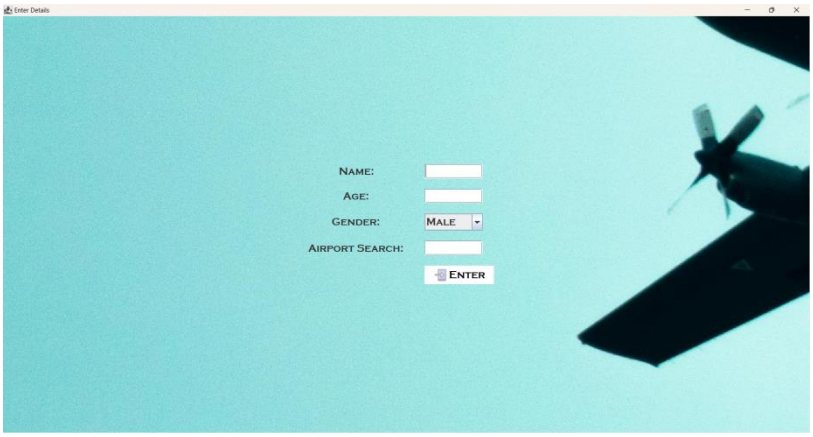
## 4.1 User Interface (UI)

The UI is designed using Java Swing, providing the following components:

Airport Input: Users can select airports from two dropdown menus to create a route between them.

Distance Field: A text field allows the user to input the distance (in kilometers) between two airports.

**Fig 4.1:Enter Window**



**Buttons for Actions:**

Add Route: Adds a new route between two selected airports and displays the route on the map.

Sort Results: Sorts and displays the results of the shortest path calculations in ascending order.

Calculate Shortest Paths: Calculates the shortest paths between all airport pairs using the Floyd-Warshall algorithm.

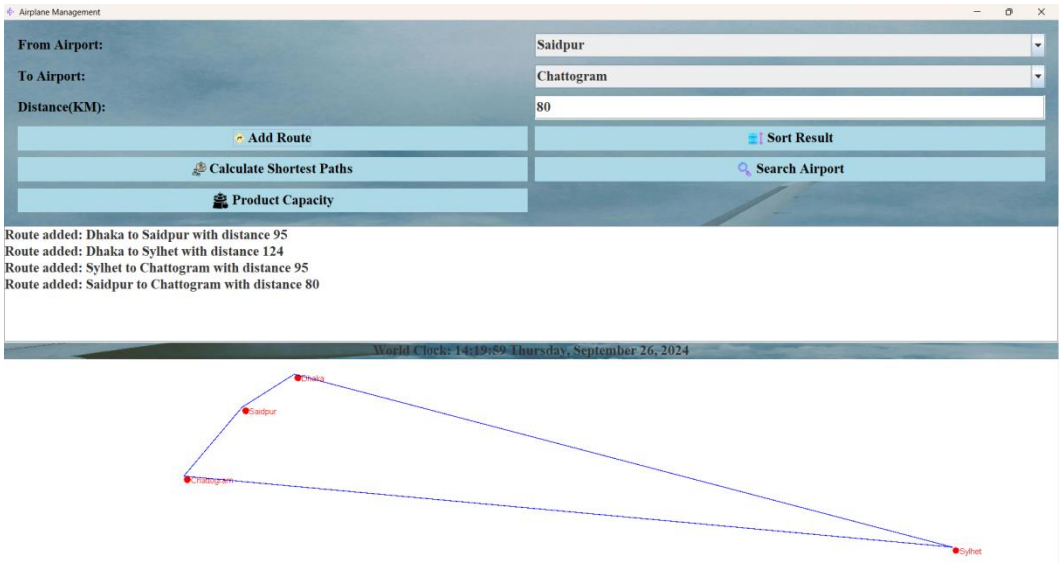
Search Airport: Searches for a specific airport using a linear search mechanism.

Product Capacity: Opens a separate window to manage product capacities using a 0/1 knapsack algorithm.

## 4.2 Route Management

Users can manage routes by selecting two airports and specifying the distance between them. The application uses a distance matrix to store and compute these routes, allowing efficient access and manipulation.

**Fig 4.2: Route Add**



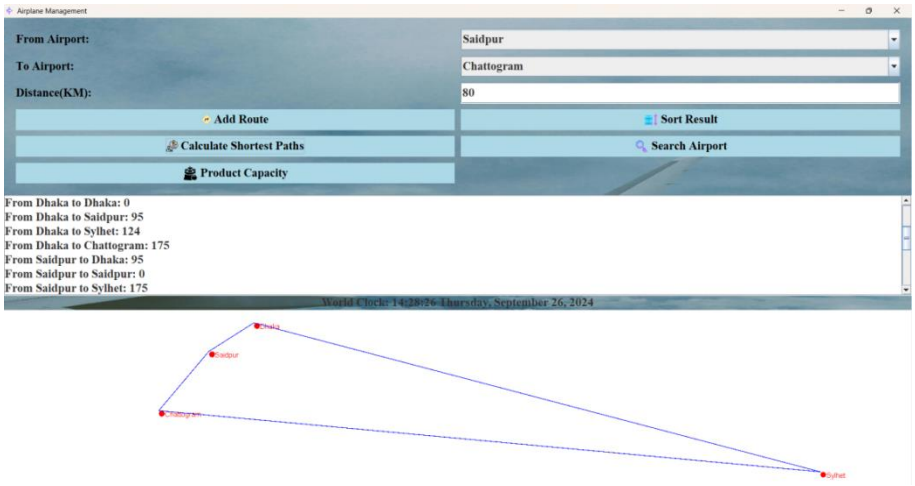

## 4.3 Shortest Path Calculation

The application employs the Floyd-Warshall algorithm to calculate the shortest paths between all pairs of airports. Results are displayed in a user-friendly format and stored in the application for further operations like sorting.

## 4.4 Map Visualization

The app features a panel that displays airport locations and routes. The airports are represented by red dots, while blue lines connect airports that have routes between them. Randomized points represent airport locations for visual demonstration.

**Fig 4.3: Calculate Shortest Path**




## 4.5 World Clock

A real-time world clock displays the current time, day, and date. This clock updates every second using Java's Timer class.

## 4.6 Background Image Slide show

The background of the app features a rotating slideshow of airplane-themed images. The slideshow changes images every 5 seconds, adding aesthetic appeal to the user experience.

## 4.7 Product Capacity Management (Knapsack Problem)

A separate window allows users to input product weights, values, and knapsack capacities. The application then calculates the maximum value that can be carried using the **0/1 Knapsack algorithm**, helping airport logistics teams to optimize their cargo loads.

**Fig 4.4: Calculate  Knapsack Capacity**

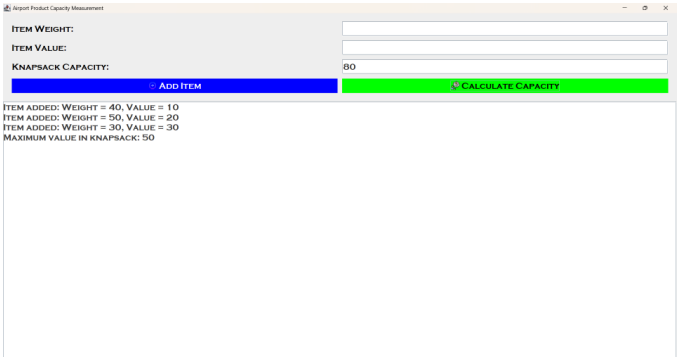
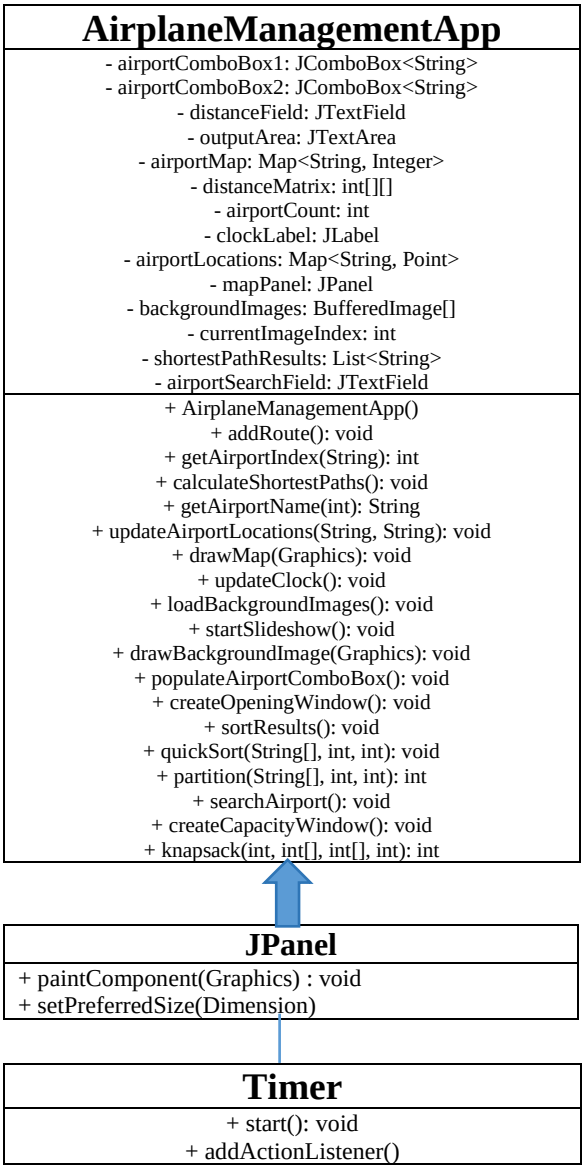



**Fig. 4.5 UML(Unified Modeling Language)**



### 4.8 Time Complexity and Space Complexity

| | |
|---|---|
| **Time Complexity** | **$O(n^3)$** |
| **Space Complexity** | **O(n*W)** |
| | |

**Table 4.1 Complexity**

## 5. Future Goals and Enhancements

### 5.1 Dynamic Airport Locations

In the current implementation, airport locations are randomized. Future versions could integrate real geographic coordinates to provide accurate, real-world mapping of airports.

### 5.2 Improved Route Visualization

Enhancements can include advanced mapping techniques such as integrating external mapping APIs (e.g., Google Maps) to display routes between airports in real-world locations.

### 5.3 Database Integration

Future updates could incorporate database storage for airport data and routes, allowing persistence across sessions and easier scalability.

### 5.4 Flight Scheduling and Tracking

The application could be expanded to include flight scheduling, status tracking, and real-time route updates for airport and airline operations.

### 5.5 Machine Learning Integration

By integrating machine learning algorithms, the application could provide predictive analysis for flight delays, weather impacts, and optimal route suggestions based on historical data.

### 5.6 Advanced Capacity Management

The capacity management feature could be expanded to consider multiple constraints (e.g., time, fuel efficiency) and more advanced optimization techniques for real-world logistics solutions.

### 5.7 Cross-Platform Compatibility

Currently, the app runs as a desktop Java Swing application. In the future, the app could be ported to mobile platforms or as a web-based application to increase accessibility and usability.

### 5.8 Technologies:

| SI | Technologies Name |
|---|---|
| 01 | Java Swing |
| 02 | Floyed Warshall Algorithm |
| 03 | 0/1 Knapsack Algorithm |
| 04 | Linear Search Algorithm |
| 05 | Quick Sort Algorithm |

Table 5.1 Uses of Technologies

## 6. Limitations

1. No Database or File Storage
2. No Authentication.
3. Limited Real- time Functionalities

## 7.Conclusion

The Airplane Management Application successfully meets its objectives by providing a comprehensive tool for airport route management and cargo capacity analysis. Through its simple yet powerful features, the app enhances decision-making processes for airport managers and logistics teams. With future enhancements, it has the potential to become an invaluable tool for the aviation industry.