Flight Route Optimization System

In today's world, where travel and logistics are constantly evolving, optimizing flight routes is crucial for both individuals and businesses. Our innovative Flight Route Optimization System is designed to streamline travel planning, ensuring cost-effective and efficient journeys. This system leverages the power of graph theory and cutting-edge algorithms to analyze complex transportation networks and find the most optimal flight paths.

Purpose: Optimizing Flight Routes

The Flight Route Optimization System is designed to address the challenge of finding the most cost-effective and efficient flight routes. Whether you are a frequent traveler seeking to save money on airfare or a logistics company looking to optimize transportation costs, our system can help you achieve your goals. It takes into account various factors such as distance, cost, and flight duration to identify the most optimal route for your needs.

1. **Cost Minimization**

   The system prioritizes finding the most economical flight routes, minimizing your overall travel expenses.

2. **Distance Optimization**

   It considers the shortest flight distances to reduce travel time and minimize fuel consumption.

3. **Efficiency Enhancement**

   By identifying the most efficient routes, the system helps you avoid unnecessary layovers and delays, ensuring a smooth and seamless travel experience.

Minimizing Cost as Optimization Criteria

The Flight Route Optimization System prioritizes cost minimization as the key optimization criteria. This means that the system seeks to find the most economical flight route, balancing factors such as distance and duration to achieve the lowest possible travel cost.

Cost

The system considers the cost of each flight ticket, factoring in discounts, promotions, and other relevant factors.

Distance

While distance is considered, it is secondary to cost. The system prioritizes routes with lower costs, even if they might be slightly longer in distance.

Duration

The duration of the flight is also considered, but only to the extent that it does not significantly impact the overall cost of the route.

Graph-Based Approach to Representing Networks

The Flight Route Optimization System utilizes a graph-based approach to represent transportation networks. This approach allows for a structured and efficient way to model the complex relationships between cities and flights.

Cities as Nodes

Cities are represented as nodes within the graph, representing key locations in the transportation network.

Flights as Edges

Flights between cities are represented as edges, connecting the nodes and providing information about the flight route.

Attributes

Each edge is associated with attributes such as cost, distance, and flight duration, providing crucial data for route optimization.

Path Optimization Using Dijkstra's Algorithm

The Flight Route Optimization System utilizes Dijkstra's algorithm to calculate the shortest path between two cities. This algorithm efficiently analyzes the graph, considering the cost, distance, and other attributes of each flight, to find the most optimal route. It systematically explores the network, identifying the shortest path based on the chosen optimization criteria.

1

Step 1

Initialize distances from the starting node to all other nodes as infinity.

2

Step 2

Set the distance from the starting node to itself as 0.

3

Step 3

While there are unvisited nodes:

4

Step 4

Select the unvisited node with the shortest distance from the starting node.

5

Step 5

Mark the selected node as visited.

6

Step 6

Update the distances of all unvisited neighbors of the selected node.

7

Step 7

Repeat steps 3-6 until all nodes have been visited.

Key Benefits of the Flight Route Optimization System

The Flight Route Optimization System offers a wide range of benefits for both travelers and logistics companies, streamlining travel planning and improving operational efficiency. By leveraging advanced algorithms and a user-friendly interface, our system empowers users to make informed decisions, saving them time, money, and effort.

1 Cost Savings

The system helps users identify the most cost-effective flight routes, minimizing travel expenses.

2 Time Efficiency

By optimizing flight routes, the system reduces travel time and minimizes delays, ensuring a smooth and efficient travel experience.

3 Enhanced Decision-Making

The system provides comprehensive route information, empowering users to make informed decisions based on cost, distance, and duration.

4 Improved Operational Efficiency

For logistics companies, the system optimizes supply chain operations, ensuring timely deliveries and reducing transportation costs.

Dijkstra's Algorithm Workflow

1 Initialization

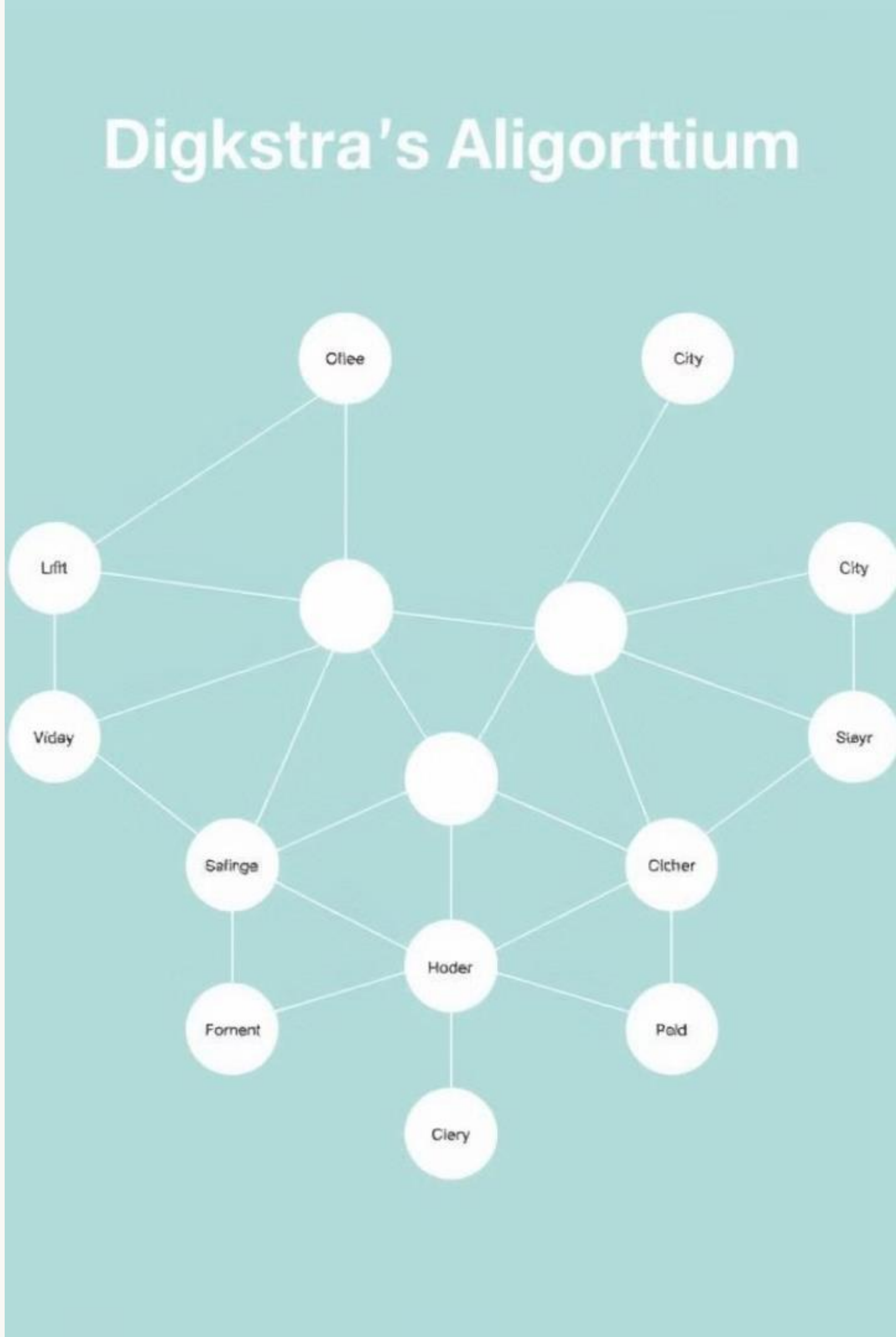
Set source city cost/distance to zero, others to infinity.

2 Priority Queue

Explore neighboring cities in ascending order of cost/distance.

3 Path Update

Update minimum path if a cheaper/shorter path is found.



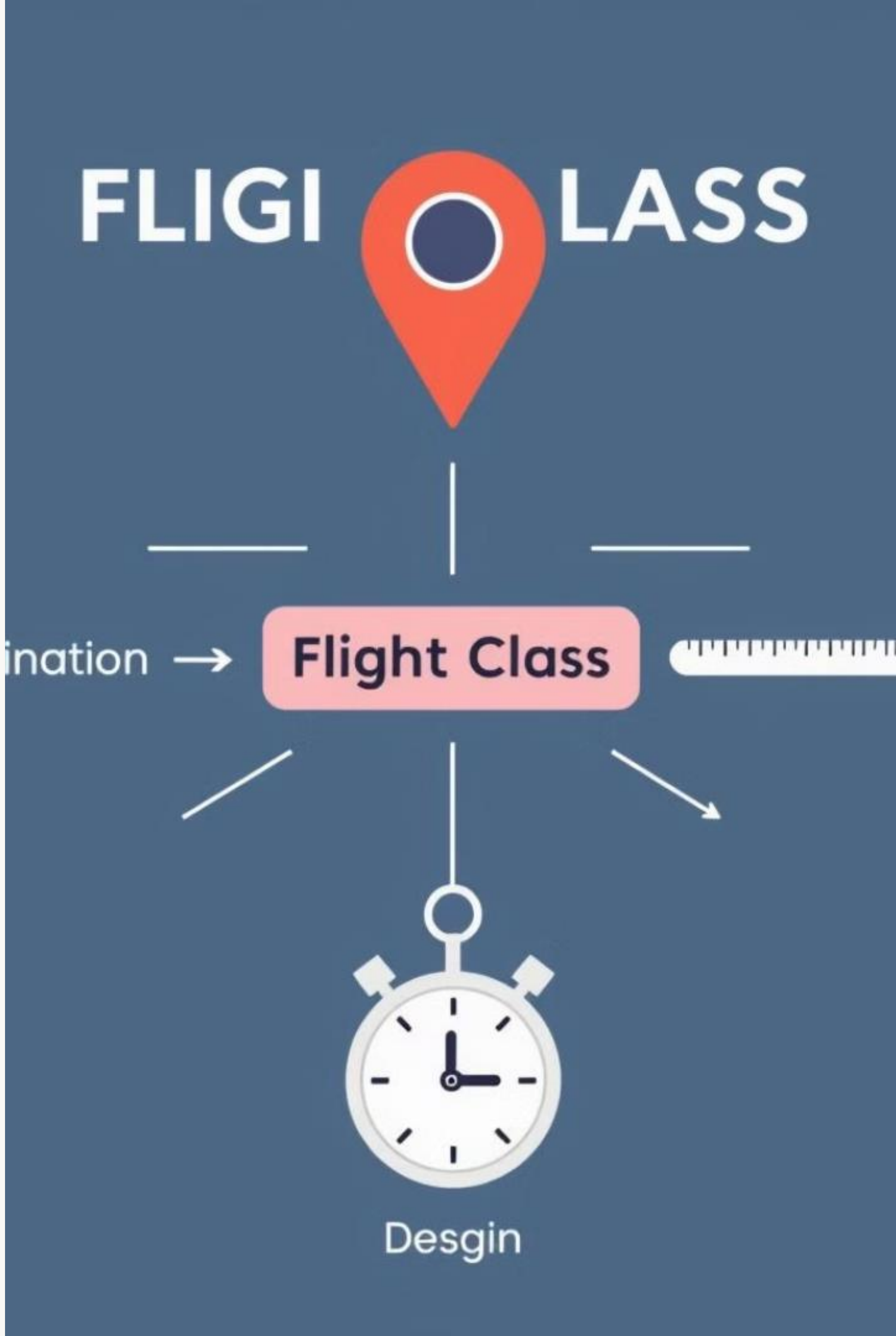
Flight Class Workflow

Flight Class

Data structure for flight information.

Attributes

Destination, cost, distance, and duration.



FlightGraph Class Workflow

1 Graph Management

Manages cities, flights, and connections.

2 Adjacency List

Stores flight connections between cities.

3 Methods

Add cities, flights, calculate costs and distances.

Display Module Workflow

Output

Presents optimized results in a user-friendly format.

Information

Travel cost, distance, and estimated fuel cost.

Unreachable Cities

Notifies the user if a city is unreachable.

City and Flight Addition

Add Cities

FlightGraph adds cities based on user input.

Add Flights

FlightGraph adds flights based on user input.

Shortest Path Calculation

dijkstraCost	Calculates shortest path based on cost.
dijkstraDistance	Calculates shortest path based on distance.

Display Output

1 Costs

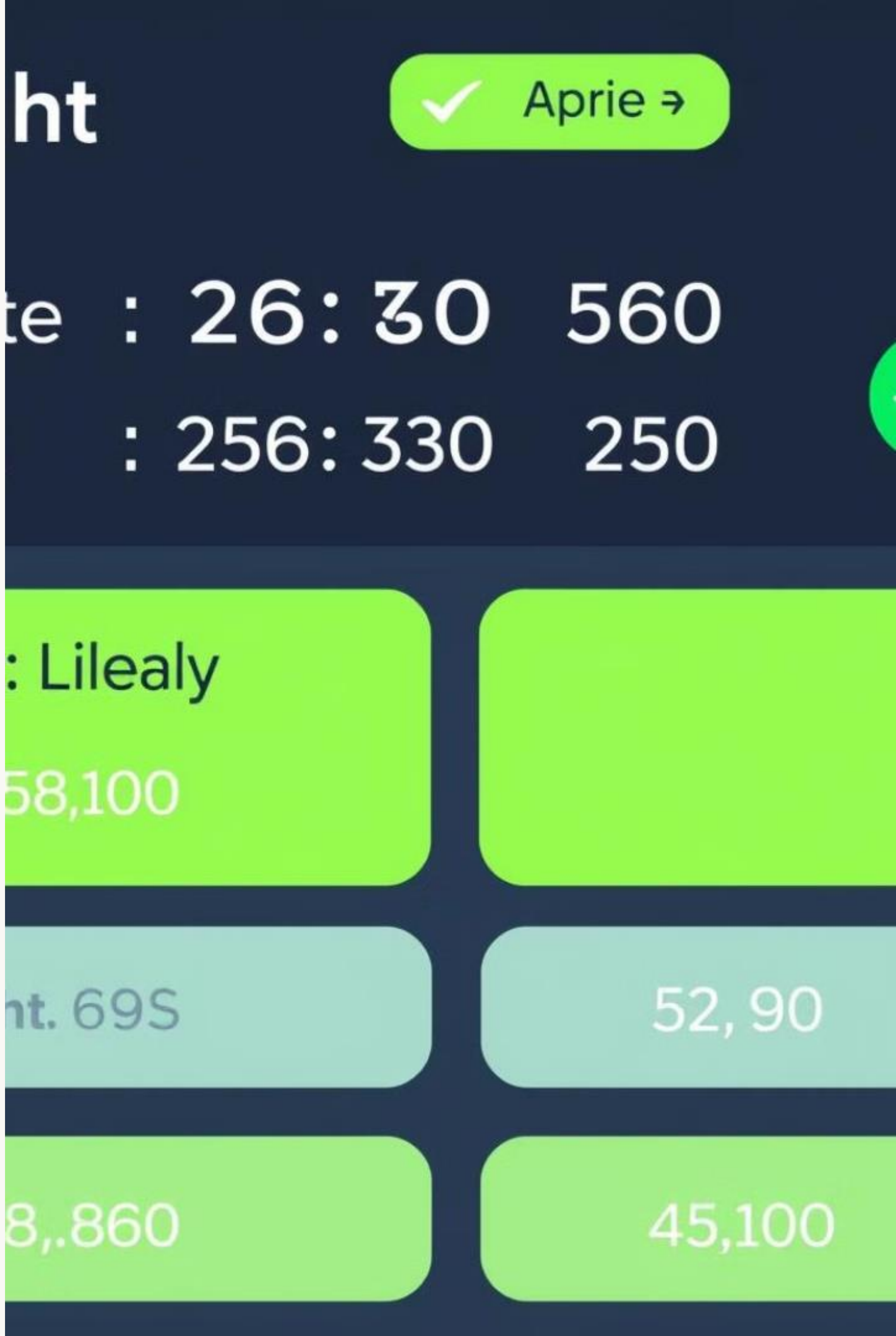
Displays minimum travel costs for each city.

2 Distances

Displays minimum travel distances for each city.

3 Fuel Estimates

Provides estimated fuel costs based on distances.



System Benefits

Modular Design

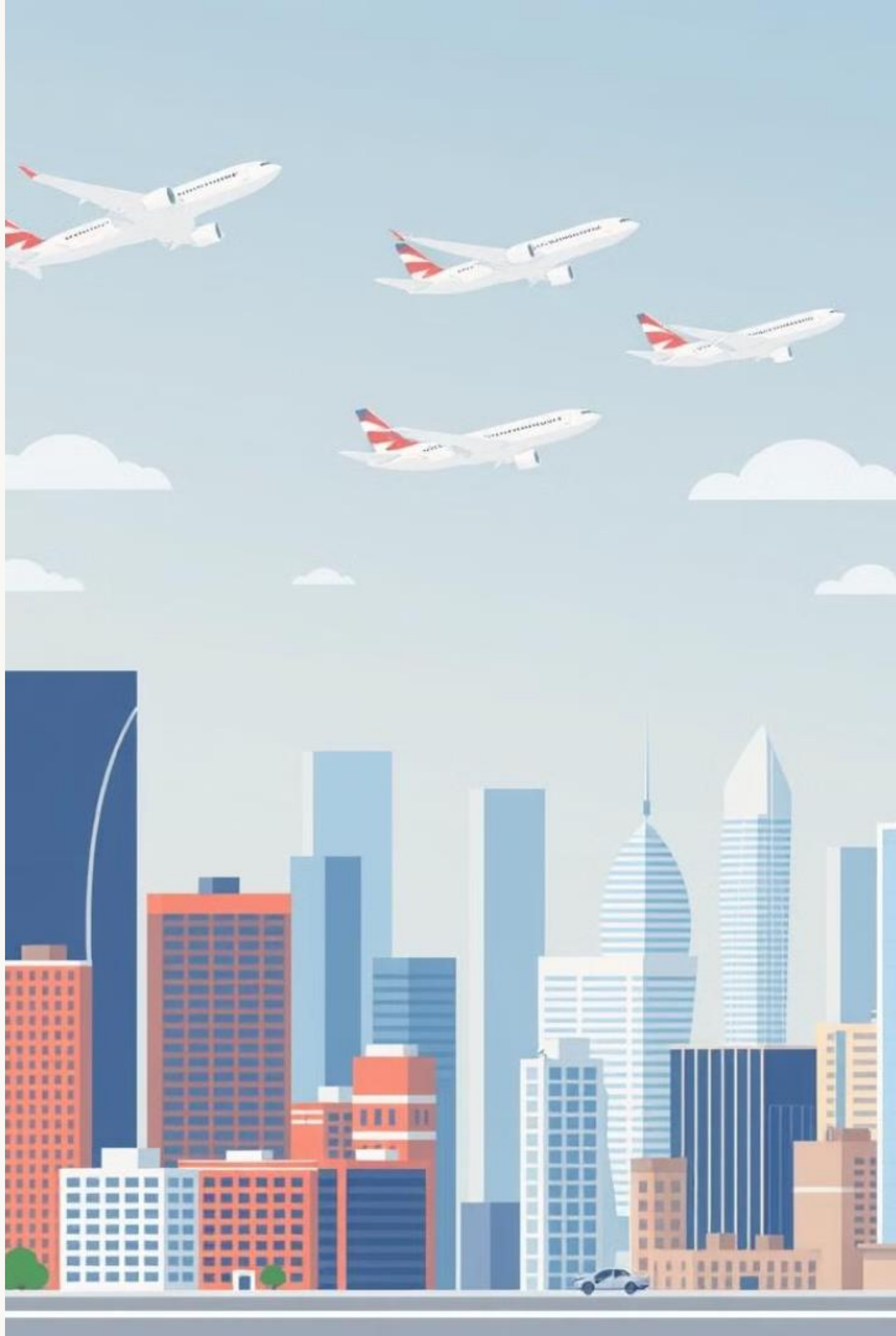
Separate components for easy modification and expansion.

Efficiency

Optimizes flight routes based on cost and distance.

Real-World Applications

Useful for transportation and travel logistics.



CODE :-

```
 #include <iostream>
#include <string>
#include <vector>
using namespace std;

void addFlight(string id, string dep, string dest, string date, string time, int totalSeats, double basePrice) {
   // Add flight logic
}

bool authenticateUser(string username, string password, bool &isAdmin) {
   // Authentication logic
   return true;
}

void bookTicket(string name, string nationality, int mobile, string date) {
   // Booking logic
}

void displayFlights(vector<string> flights, string date) {
   // Display flights logic
}

void cancelTicket(int bookingID, string date) {
   // Cancel ticket logic
}
```

```
int main() {
    string username, password;
    bool isAdmin = false;

    cout << "Enter Username: ";
    cin >> username;
    cout << "Enter Password: ";
    cin >> password;

    if (!authenticateUser(username, password, isAdmin)) {
        cout << "Invalid credentials! Exiting\n";
        return 0;
    }

    int choice;
    while (true) {
        cout << "Flight Management System\n";
        if (isAdmin) {
            cout << "1. Add Flight\n";
            cout << "2. View Payment Details\n";
        }
        cout << "3. Book Ticket\n";
        cout << "4. Display All Flights for a Date\n";
        cout << "5. Display Passenger List for a Flight\n";
        cout << "6. Cancel a Booking\n";
        cout << "7. Exit\n";
cout << "Enter your choice: ";
        cin >> choice;

        if (isAdmin && choice == 1) {
            string id, dep, dest, date, time;
            int totalSeats;
            double basePrice;
            cout << "Enter Flight ID: ";
            cin >> id;
            cout << "Enter Departure Location: ";
            cin >> dep;
            cout << "Enter Destination: ";
            cin >> dest;
            cout << "Enter Date (YYYY-MM-DD): ";
            cin >> date;
            cout << "Enter Time of Departure: ";
            cin >> time;
            cout << "Enter Total Seats: ";
            cin >> totalSeats;
            cout << "Enter Base Price: ";
      cout << "Enter Base Price: ";
            cin >> basePrice;
            addFlight(id, dep, dest, date, time, totalSeats, basePrice);
        } else if (isAdmin && choice == 2) {
            string flightID, date;
            cout << "Enter Flight ID to view Payment Details: ";
```

```
cin >> flightID;
          cout << "Enter Date of Flight (YYYY-MM-DD): ";
          cin >> date;
          // Logic to view payment details
       } else if (choice == 3) {
          string name, nationality, date;
          int mobile;
          cin.ignore();
          cout << "Enter Passenger Name: ";
          getline(cin, name);
          cout << "Enter Nationality: ";
          cin >> nationality;
          cout << "Enter Mobile Number: ";
          cin >> mobile;
          cout << "Enter Date of Flight (YYYY-MM-DD): ";
          cin >> date;
          bookTicket(name, nationality, mobile, date);
       } else if (choice == 4) {
          string date;
          cout << "Enter Date of Flight (YYYY-MM-DD) to view available flights: ";
          cin >> date;
          vector<string> flights; // Example vector for flights
          displayFlights(flights, date)
```

```
} else if (choice == 5) {
            string flightID, date;
            cout << "Enter Flight ID to view Passenger List: ";
            cin >> flightID;
            cout << "Enter Date of Flight (YYYY-MM-DD): ";
            cin >> date;
            // Logic to display passenger list
        } else if (choice == 6) {
            int bookingID;
            string date;
            cout << "Enter Booking ID to cancel: ";
            cin >> bookingID;
            cout << "Enter Date of Flight (YYYY-MM-DD): ";
            cin >> date;
            cancelTicket(bookingID, date);
        } else if (choice == 7) {
            cout << "Exiting Flight Management System.\n";
            break;
        } else {
            cout << "Invalid choice! Please try again.\n";
        }
    }
    return 0;
}
```

Output :-

```
  Enter Username: admin

Enter Password: adminpass

*** Flight Management System ***

1. Add a Flight

2. View Payment Details

3. Book a Ticket

4. Display All Flights for a Date

5. Display Passenger List for a Flight

6. Cancel a booking

7. Exit

Enter your choice: 3

Enter Passenger Name: John Doe

Enter Nationality: USA

Enter Mobile Number: 1234567890

Enter Date of Flight (YYYY-MM-DD): 2023-12-01 Booking Successful!
```