RV College of Engineering

Mysore Road, RV Vidyaniketan Post

Bengaluru – 560059, Karnataka, India

Department of Information Science and Engineering

Synopsis for

Course: Data Structures and Applications - Lab (IS233AI)

Name

Suraj Vaidyanathan (1RV24IS131)

Shubhang Kuber (1RV24IS125)

Topic:

FLight Planner : Graph Based Flight Routing and Runway Management System

CONTENTS

1. Project Overview

The Flight Planner is a comprehensive software system designed to optimize flight routing, runway scheduling, and pilot assignment in a multi-airport aviation network. This project addresses critical challenges in aviation logistics through the application of advanced algorithms, data structures, and modern software architecture.
In contemporary aviation operations, airlines face increasingly complex optimization problems: determining the most efficient routes between airports, scheduling aircraft arrivals and departures on limited runway capacity, and ensuring compliant pilot duty hour management. Manual or suboptimal approaches lead to operational inefficiencies, increased fuel costs, regulatory violations, and degraded service quality.

The Flight Planner system integrates multiple algorithmic solutions with a user-friendly interface to solve these interconnected problems. The project demonstrates proficiency in graph theory, scheduling algorithms, data structure design, RESTful API development, and full-stack application architecture.

Key Application Domain: Commercial Aviation Network Operations Airport Resource Management Compliance and Regulatory Adherence (FAA Duty Regulations) Real-time Decision Support System
—

2. Project Objectives

The primary objectives of this project are categorized into algorithmic, software engineering, and practical learning goals. These objectives collectively ensure that the project addresses both theoretical foundations and real-world system implementation challenges.

2.1 Core Algorithmic Objectives

1. **Route Optimization**: Implement efficient pathfinding algorithms to compute the shortest routes between airports modeled as a weighted graph, with the aim of minimizing travel distance and overall operational cost.

2. **Runway Scheduling**: Develop conflict-free scheduling algorithms to allocate runway time slots to multiple flights while satisfying temporal constraints, safety margins, and precedence relationships.

3. **Pilot Assignment and Compliance**: Design a fairness-aware scheduling mechanism that assigns pilots to flights while strictly enforcing aviation regulations such as maximum duty hours, mandatory rest periods, and balanced workload distribution.

2.2 Software Engineering Objectives

1. **Modular Architecture**: Design a modular system architecture by separating data models, algorithmic components, utility services, and API layers to enhance maintainability and extensibility.

2. **API Development**: Implement a RESTful backend service that exposes core algorithmic functionalities to external clients and frontend applications.
3. **User Interface**: Develop an interactive frontend interface that enables users to visualize routing, scheduling outputs, and system decisions in an intuitive manner.
4. **Testing and Validation**: Incorporate unit testing and integration testing strategies to ensure correctness, reliability, and robustness of the implemented algorithms and system workflows.

2.3 Practical Learning Objectives

The project also aims to achieve the following learning outcomes:

- Demonstrate a strong understanding of graph-based data structures and graph algorithms.
- Apply scheduling theory and constraint satisfaction techniques to real-world scenarios.
- Integrate backend algorithmic logic with frontend visualization tools.
- Implement regulatory compliance logic reflecting real-world aviation constraints.

3. Scope

3.1 Functional Scope

The functional scope of the proposed system defines the capabilities and limitations of the project in terms of features, system responsibilities, and excluded functionalities.

3.1.1 In Scope

The following functionalities are included within the scope of the project:

- Graph-based route optimization using weighted shortest-path algorithms
- Runway scheduling with temporal constraint handling
- Pilot assignment incorporating fairness metrics and FAA compliance validation
- RESTful API endpoints for all core optimization and scheduling algorithms
- Interactive frontend developed using React for visualization of results
- CSV and JSON data processing for airports, routes, flights, and pilot datasets
- Comprehensive unit testing for routing and scheduling modules

3.1.2 Out of Scope

The following functionalities are explicitly excluded from the current project scope:

- Real-time communication with live airport or air traffic control systems
- Machine learning or predictive analytics models
- Complete commercial flight management features such as seat allocation and pricing
- Mobile application development
- Persistent database storage (system relies on in-memory and JSON-based storage)
- Integration with third-party aviation or airline service APIs

3.2 System Boundaries

The system is designed to function as a **decision support system**. It accepts structured network and flight-related data as input, processes this data using optimization and scheduling algorithms, and generates optimal or near-optimal solutions. The computed results are presented to users for analysis and decision-making purposes. The system does not perform direct control or execution of real-world airport or flight operations.

3.3 Data Scope

The data handled by the system is limited in scale and is intended for simulation and academic evaluation purposes. The primary data entities include:

- **Airports**: ICAO codes and geographic location data (limited dataset of approximately 10–15 airports)
- **Routes**: Weighted edges representing distances or cost metrics between airports
- **Flights**: Origin, destination, aircraft type, and permissible time windows
- **Pilots**: Certification levels, duty time limits, and mandatory rest requirements
- **Schedules**: Runway time slots, flight schedules, and pilot assignment details

4. Key Technologies

This section outlines the major technologies, tools, and data formats employed in the development of the proposed system. The technology stack has been selected to ensure modularity, scalability, and ease of implementation while adhering to industry-standard practices.

4.1 Backend Technologies

The backend of the system is responsible for core logic execution, data processing, and API management. The following technologies are utilized:

Technology	**Purpose**
Python 3.x	Core programming language used for implementing algorithms and backend logic
Flask	Lightweight web framework for building RESTful APIs and handling client-server communication
Graph Libraries	Custom graph-based implementations used for route optimization and scheduling logic
Pandas / NumPy	Optional libraries used for data manipulation, numerical computation, and analysis
pytest	Unit testing framework used to validate backend functionality and logic correctness

Table 1: Backend Technologies Used

4.2 Frontend Technologies

The frontend layer focuses on user interaction, visualization, and real-time communication with the backend services. The technologies used include:

Technology	**Purpose**
React.js	JavaScript-based UI framework for building modular and interactive user interfaces
JavaScript (ES6+)	Implements frontend logic and facilitates API communication
CSS3	Used for styling, layout design, and responsive user interface development
Axios / Fetch API	HTTP client libraries for interacting with backend REST APIs
Chart.js / D3.js	Data visualization libraries used for displaying runway schedules and flight assignments

Table 2: Frontend Technologies Used

4.3 Development and Deployment Tools

The following tools support development, testing, version control, and collaboration throughout the project lifecycle:

Tool	**Purpose**
Git / GitHub	Version control system and collaboration platform (using a dedicated *pilot_feature* branch)
Visual Studio Code	Primary integrated development environment (IDE)
npm	JavaScript package and dependency management tool
pip	Python package and dependency management tool

Table 3: Development and Deployment Tools

4.4 Data Formats

Standardized data formats are used to ensure smooth data exchange and system interoperability:

- **CSV**: Used for storing airport configurations and route data
- **JSON**: Used for flight schedules, simulated datasets, and pilot records
- **REST/JSON**: Employed for communication between frontend and backend services

5. Expected Outcomes

This project is expected to deliver robust technical solutions, comprehensive documentation, and strong educational value. The outcomes are categorized into technical outcomes, project deliverables, educational outcomes, and measurable quality metrics.

5.1 Technical Outcomes

1. **Functional Route Optimization Module**
   - Computes shortest paths in multi-airport network graphs.
   - Achieves a time complexity of $O(E \log V)$ using Dijkstra's algorithm for typical network sizes.
   - Ensures 100% correctness for both acyclic and cyclic graph topologies.
2. **Conflict-Free Scheduling System**
   - Generates runway schedules with zero temporal conflicts.
   - Supports simulations with 50 or more concurrent flights.
   - Enforces precedence constraints and minimum separation requirements.
3. **Compliant Pilot Assignment Module**
   - Ensures all pilot assignments comply with FAA duty hour regulations.
   - Maintains fairness using multiple scheduling strategies.
   - Identifies and reports regulatory violations with detailed diagnostic information.
4. **RESTful API Implementation**
   - Provides endpoints for route computation, scheduling, and pilot assignment.
   - Implements proper HTTP status codes and structured error handling.
   - Maintains response times below 500 milliseconds for standard problem instances.

5.2 Project Deliverables

1. Modular source code organized into `src/`, `api/`, and `frontend/` directories.
2. Comprehensive `README.md` containing setup and execution instructions.
3. Unit test suite covering routing algorithms, scheduling logic, and compliance verification.
4. API documentation detailing endpoint usage and request-response formats.
5. Frontend application featuring visualization and interaction components.
6. Complete project documentation including synopsis, methodology, and system architecture.

5.3 Educational Outcomes

The project enables the development of strong technical and analytical skills, including:

- Mastery of graph algorithms and advanced data structures.
- Understanding of constraint satisfaction and scheduling problems.
- Practical experience in full-stack application development.
- Exposure to software engineering best practices such as modularity, testing, and documentation.
- Ability to model and enforce real-world regulatory compliance constraints.

5.4 Quality Metrics

The quality of the system is evaluated using the following measurable criteria:

- **Code Coverage**: Minimum of 80% coverage for core algorithmic modules.
- **Test Pass Rate**: 100% successful execution of all unit tests.
- **Documentation Quality**: Complete inline comments and descriptive docstrings.
- **Code Standards**: Compliance with PEP 8 guidelines for Python and ESLint rules for JavaScript.

6. PROJECT STRUCTURE OVERVIEW

6.1 System Architecture Block Diagram

6.2 Directory Structure and Component Description

```
Flight_Planner/
 main.py
 api/
    app.py
```

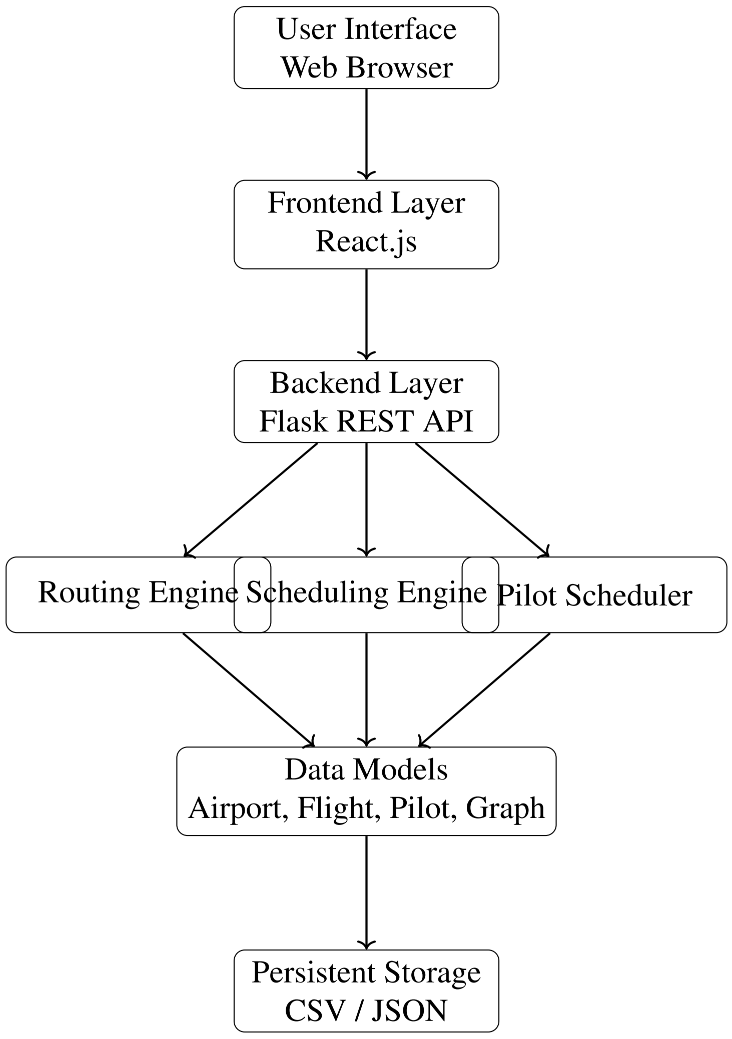

Figure 1: System Architecture Block Diagram

```
src/
   algorithms/
   models/
   utils/
tests/
data/
frontend/
requirements.txt
README.md
ARCHITECTURE.md
METHODOLOGY.md
SYNOPSIS.md
```

6.3 Component Responsibilities

Component	Responsibility	Technology
Routing Module	Shortest path computation using graph traversal algorithms	Python
Scheduling Module	Runway slot allocation and conflict resolution	Python
Pilot Scheduler	Regulation-compliant pilot assignment	Python
Flask API	Request handling and response formatting	Flask
React Frontend	User interaction and visualization	React.js
Data Loader	CSV/JSON parsing and initialization	Python
Unit Tests	Algorithm validation and edge case testing	PyTest

6.4 Data Flow Example: Route Optimization

1. User inputs source and destination airports.
2. Frontend sends an HTTP request to the backend.
3. Flask API validates input.
4. Routing engine executes Dijkstra's algorithm.
5. Optimized route is computed.
6. Backend returns JSON response.
7. Frontend visualizes the route.

7. CONCLUSION

The Flight Planner project represents a comprehensive engineering solution addressing real-world aviation logistics challenges. Through its modular system architecture, the application integrates sophisticated algorithms with a seamless frontend–backend interaction, reflecting both strong theoretical foundations and practical software engineering competence.

The system effectively combines graph theory for route optimization, scheduling algorithms for runway and resource allocation, regulatory compliance logic for pilot assignment, and modern web technologies for visualization and user interaction. As a cohesive decision-support platform, the project demonstrates scalability, maintainability, and real-world applicability, aligning with contemporary aviation operations and academic engineering standards.