Software Requirements Document (SRD)

Project Title: Flight Route Optimizer

1. Introduction

1.1 Purpose
- The purpose of this document is to outline the requirements for the Flight Route Optimizer application. This application aims to optimize flight routes by finding the shortest path between airports based on distance using Dijkstra's algorithm. It provides a web interface for users to input origin and destination airports and receive the shortest route and its cost.

1.2 Scope
- This project covers data ingestion and cleaning, storage in a PostgreSQL database, implementation of Dijkstra's algorithm for route planning, and a web interface using Flask. Additionally, the system will provide real-time data updates for estimated time remaining and health checks.

1.3 Definitions, Acronyms, and Abbreviations
- CSV: Comma-Separated Values
- SRD: Software Requirements Document
- HLD: High-Level Design
- LLD: Low-Level Design
- API: Application Programming Interface
- UI: User Interface

1.4 References
- N/A

1.5 Overview
- This document details the functional and non-functional requirements for the Flight Route Optimizer application. It includes system features, external interface requirements, system attributes, and other constraints.

2. General Description

2.1 Product Perspective
- The Flight Route Optimizer is a standalone web application designed to help users find the shortest flight route between two airports. It uses a backend server for data processing and a frontend interface for user interaction.

2.2 Product Functions
- Data ingestion and cleaning of flight routes
- Storage of flight routes in a PostgreSQL database
- Calculation of the shortest route using Dijkstra's algorithm
- User interface for inputting origin and destination airports
- Display of the shortest route and its cost
- Real-time updates for estimated time remaining and health checks

2.3 User Characteristics
- Users of the Flight Route Optimizer are expected to have basic knowledge of web applications and familiarity with airport codes.

2.4 Constraints
- The application relies on accurate and complete flight route data.
- The performance is dependent on the size of the dataset and database efficiency.
- The application should handle a reasonable number of concurrent users without significant performance degradation.

2.5 Assumptions and Dependencies
- PostgreSQL database is properly installed and configured.
- The web server is properly configured to serve the Flask application.
- Users have a reliable internet connection to access the web application.

3. Functional Requirements

3.1 Data Ingestion and Cleaning
- The system shall read flight route data from a CSV file.
- The system shall clean the data by selecting relevant columns and dropping rows with missing values.
- The system shall save the cleaned data to a new CSV file.

3.2 Database Storage
- The system shall create a `routes` table in the PostgreSQL database if it does not already exist.
- The system shall insert cleaned flight route data into the `routes` table.

3.3 Route Planning
- The system shall build a graph representation of airports and routes from the database.
- The system shall implement Dijkstra's algorithm to calculate the shortest route between two airports.
- The system shall return the cost and path of the shortest route.

3.4 Web Interface

- The system shall provide a home page with an input form for origin and destination airports.
- The system shall handle POST requests with origin and destination data.
- The system shall return the shortest route and its cost as a JSON response.
- The system shall display the result on the web page.
- The system shall allow users to zoom in, zoom out, and scroll the map.
- The system shall provide a button at the top right of the map for zoom controls.

3.5 Real-Time Data Updates

- The system shall fetch real-time data from an API every 50 seconds.
- The system shall display estimated time remaining and health check information on the web interface.
- The health check information shall include dummy data for fuel and engine status.
- Other data provided by the API shall be displayed as real-time data.

4. External Interface Requirements

4.1 User Interfaces

- Home Page: A simple form for users to input origin and destination airports.
- Result Display: A section to display the shortest route, its cost, estimated time remaining, and health check information.
- Map Controls: Zoom in, zoom out, and scroll functionality, with a button for zoom controls at the top right of the map.

4.2 Hardware Interfaces

- N/A

4.3 Software Interfaces

- PostgreSQL Database: For storing and querying flight route data.
- Flask Framework: For handling web requests and rendering templates.
- External API: For fetching real-time data updates.

4.4 Communications Interfaces

- HTTP/HTTPS: For communication between the client (web browser) and the server.
- API Calls: For fetching real-time data updates.

5. System Features

5.1 Data Ingestion

  - Description: Reads and cleans flight route data from a CSV file.
  - Priority: High
  - Stimulus/Response Sequences:
    - Stimulus: User provides a CSV file.
    - Response: System reads and cleans the data, saving it to a new CSV file.
  - Functional Requirements:
    - The system shall read flight route data from `input_file`.
    - The system shall clean the data and save it to `output_file`.

5.2 Database Setup
  - Description: Creates necessary database tables and inserts cleaned data.
  - Priority: High
  - Stimulus/Response Sequences:
    - Stimulus: Cleaned CSV file is available.
    - Response: System creates the `routes` table and inserts the data.
  - Functional Requirements:
    - The system shall create the `routes` table if it does not exist.
    - The system shall insert data from the cleaned CSV file into the `routes` table.

5.3 Route Calculation
  - Description: Calculates the shortest route between two airports.
  - Priority: High
  - Stimulus/Response Sequences:
    - Stimulus: User inputs origin and destination airports.
    - Response: System calculates and returns the shortest route and its cost.
  - Functional Requirements:
    - The system shall build a graph from the database data.
    - The system shall calculate the shortest route using Dijkstra's algorithm.
    - The system shall return the cost and path of the shortest route.

5.4 Web Interface
  - Description: Provides a user-friendly interface for inputting airport codes and displaying results.
  - Priority: High
  - Stimulus/Response Sequences:
    - Stimulus: User accesses the home page.
    - Response: System renders the form and handles input submissions.
  - Functional Requirements:
    - The system shall provide an input form on the home page.
    - The system shall handle POST requests with input data.
    - The system shall display the shortest route and its cost.
    - The system shall allow users to zoom in, zoom out, and scroll the map.
    - The system shall provide a button at the top right of the map for zoom controls.

5.5 Real-Time Data Updates

- Description: Provides real-time updates for estimated time remaining and health checks.
- Priority: High
- Stimulus/Response Sequences:
  - Stimulus: User views the route result page.
  - Response: System fetches and displays real-time data every 50 seconds.
- Functional Requirements:
  - The system shall fetch real-time data from an API every 50 seconds.
  - The system shall display estimated time remaining.
  - The system shall display health check information, including dummy data for fuel and engine status.

6. Non-Functional Requirements

6.1 Performance Requirements

- The system shall respond to route calculation requests within 5 seconds for datasets up to 10,000 routes.
- The system shall support at least 100 concurrent users without performance degradation.

6.2 Safety Requirements

- N/A

6.3 Security Requirements

- The system shall validate all user inputs to prevent SQL injection attacks.
- Sensitive data such as database connection details shall be stored securely.

6.4 Software Quality Attributes

- Reliability: The system shall be reliable, with a target uptime of 99.9%.
- Usability: The web interface shall be user-friendly and intuitive.
- Maintainability: The system code shall be well-documented to facilitate maintenance and updates.

6.5 Other Requirements

- The system shall be deployable on both local machines and cloud servers.

7. Other Requirements

7.1 Database Backup
- The system shall support regular database backups to prevent data loss.

7.2 Logging and Monitoring
- The system shall log significant events and errors to assist in debugging and maintenance.

This SRD document outlines the detailed requirements for the Flight Route Optimizer project, providing a comprehensive guide for development and implementation, including the addition of real-time data updates for estimated time remaining and health checks.