Software Requirements Specification

SkyHigh



– An application for Flight route optimization

Prepared By

Krithika Ravishankar ( 2022503035 )

K Buvanes Srivardan ( 2022503037 )

MIT Campus,

Anna University

**Table of Contents**

1. **Introduction**

1.1 Purpose  
1.2 Scope  
1.3 Definitions, Acronyms, and Abbreviations  
1.4 References  
1.5 Overview

1. **Overall Description**

2.1 Product Perspective  
2.2 Product Functions  
2.3 User Classes and Characteristics  
2.4 Operating Environment  
2.5 Design and Implementation Constraints  
2.6 Assumptions and Dependencies

1. **Functional Requirements**

3.1 User Management  
3.2 Flight Search and Optimization  
3.3 Budget Handling  
3.4 Flight Details Visualization  
3.5 Animation and Visualizer  
3.6 Time Input Management

1. **Non-Functional Requirements**

4.1 Performance Requirements  
4.2 Security Requirements  
4.3 Usability Requirements  
4.4 Scalability Requirements  
4.5 Availability and Reliability Requirements

1. **System Interfaces**

5.1 User Interface  
5.2 Third-Party Integration  
5.3 Database

1. **Conclusion**
2. **Introduction**
   1. **Purpose**

The primary aim of the SkyHigh application is to find the best routes for the passenger flights originating from one home airport to any destination and back, via one or many intermediate airports, without exceeding the maximum cost of all intermediate flights as defined by the user. The application will display available airlines and animation of the flight details alongside the routes map to the users.

* 1. **Scope**

SkyHigh is proposed as a Java based front-end and back-end application. This includes ability to enter the airport codes, choose an airline, and have information about the flight such as the takeoff and arrival times. An animated flight visualization will be part of the application to make it more user friendly.

* 1. **Definitions, Acronyms, and Abbreviations**
* **IATA:** International Air Transportation Association
* **GUI:** Graphic User Interface
* **API:** Application Programming Interface
  1. **References**

Java Development Kit'(JDK) Documentation.

Documentation for JavaFX to be used for Frontend Development.

APIs Related to Flight Information Integration

* 1. **Overview.**

This document presents a detailed account on the development of SkyHigh application including its features, the requirements that it has to fulfill and the design of the systems. This document is intended to help in the development and use of the application.

1. **Overall Description**
   1. **Product Perspective**

SkyHigh will be a desktop application, built solely using Java with a GUI for users to interact with. The application will make use of external APIs or databases for flight information and implement JavaFX for the visuals.

* 1. **Product Functions**
* **User Input**:

Users only can fill in home airport and destination airport, layovers, and preferred airline.

* **Flight Optimization:**

The most economical and the shortest routing are computed and illustrated.

* **Budget Handling:**

Requests the user budgeting and reports whether the chosen airline can be within the self-imposed budget.

* **Flight Details Visualization:**

Gives flight information like taking off or arrival times.

* **Animation and Visualizer:**

Gives a picture of a flight course including animation direction taking and landing.

* 1. **User Classes and Characteristics**
* **General Users:** Users who want to find the best possible routes between the designed away and cost schedules.
* **Travel Planners:** Users looking for all the flight details and pictures assisting in the planning.
  1. **Operating Environment**
* **Frontend:** GUI application developed using JavaFX.
* **Backend:** The application logic and data processing are done in Java.
* **Database/External APIs:** For downloading flight info and airlines.
  1. **Design and Implementation Constraints**
* **Performance:**

The application shall handle multiple user inputs and calculations efficiently. Must be able to handle the test cases of the algorithms used such as Dijkstra’s, Knapsack and BFS.

* **Data Accuracy:**

Flight data should be accurate and up-to-date.

* **Compatibility:**

The application should be compatible with major operating systems (Windows, macOS, Linux).

* 1. **Assumptions and Dependencies**
* Availability of external APIs or databases for flight information.
* Users have basic knowledge of airport codes and flight terminologies.

1. **Functional Requirement**
   1. **Introduction to User Management**

* It should allow users to enter airport codes, layovers, and select an airline for the domestic flight.
* Users will be able to suggest the time and date of arrival and departure.
  1. **Flight Search and Optimization**
* Look for particular flights the users input.
* Calculate cost and time for a particular route keeping the limits entered by the users.
  1. **Budget Handling**
* Ask the users to state the amount they are willing to spend.
* Check whether the flight options for that airline selected are within the budget stated.
  1. **Visualizing the Flight Details**
* Provide the flight details including the time of takeoff and the time of arrival at the destination.
* Give a list of airlines and available flights for the customers.
  1. **Animation and Visualizer**
* Show animations to flight routes via a map. Flyover simulation for a given flight route.
* Make interesting graphics and maps for presenting information using JavaFX.
  1. **Time Input Management**
* Enable users to state and manipulate timely arrangements for arrival and departure.
* Make sure that proper time entries have been made and complemented in the necessary computations.

1. **Non - Functional Requirement**
   1. **Performance Requirements**

* When the user operates the application, he/she should be able to do so within 2 seconds.
* All route preferences and calculations should be done with optimum efficiency.
  1. **Security Requirements**
* Concerns regarding handling of user data and exposure should be addressed in the application.
* All types of input should be well validated to handle possible intentional and accidental wrong inputs.
  1. **Usability Requirements**
* The timeline should be friendly; in this case, a graphical user interface should allow for easy access to and operations on the application.
* Distribute sufficient instructions and feedback regarding users on what actions to take during interactions.
  1. **Scalability Requirements**
* The total number of users and flight details the application will accommodate obviously increases with time.
* Additional features should be easy to incorporate in relation to software structure without too many complications.
  1. **Availability and Reliability Requirements**
* The application should always be ready for the user with no unexpected breakdowns.
* Always maintain the correctness of flight data against the scheduled changes.

1. **System Interface**
   1. **User Interface**

* The User enters the basic flight details and views results using the GUI based on JavaFX.
* Flight routes and their animations.
  1. **Third-Party Integration**
* Use of external APIs or databases for getting flight information and Airlines only.
* Appliance of external APIs / Data Sources and External Updates.
  1. **Database**
* Storing the user inputs and the results if any are feasible.
* The flight as well as the airline data is sub handled stored locally or through external databases.

1. **Conclusion**

The objective of SkyHigh application is to better and cheaper elongating and re-routing of users in a terminal wishing for the optimal flight routes. In order to meet both the frontend and backend needs of the application, Java will be settled. The application will have visualizations as well as interactive features. This SRS document therefore outlines the Core functionality requirements and the core design principles to be focused on to ensure the successful delivery of the application SkyHigh.