# SENG202 – Software Engineering Project Workshop

## 2020

## Project Description – Flight Data Manager

## Document history

This document will evolve during the first weeks of the course.

* Last update: July 13, 2020

## Introduction

The project is about designing, implementing, testing and releasing a flight data monitoring and analysis system. The purpose of the system is to support the recording, monitoring and analysis of flight and airport data. Individuals should be able to record and analyse flight data, see maps, graphs, and tables of flight-related information. Examples of such systems include SARA[[1]](#footnote-1), flightradar24[[2]](#footnote-2), etc.

**Requirements and features described in this document are intentionally somewhat open-ended. It is your responsibility to define the goal, scope, users and features of the app that are feasible and satisfy potential stakeholders, as well as communicating your reasoning. This may involve research and consultation with course staff.**

## Initial design considerations

* Your system should be implemented in Java as a desktop application (you can use Swing, JavaFX and other libraries). This will allow you to port your application to a web-based application or Android app later more easily if you wish to.
* Your app and deliverables must be compile-able and runnable on the lab machines in the CSSE computer labs (including the operating system, version of Java, IDEs, etc.).
* The application should be designed as a single-user application (similar to Microsoft Office or Firefox), i.e., one user would use an instance of the app on a device at a time. However, different users may use it at different times or on different devices (Office or Firefox).

## Overview of features

Key features are grouped into feature packages. Feature packages are not meant to map to project phases, but should help you scope the functionality and define concrete project tasks.

It is likely that not all features are listed that your team will have time to develop into detailed designs and to implement subsequently. Consequently, scoping your project is very important. While the first few feature packs should be regarded as essential, the later ones may be regarded as a partial order: it is likely that you will wish to negotiate some aspects of feature ordering and content. Furthermore, the features listed in the following sections may lack detail, may be contradictory, may differ in business value and may require widely different amounts of effort to implement. You, in collaboration with your group members and the teaching team, will refine the initial features and express the details in terms of UML, acceptance tests and other relevant documents.

**Feature package 1 (basic UI and loading existing data)**

* The main GUI of the application is split in several parts (panels) and has a menu bar.
* The application should be able to load airport data[[3]](#footnote-3), airline data[[4]](#footnote-4) and route data[[5]](#footnote-5) from files. We provide datasets for you to use as a starting point based on the sources listed above (a description of data formats and entries is available online[[6]](#footnote-6)). Note that the datasets from the above sources may contain many records and therefore may not be viewed in full in Microsoft Excel. Therefore, you may wish to use a subset of that data for initial development and testing. You can open the files in an ASCII text editor, such as Wordpad or Sublime Text, to view and search the data.
* The application can show airport data, airline data and route data in separate tables (“raw data viewer”). All data fields should be shown for a record.
* There are potentially many records in the provided data files. Therefore, it may be difficult for your app to show the entire content of a file at once. The “raw data viewer” should allow users to filter based on country. Filtering should be implemented as drop-down menu or a similarly usable mechanism.
* The application can load individual flights. Flight data can be generated and downloaded using an online service[[7]](#footnote-7) (we provide an example data file as CSV for a flight from Christchurch to Singapore Changi). Flight data should be shown after loading the related file.

**Feature package 2 (extended viewing and data filtering)**

* The “raw data viewer” should be extended to a more convenient viewer. For example, only basic data is shown in the list, but details become available when clicking on a record.
* The “raw data viewer” should allow users to filter data:
  + Airports: based on countries
  + Airlines: based on countries
  + Airlines: active or inactive
  + Routes: based on departure location
  + Routes: based on destination
  + Routes: direct flight or with stops
  + Routes: based on equipment

The filtering should be implemented as drop-down menu or a similarly usable mechanism.

**Feature package 3 (basic analyses)**

* Users should be able to calculate distance between two airports (based on difference of latitude and longitude of airports)
* Users should be able to search for flights based on search criteria.
* Users should be able to search for airports based on search criteria.
* Users should be able to search for routes based on search criteria.
* Users should be able to rank airports with most routes.
* Users should be able to rank airports based on fewest routes.
* Warning should be shown for airports with no routes.
* Integration of web search features to search for additional flight-related information.

**Feature package 4 (extended loading and data input)**

* If a new data file from an external source is uploaded, the user should have the option to a) append the data to already existing lists, or b) create a new list of the same type of data. If more than one list of the same type of data exists, some browsing mechanism must be implemented (e.g., a tree structure in one panel to select different record lists). It should not be possible to import the same data more than once to the same list of records (i.e., duplicate records should not be imported and there should be a check whether this record already exists).
* Users should be able to manually add new airports to the list of airports. It should not be possible to add an airport with an existing three-letter-code.
* Users should be able to manually add a new airline to the list of airlines. It should not be possible to add an airline with an existing two-letter-code.
* Users should be able to manually add a new route to the list of routes.
* Users should be able to update records.
* Users should be able to delete records.

**Feature package 5 (persistent data storage)**

* Data and the additionally calculated information must be stored persistently. Users should be able to access this information when closing and re-opening the application. This requires a data structure and storing mechanism. You will need to define your own data format for storing data in your application. This data format will most likely overlap with the data format of the external data. Also, you need to think about how different lists of data are stored.

When implementing the persistent storage we suggest an iterative approach. You could start by defining a data access layer (DAL) with high-level methods (e.g., getAllRecords(), getAllRecordsInDateRange(<range>), etc.) and then implement the actual persistence in a very simple way to start with (e.g., [text] files). Once you get a bit further into COSC265 you could then easily modify the functionality underneath the DAL to use an SQL database (e.g., SQLite) without having to modify much of the rest of the application.

* Users may wish to separate data across several files.
* If a file is currently open then the application should offer to close that file when the user attempts to open another one.
* Users should be able to persistently store newly added records.

**Feature package 6 (visualizations)**

* + Graphical representation of airports with routes
  + Graphical representation of equipment used on routes
  + Graphical representation of airports per country
  + Graphical representation of airlines per country
  + Map of airports
  + Map of routes
  + Map of routes per equipment
  + Map of routes based on airport

We suggest that you use the Google Maps API or OpenMaps for visualization of spatial information.

**Feature package 7 (optional)**

* Additional reporting and analyses
* Export data into database over a network

Note: For all features you may also implement some filtering mechanisms, e.g., only display, export, save data based on certain parameters (e.g., country, time zone).

1. http://ergoss.net [↑](#footnote-ref-1)
2. https://www.flightradar24.com [↑](#footnote-ref-2)
3. [http://openflights.org/data.html#airport](http://openflights.org/data.html%23airport) [↑](#footnote-ref-3)
4. http://openflights.org/data.html#airline [↑](#footnote-ref-4)
5. http://openflights.org/data.html#route [↑](#footnote-ref-5)
6. [https://openflights.org](https://openflights.org/) [↑](#footnote-ref-6)
7. <https://flightplandatabase.com/planner> [↑](#footnote-ref-7)