# SENG202 – Software Engineering Project Workshop

## 2018

## Project Description – Health Tracking and Analysis App

## Document history

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

* Last update: July 16, 2018

## Introduction

The project is about designing and implementing a health monitoring and analysis app. The purpose of the app is to support the monitoring, analysis and management of physical activity data typically recorded by health and activity monitors and trackers (e.g., Polor Loop, BodiMediaFit, Fitbit Tracker, Samsung Gear Fit, Nike+ FuelBand, Pebble, Technogym, Lark, Zeo, SleepBot). Data from these devices can be analysed using tools such as www.polarpersonaltrainer.com, connect.garmin.com or the Garmin Training Center (more examples can be found here: http://thefitnesstracker.com/top-10-best-fitness-tracker-apps-reviews/).

Throughout the course documents we will refer to the health tracking and analysis app as HTAS. However, you should decide on a catchy name for your own project.

**Requirements and features described in the course documents are intentionally somewhat open-ended. It is your responsibility to define features that you think are feasible and will satisfy potential stakeholders.**

## Initial design considerations

* Your system should be implemented in Java as a desktop application (including 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 must run on the platform and environment provided in the computer labs. This means, deliverables must be compile-able and runnable on the environment provided by the department on the lab machines (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 at a time. However, different users may use it at different times (again, just like Microsoft Office or Firefox) or you may have different types of users.
* You should think about making your final product available on open source platforms, such as GitHub or SourceForge, or submitting it to a programming or design competition.

## Overview of features

We assume that users of your application already own a health and activity monitoring device. Health and activity monitoring devices usually monitor motion through an accelerometer, steps (distinct patterns of walking or running), heart rate monitors, etc. Users of the application would upload activity data into your application for analysis.

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

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

* The main GUI of the application is split in several parts (e.g., panels) and has a menu bar. Users will also want to be able to manually enter data into the app so think about UI features that could enable this (see also feature package 4).
* The HTAS should allow users to upload an activity data file with activity data (obtained from some “smart” device) for analysis. Products such as Polar Loop or Garmin Forerunner 70 allow users to export activity data. For simplicity reasons we will assume a data format that the app needs to be able to process (see seng202\_2018\_example\_data.csv).

Activity data:

A row (or “data point”) in the activity data file consists of (at least) the following information (in that order):

* + Date (dd/mm/yyyy)
  + Time (hh:mm:ss; 24h)
  + Heart rate (bpm)
  + Latitude
  + Longitude
  + Elevation (m)

Rows/data points in the activity data file are grouped in activity sets (“activity record”). Each activity record has a title/name.

A record in the activity data file starts with “#start”, followed by the title/name of the activity.

The provided activity data file is just an example to get started. More data can be added (e.g., by adding GPS data from other .gpx files or heart rate data from other .tcx files [this may require converting these files to plain text/csv; converters are available online]). Other “tags” or types of data may be added to the activity data file format if desired.

The example data file contains real GPS and heart rate data. Based on how GPS devices record data, the difference between two time points varies, i.e., the time intervals between data points vary and have irregular patterns. Thus, when analysing and visualizing data, some interpolation or approximation may be required. For example, if data for every 10 seconds should be visualised for five data points (e.g., 23:42:28, 23:43:05, 23:43:15, 23:43:34, 23:43:46), then some mechanism must be implemented to “approximate” or interpolate the missing data so that it can be analysed/visualised properly.

* For each record (or “data point”) the following items should be calculated based on information provided in the activity data file:
  + Distance (m): Distance can be calculated based on the difference of latitude and longitude of two data points (for example, the distance for a point “*pn”* would be calculated based on the difference of latitude and longitude of point “*pn*” and “*pn*-1*”*).
  + Speed (km/h): Speed can be calculated based on distance and time difference between two data points (for example, the speed for a point “*pn*” would be the average speed to move the distance between “*pn*-1*”* and “*pn*”).

Speed and distance for the first “data point” (“*p*0”) in an activity set are 0.

* The application can show the content of data file in a table (“Raw data viewer”, including distance and speed).
* Note that the datasets 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 a plain text editor, such as Wordpad or Sublime Text, to view and search the data. Furthermore, it may be difficult to show the entire content of a file at once. The “raw data viewer” should allow users to filter data. The filtering should be implemented as drop-down menu or a similarly usable mechanism.

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

* 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.
* User data can be stored in one file. You need to decide on your own data structures and how the data should be stored (e.g., by serialization). User data consists of the following:
  + User profile data (e.g., name, day of birth, height, weight, results of data analyses, etc.)
  + Health and activity data from uploaded activity data file(s).
* User data files should be loaded and displayed properly. If a user data file is currently open, it should be closed.

**Feature package 3 (basic analyses and information)**

* Basic data analysis and view (e.g., BMI)
* Depending on the value of health parameters, warnings should be shown:
  + Tachycardia: Heart rate is over normal range for a particular age group. The age of the user could be determined from the profile data.
  + Bradycardia: Heart rate is lower than usual for a person of a certain age group (50bpm for adults).
  + Cardiovascular mortality: Adults with a resting heart rate of greater than 83bpm are prone to cardiac diseases.

Warnings would depend on the activity. For example, if the name/title of the activity indicates heavy physical activities (e.g., running), then these warnings would not make sense. However, if the name/title of an activity indicates light physical activities (e.g., a walk), then warnings are more reasonable.

* Integration of web search feature to search for health-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 for the same user. 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 add new activities manually.
* Users should be able to update 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 of your application 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.

**Feature package 6 (visualizations)**

* Graphical representation of distance over time
* Graphical representation of heart rate over time
* Graphical representation of calories burned
* Graphical representation of stress level over time
* Map of routes based on activities
* Calendar views of activities

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

**Feature package 7 (optional)**

* Export data into a database over a network
* Additional reporting and analyses
* Activity recommender
* Import activity data files of different formats
* Export user data files to different formats

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