CSE 350

Team #16

Software Requirements Specification

Document

Ayush Thengne, Muhammad Seyal, Ivan Lamb, Ashleigh Gray, Stephanie Mcdonald, Carsten Kurtz

Version: 2.0 Date: 11/27/2022

Table of Contents

**1. Introduction**

*1.1 Purpose*

*1.2 Scope*

*1.3 Definitions, Acronyms, and Abbreviations*

*1.4 References*

*1.5 Overview*

**2. The Overall Description**

*2.1 Product Perspective*

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communications Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

*2.2 Product Functions*

*2.3 User Characteristics*

*2.4 Constraints*

*2.5 Assumptions and Dependencies*

*2.6 Apportioning of Requirements*

**3. Specific Requirements**

*3.1 External interfaces*

*3.2 Functions*

*3.3 Performance Requirements*

*3.4 Logical Database Requirements*

*3.5 Design Constraints*

3.5.1 Standards Compliance

*3.6 Software System Attributes*

3.6.1 Reliability

3.6.2 Availability

3.6.3 Security

3.6.4 Maintainability

3.6.5 Portability

*3.7 Specific Requirement Analysis*

*3.8 State Transition Diagram*

1. Introduction

1.1.Purpose

The purpose of this document is to outline the process and plan used to meet the requirements given in the term project. This document is intended to be used both as an outline of what the product is going to contain, but also as a guide to how the product should function and be built

1.2 Scope

A program to import the data collected from a sensor and provide the means to do analysis on it.

1.3 Glossary

* + - OO – Object Oriented

1.4 References

* + - IEEE 830 – 1998 sample
    - IEEE SRS Template
    - Term Project Overview

1.5 Overview

This document uses a standard IEEE 830 - 1998 format with a few changes. Sections 4-6 have been omitted, 3.8 is the state transition diagram, and section 3 is in OO format. Important sections are listed below

* + Section 2: General description of the project, contains information on constraints, user requirements, software and hardware requirements.
  + Section 3: gives a more Indepth look at functional as well as nonfunctional requirements within the project. This is an OO design written in java which will contain all information of classes and relationship

2. The Overall Description

The Term Project will allow the user to take data collected from a sensor and import it into the program to be analyzed. It will take the csv files provided by the monitor itself and import the statistics into the program so that they may be visualized using graphs. The application will provide several

2.1 Product Perspective

The product will be a python executable that on startup will allow the user to select the csv files to be imported from. It will give them a view of several different graphs that display the data from that csv file. The user will then be able to scroll through the data, zoom in and out on certain points, and use the program to perform functions that allow them to see the data in different ways. The application should be able to run on any current windows machine and allow the user to view them without a network connection or any other external help.

2.1.1 System Interfaces

The application only needs to interface with the csv file on the host device, so it only needs read access on the host device

2.1.2 User Interfaces

The application will interface with the user with a graphical user interface that allows them to interact with their data as well as see trends across it.

2.1.3 Hardware Interfaces

There are no hardware requirements necessary for the Term Project Applet outside of basic input/output such as keyboard and mouse, as well as a display on the user end. The database will require no additional add-on downloads or hardware interfaces.

2.1.4 Software Interfaces

At the software end we have no concrete interfaces for the term project outline that would need to be used in order to function to customers specs. Optimally we recommend an internet connection as the Term Project is written in Python and requires an internet connection to fetch updated data from an updated repository to dump into the local CSV database.

- Python Programming Language

- CSV for database

2.1.5 Communications Interfaces

No communications interfaces are required because the data is in a static database.

2.1.6 Memory Constraints

For the application to run on any windows 10 or later machine the application should take no more memory than the windows minimum requirements of 1 gigabyte

2.1.7 Site Adaptation Requirements

2.2 Product Functions

The startup of the program calls all of the components needed to build the outline of the application your such as your main window. The program will then function in an event-based fashion where the user clicking on interface buttons allows them to manipulate their data for them to draw conclusions.

Functional Requirements

1. Initialize the application and put it into a functioning state for the user

1.1 Get the file path of the files we are looking at with our program

1.2 Generate the gui we are working with including time graph, the buttons to control the data, and an area for our graphs to go

1.3 populate graphs using the data from our csv files

2. Manipulate the data for the user to be able to change the way that you can view it

2.1 The ability to zoom in and out of the graph

2.2 Functionality to pan the data left and right to scroll across it

2.3 functionality to the timetable to be able to quickly pan across the data using it  
 2.4 change the way that the certain labels on the data are displayed such as the time

2.5 sync data across several time zones so they are viewable the same way

3. change the data itself

3.1 a functionality to change the csv file that is loaded into the graph in order to view different data

3.2 an aggregation function that takes the average of the data over a period of time and collapses it into a single data point so it is easier for the user to see trends that may be there

3.3 a summarization function that gives various interesting things about the graph such as its running average, its extrema the midpoint and standard deviation

2.3 User Characteristics

All users of the application are treated the same in that they have the same ability to import data and manipulate it in any way they see fit

2.4 Constraints

*There are no further constraints indicated for users of the system.*

2.5 Assumptions and Dependencies

The user has a version of python compatible with the application and an appropriately formatted csv file accordingly.

3. Specific Requirements

3.0.1) The system shall allow the user to visualize data from a multimodal monitor

3.0.2) The system shall have graphs to display the data and a timetable to interact with it

3.0.3) The application will allow the user to manipulate the data in order to see different views of it

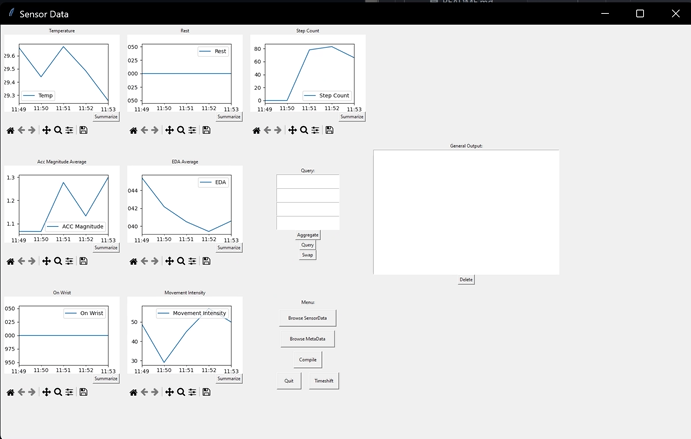
3.0.4) The application will allow you aggregate and summarize the data

3.0.5) The system shall allow you to switch between your time zone and utc

3.0.6) The program will automatically sync across time zones

3.1 External Interface Requirements

This section outlines the graphical user interface for the Term Project.



User interface features the time box along the top displaying the current range displayed along with its position within the whole. There are several graphs below it that show the data. Temp is the temperature at that time ACC magnitude is the user's acceleration, on wrist is if they had the sensor on, step count is how many steps they averaged in a minute and rest is whether the user was at rest or not. These graphs give you an idea of what the user was doing at all times

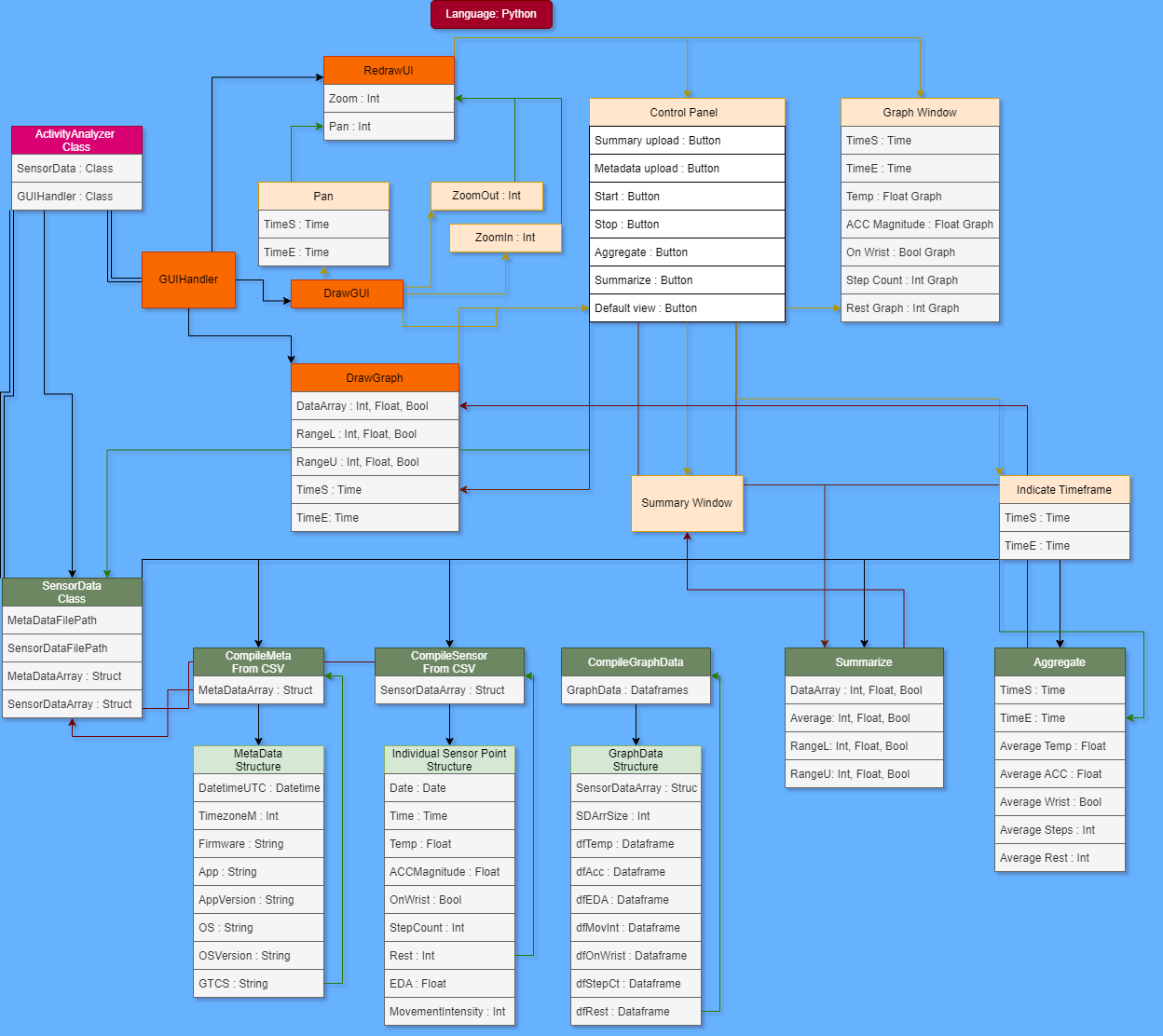
There are several buttons along the side of the window allowing the user to perform functions on the data such as designating where in the data they want to look and what file they are pulling data from.

The aggregate allows you to select from a drop-down menu how you want the data to be clustered to allow you to see trends from average values.

The summarize function gives you an output in another window with various interesting data points.

Default view displays the data in the way that the initial values on startup were to bring you back to the baseline.

3.2 Classes/Objects



3.2.1.1 ActivityAnalyzer Class

This class will handle initializing the GUIHandler class, the SensorData class, and the Files Structure struct.

3.2.1.1.1 MetaData()

This method compiles all of the data from each row of the metadata CSV file and creates a temporary structure to add to an array containing all of the metadata.

3.2.1.1.1.1 MetaDataArray Struct

This structure is temporary and contains the metadata for a single row of the CSV file to initialize itself.

3.2.1.1.1.1.1DatetimeUTC Datetime

This element describes the day and time the metadata was recorded.

3.2.1.1.1.1.2 TimezoneM Int

This stores the Time zone as an int.

3.2.1.1.1.1.3 Firmware String

This stores the client end firmware as a string.

3.2.1.1.1.1.4 App String

This stores the client-side app as a string.

3.2.1.1.1.1.5 AppVersion String

This stores the client-side application version as a string

3.2.1.1.1.1.6 OS String

This stores the client's OS as a string.

3.2.1.1.1.1.7 OSVersion String

This stores the clients OS Version as a string.

3.2.1.1.1.1.8 GTCS String

This stores the clients GTCS as a string.

3.2.1.1.2 SensorData:

This stores Both of the SensorData and MetaData Array sizes of both arrays and can switch between each other and allows a recursive Search parsing date and time on it.

3.2.1.1.2.1 SDarr:

This stores the SensorData Array

3.2.1.1.2.2 MDarr:

This stores the MetaData Array

3.2.1.1.2.3 SDSize:

This stores the SensorData Array size

3.2.1.1.2.4 MDSize:

This stores the MetaData Array Size

3.2.1.1.2.5 SwitcherSum:

This Switches between the sum of MetaData and/or Sensordata based on the user input from bool to int

3.2.1.1.2.5.1 Bool2int:

This switches the Boolean to an Int when needed

3.2.1.1.3 RecursiveSearch:

This Searches and Parses through the SensorData and Metadata for Date and Time

3.2.1.1.3.1 ParseDateTime:

This stores the date and time found in the recursive search

3.2.1.1.3 CompileSensor()

This method compiles all of the data from each row of the summary CSV file and creates a temporary structure to add to an array containing all of the summary data.

3.2.1.1.3.1 CompileMeta Struct

This structure is temporary and contains the metadata for a single row of the CSV file

3.2.1.1.3.1.1 Date Date

This stores the date of the sensor data as a Date object.

3.2.1.1.3.1.2 Time Time

This stores the time of the sensor data as a Time object

3.2.1.1.3.1.3 Temp Float

This stores the temp of the sensor as a float.

3.2.1.1.3.1.4 ACCMagnitude Float

This stores the ACC Magnitude from the sensor data as a float.

3.2.1.1.3.1.5 OnWrist Bool

This stores whether the sensor is on the wrist or not. True if it is on the wrist, false otherwise.

3.2.1.1.3.1.6 StepCount Int

This stores how many steps were taken as an integer.

3.2.1.1.3.1.7 Rest Int

This stores how long the user is at rest and is taken as an integer.

3.2.1.1.3.1.8 EDA Float

This stores the Temp and ACCMagnitude as a float inside of the sensor data array struct

3.2.1.1.3.1.9 MovementIntensity Int

This will be timed with the ACCMagnitude and time to estimate the movement intensity as an INT

3.2.1.1.3.2 CompileGraphData:

This stores and compiles the GraphData onto the Graph when all is compiled from the Meta Struct.

3.2.1.2.7 Summarize()

This class will take a data point and summarize it with statistics such as the average and the range.

3.2.1.2.7.1 DataSet

This variable will hold an array of the data to be summarized.

3.2.1.2.7.2 Average

This will store the average value of the data point being analyzed.

3.2.1.2.7.3 RangeL

This will store the RangeL as an Int/Float/Boolean.

3.2.1.2.7.4 RangeU

This will store the RangeU as an Int/Float/Boolean.

3.2.1.2.8 Aggregate()

This method will find the averages of all recorded sensor data over a specified period of time.

3.2.1.2.8.3 AverageTemp

This will store the average temperature of a given range as a float.

3.2.1.2.8.4 AverageACC

This will store the average ACC of a given range as a float.

3.2.1.2.8.5 AverageWrist

This will store the average Boolean value of a given range for whether or not the sensor is on the wrist.

3.2.1.2.8.6 AverageSteps

This will store the average steps as an integer value of the given range.

3.2.1.2.8.7 AverageRest

This will store the average rest period as an integer for the given range.

3.2.1.3 GUIHandler Class

This class will handle the initialization of the GUI, and the graph visualization.

3.2.1.3.1 DrawGUI()

This method draws the GUI and its elements.

3.2.1.3.2 RedrawUI()

This method is called when a new timeframe is specified by the Pan window. Additionally, this method is called when a user changes the zoom on a graph.

3.2.1.3.3 DrawGraph()

This method will draw a graph visualization for each of the data points.

3.2.1.3.3.1 DataArray

This variable will hold an array of the data to be summarized.

3.2.1.3.3.2 RangeL

This will store the RangeL as an Int/Float/Boolean.

3.2.1.3.3.3 RangeU

This will store the RangeU as an Int/Float/Boolean.

3.2.1.3.3.4 TimeS

This will store the start time of the graph

3.2.1.3.3.5 TimeE

This will store the end time of the graph

3.2.1.4 User Interface Class:

This will handle the User Interface of the Graph that is Compiled from CompileGraph

3.2.1.4.1 Initializes Plots:

This will initialize plots for the graph (data points: temp, ACCMag, Onwrist, StepCount, Rest) based on aggregate and the previous call throughs.

3.2.1.4.2 Summarize for each plot:

This will store the summarize temps for each of the plots being ACC\_Mag, on\_Wrist and Step\_count and Rest as well as EDA and Movement Summarize

add window for compiling new data with two fields

3.2.1.4.3 addNewData:

add window for compiling new data with two fields

3.2.1.4.4 SwitchTimeSeries:

Changes the time series value for all of the data between Unix and current time zone

3.2.1.4.5 AggregateData:

Data that is searched from inside of the DummyData CSV for parsing into the graph as stock graph data. This data is raw data that is gathered and expressed in a summary form for analysis.

3.2.1.4.6 QueryData:

Adds a window to query based on the timeframe

3.2.1.4.7 SwitchGraphData:

switches between query graphs and full graphs

3.2.1.4.8 Summarize:

Gives the sum highest and lowest value of the inputted data

3.2.1.4.9 CallBack:

This will set the axis to clear and plot the axis with time along the X axis and sets the title as well as draws the final graph calling back unto the initializer on a main loop for efficiency.

3.2.1.5.0 Regraph:

takes the data from the imported CSV using the browsefiles.Py and parses the data onto the graph as a query graph

3.2.1.5.1 BrowseFileSD:

takes the data imported from the CSV and adds It to SensorData

3.2.1.5.2 BrowseFileMD:

takes the data imported from the CSV and adds it to MetaData

3.3 Performance Requirements

The software will have a user interface which will allow the user to load in the sensor data. This will be categorized by time, temperature, ACC magnitude, time on wrist, step count, and time resting. The user will also be able to explore summary and aggregate data with the ability to switch between local time and UTC time. The user will be able to choose which csv file they want to visualize which will be imported from the sensor using the software.

3.4 Logical Database Requirements

Four csv files will be used to store the multiple time series (summary.csv), the time zone information (meta.csv), ACC data (acc.csv), and the EDA data (eda.csv). Data will be added to these csv files from the sensor.

3.5 Design Constraints

Any hardware limitations would come from the sensor’s ability to efficiently collect data and the sensor’s ability to connect to the software. The software will not be hindered by hardware limitations except for storage limitations regarding the csv files, but that is unlikely to be a factor as the csv files will most likely not be big enough for it to be seriously constrained by storage limitations.

3.6 Software System Attributes

The software will add data to the csv files. Making sure that the sensor can load the data in important to the functioning of the software. The software will need to be maintained to make sure that the sensor can still connect to the user interface and csv files to input data. The software will be written in Python.

3.7 Specific Requirements Analysis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Priority | 1 | 2 | 3 | 4 | 5 |
| Requirements | \* |  |  |  |  |  |
| User Interface  For sensor Data |  | \* |  |  |  |  |
| Sync across Time Series = |  |  | \* |  |  |  |
| Switch between UTC and Local |  |  |  |  |  | \* |
| Zoom in/ zoom out |  |  |  | \* |  |  |
| Pan |  |  |  | \* |  |  |
| Aggregate |  |  |  |  | \* |  |
| Summarize |  |  |  |  | \* |  |

3.8 State Transition Diagram