SYSTEM DESIGN DOCUMENT

for

FAULT TOLERANT SHIP-BOARD DATA LOGGING AND PROCESSING SYSTEM

Customer:	Nils Haëntjens School of Marine Science
Prepared by:	**Team Aqua:** Jacob Hall Sam Segee Chi Anh Nguyen
Guided by:	Prof. Terry S. Yoo

Fall 2018, COS 397
November 16, 2018

FAULT TOLERANT SHIP-BOARD DATA LOGGING AND PROCESSING SYSTEM

System Design Document

Table of Contents

	Page
1. Introduction	**2**
1.1. **Purpose of This Document**	**2**
1.2. **References**	**2**
2. System Architecture	**3**
2.1. **Architectural Design**	**4**
2.2. **Decomposition Description**	**4**
3. Persistent Data Design	**9**
3.1. **File Descriptions**	**9**
4. Requirements Matrix	**10**
Appendix A – Agreement Between Customer and Contractor	**11**
Appendix B – Peer Review Sign-off	**12**
Appendix C – Document Contributions	**13**
Appendix D – Version Control	**14**

1. Introduction

1.1 Purpose of This Document

The purpose of this document is to provide the reader with a description of the way the system will be built. This will give the programmers a reference while building the system and gives the client an idea of what the final system will be like.

The results of the system design process are recorded in the System Design Document (SDD). This document describes the system at the architecture level, including subsystems and their services, hardware mapping, data management, access control, global software control structure, and boundary conditions. The SDD defines a system that implements all the requirements from the System Requirements Specification (SRS) and serves as a guide while implementing the functional requirements.

1.2 References

“The University of Maine In-Situ Sound & Color Lab.” *The University of Maine In-Situ Sound & Color Lab*, 18 Mar. 2014, misclab.umeoce.maine.edu/.

-The official website for the MISC Lab located at the University of Maine.

Sommerville, Ian. *Software Engineering*. Pearson Education South Asia Pte Ltd, 2016.

Many aspect of this document are taken from this document. This is document also talk about general software engineering processes.

Fowler, Martin, and Kendall Scott. *UML Distilled: a Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, 1999.

“System Design Document.” *Control Structures*, www.cs.fsu.edu/~lacher/courses/COP3331/sdd.html

2. System Architecture

2.1 Architectural Design

The raw data output by the sensors is read by the primary computer: sensors with serial connection are connected through a serial to USB converter while sensors with analog connection are connected through a USB analog to digital converter. The primary computer writes the data coming from the sensors to raw binary files. The raw binary data is converted to decimal values and plotted in real-time on the primary computer. The conversion process requires, a sensor configuration file specific to each instrument and latest calibration date. As soon as a frame of data is received by the computer it is time-stamped, this should be the first step to achieve maximum accuracy in timestamping the data.

The primary computer can read and set the state (filtered or total) of the control switch via USB as well as read the flow rate from the two flow meters that are connected to the control switch. The primary computer timestamps and writes to a file every change of state of the control switch and the flow rate measured by each flowmeter of the control switch. The control switch can be controlled from the primary computer, there should be 3 options: automatic switching (given amount of time filtered every hour), force total, and force filtered.

The raw binary files are sent to the secondary computer from the primary computer where the files are stored internally and to the backup system. The system is redundant with the files being stored on both computers and an external backup system (e.g. USB Hard Drive). The sensor status (e.g. number of packets received in past hour, number of incomplete packet in last hour, sensor connected) will also be sent by the primary computer to the secondary computer. This data will not be stored into a file but rather passed to the web application.

The web application is a read only website and cannot control any of the operations of the system. The website will display: real-time graphs (time series and values as function of the channels) of decimal values for the past hours, and processed data from the entire cruise. Additionally, there will be a status screen which shows the status of the control switch and each of the sensor.

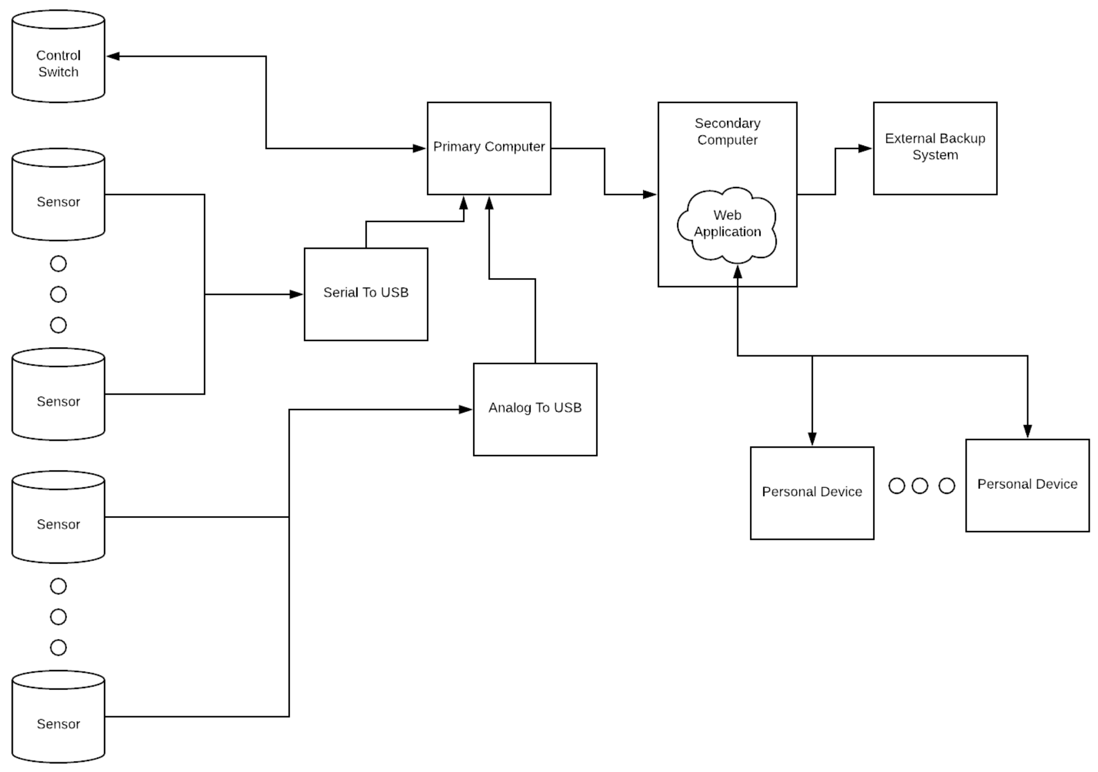


Figure 1. System Overview

2.2 Decomposition Description

Multi-paradigm programming utilizes both functional programming and object-oriented programming elements to support this project. This project will be based mostly on functional style to evaluate mathematical data, while having object-oriented elements to solve requirements stated in the System Requirements Specification document.

<ins>Data Collection, Formatting, and Backup</ins>

The *Sensor* class collects incoming data from the sensors and passes incoming data to the File class using function *write()*. For the file to be readable by the graphing software the file is passed to *Process* class which has all of the functions that allow for the data to be readable and processed.

Backup class send copies of the data files using *sendBackup()* from the primary computer to the secondary computer. The secondary computer will receive the sent data with function *receiveBackup()*.

The system will have control over the control switch which controls if the water is filtered or not. The *Sensor* class allows for the system to set the state of the control switch to filtered, total, or automatic. Automatic means the control switch will be set on a set schedule. The control switch also sends data of the current state of the switch and flow rate of the water over the same connection. This data will not be collected by the *FlowControl* class but will come in through the Sensor class.

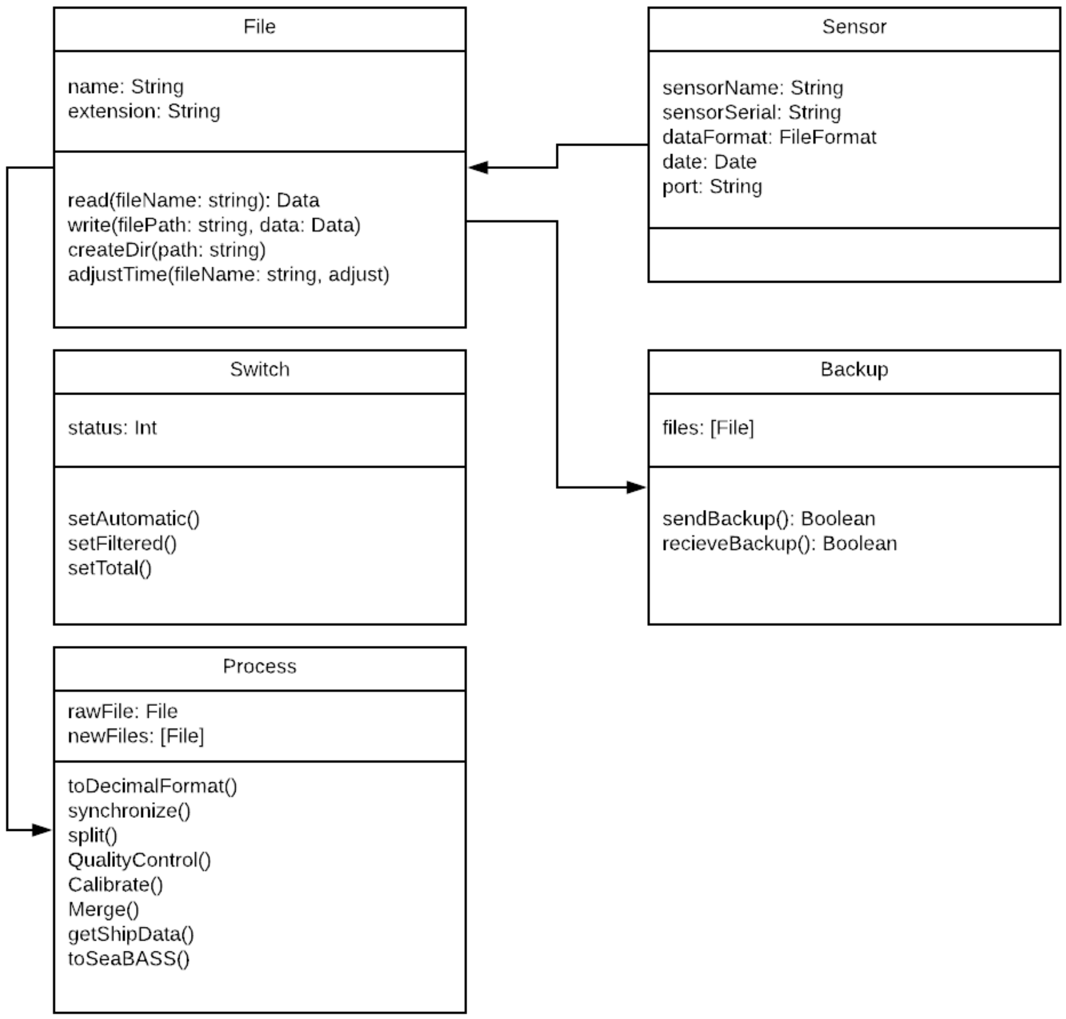


Figure 2. Object and Function Model

Web Application: System Status and Data Visualization

The web application will receive data coming in from the sensors and graph the data in real time. Additionally, the status of each of the sensors currently running will be displayed.

The web application will follow the model-view-controller paradigm meaning the data(model), displaying(view), and controlling of the view(controller) will be extracted into different components. *Figure 3.* shows how the components of the system will interact with each other. At the top is the index of the website which loads in the main view. The main view will contain the template and styling of the website. The graph and sensor views will be loaded into the main view where they'll be displayed.

There are two graph visualizations in the system for both the processed graph and real time graph. The first graph will be one specific sensor over time. If the sensor has multiple channels this graph will plot one channel over time. The sensor and the chanel that is currently plotted will be selectable in the web interface. If the sensor has multiple channels a second graph will be available. By selecting a point in the first graph will change to plot all of the channels for the time selected.

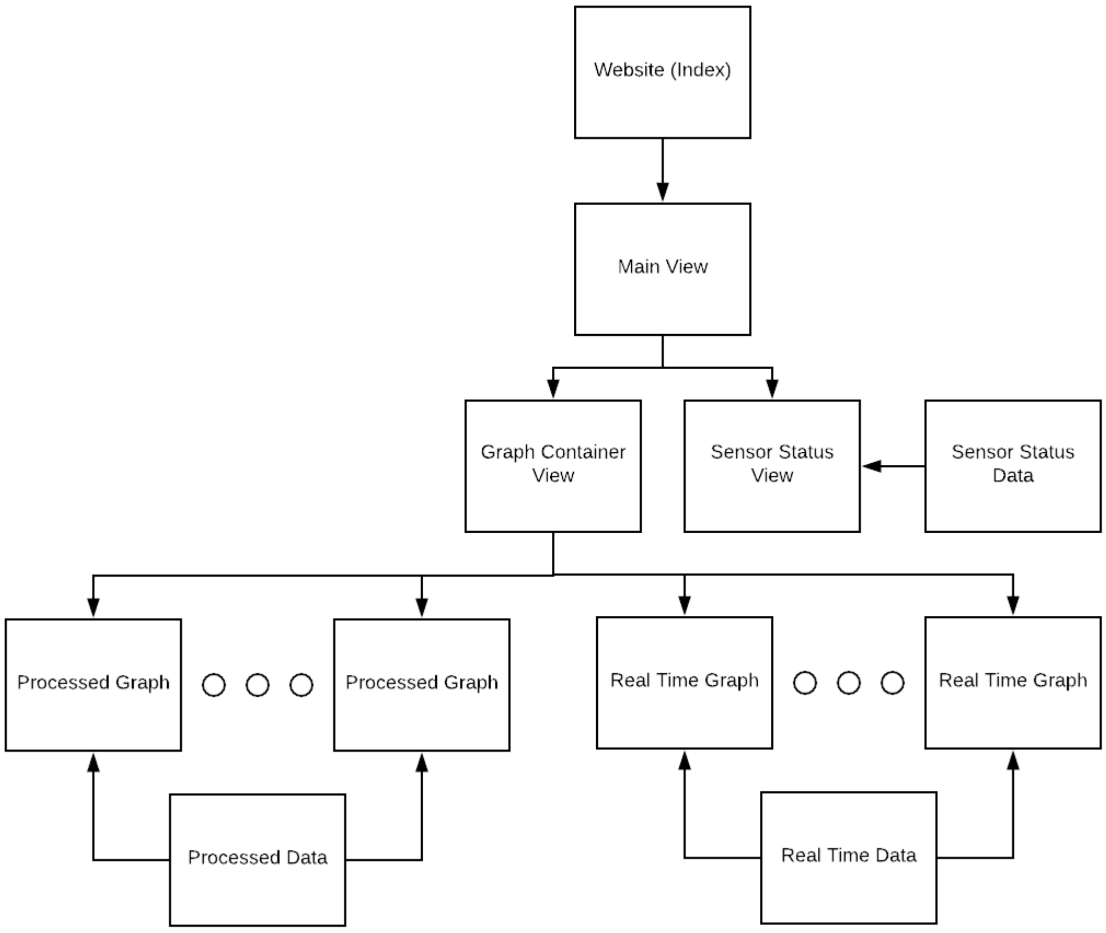


Figure 3. Website Structure

Data Processing

For data coming from the sensor to be meaningful the data must be processed before it can be graphed and exported. When the raw data comes into the system the data is converted to decimal values so it can be plotted on a graph in real time. This decimal values are saved and backed up. Every few hours (parameter to adjust) all the decimal files from the sensors goes through a more complex processing phase. Below are each of the steps during the processing phase.

Raw Data: The sensors send data in binary format. This data is stored so it can be reprocessed if changes are made to the processing algorithms.

Decimal Values: Collected raw data is converted into usable data that can be easily graphed and read by the user. The process of converting raw data into easily readable data is different for each sensor and is defined in the sensor's configuration file.

Synchronize: Each sensor has a different delay when storing and processing data. The user will be able to view a graph of data collected by different sensors and line them up based on the shape of each graph. The difference will then be stored and applied to all data collected by the sensor. This user interface process is generally only required once per voyage.

Split: separates the data between filtered seawater data and total water data. When the system toggles between filtered and unfiltered water, there's a brief adjustment time in which collected data is unusable. The user can use the split tool to separate and delete the unusable data caused by the switch between filtered and unfiltered water.

Bin: data is binned into 1 minute averages, the standard deviation, and number of samples in each bin is kept. This step is required to reduce noise in the dataset.

Quality Control: Often times data should be removed due to interference such as bubbles going through the system. Due to the nature of the interferences it is hard for an algorithm to remove this bad data so the user will select bad data through a graphical user interface(GUI). The system will skip this step and continue to the next step until a user selects any bad data. After the bad data is selected, the user input is save and this function can be run again without asking for User Input again.

Calibration and Corrections: This phase consists of five major steps. The first step is to *interpolate* filter sea water data on the total sea water data. The second step is to *subtract* the *interpolated data* from the total sea water data. The next three steps are dependent upon which sensor data is being processed. Step three takes the data from step two and *adjusts* the values based on the sensors calibration parameters. Step four makes *corrections* to the data, such as correction for temperature dependencies. Finally, step five *calculates the product* for the sensors data and sends that data to be merged.

Merge: Some information such as location, salinity, and temperature are collected by the ship instead of the computer system. After all the usable data has been separated from unusable data, the time stamp on the data is then used to add the missing data to the recorded data to produce a comprehensive SeaBASS file (CSV with a specific header) and visualization on the web application.

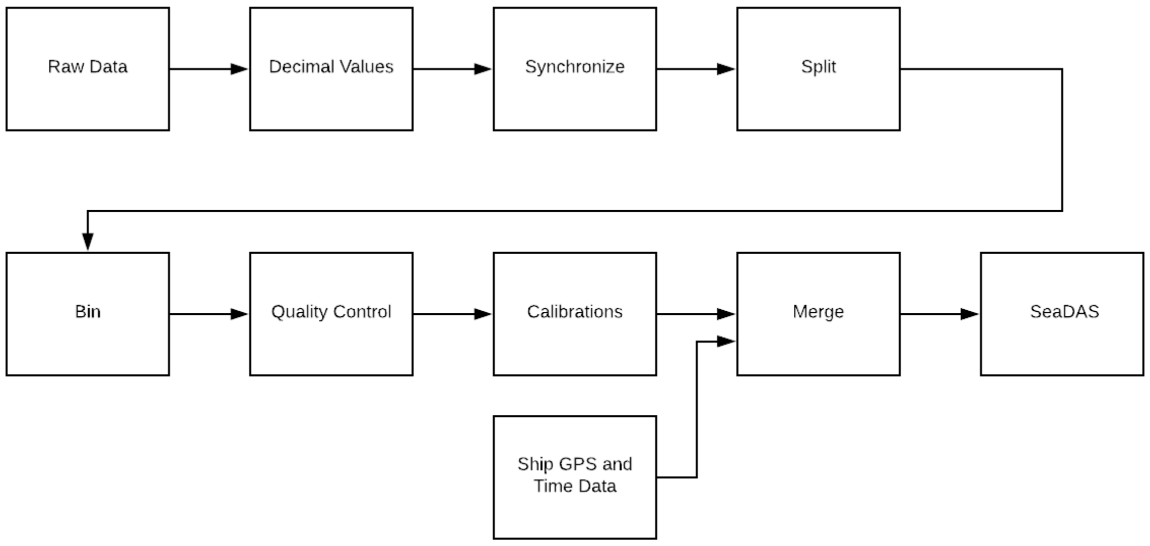


Figure 4. Processing diagram

System Failure

There are five major failures to consider for the defined system. How the system will handle each of the five failures are described below.

Primary Computer Hardware Failure: In the event where the primary computer fails a new computer will replace the primary computer. The new primary computer will need to have all of the software installed to be fully operational.

Secondary Computer Hardware Failure: In the event where the secondary computer fails a new computer will replace the secondary computer. The data from the backup hard drive will be copied over to the new secondary computer's hard drive to mitigate the risk of data loss.

Network is Inaccessible: If the network is inaccessible, a notification will be displayed on both the primary and secondary computer. Collected data will be saved on the primary computer until the network connection is restored at which point the saved data will be sent to the secondary computer.

Power Failure: If the instruments are running on battery, display a notification of the event then keep primary computer logging data for 5 more minutes. After 5 minutes, if the power is still down, display the option to disconnect the instruments and stop logging data, and minimize computer power consumption. After 30 minutes, if the power is still down, turn off the application. When the power gets back, we resume all operations.

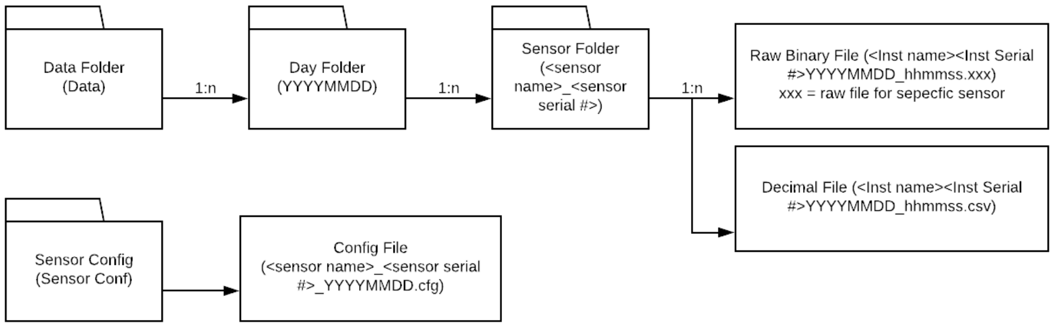
Backup Hard Drive Failure: Data will be stored on both the secondary computer and backup hard drive. If the backup hard drive fail the drive will be replaced and all the data on the secondary computer will be copied over to the new backup hard drive.

3. Persistent Data Design

3.1 File Descriptions

The files will all be stored in a different folder per day each folder name will be based on the date the folder was used. For example, all data from November 16, 2018 will be stored in the folder 20181116. Two new files will be created per sensor every hour with one file containing the processed data from the past hour with time stamps and another containing the raw data. Each file will be named using the date, hour, and type of data stored. Raw data will be stored as a binary file while processed data is stored as a text file. For example, raw data from 3:00 pm on November 16, 2018 would be stored as *201811161500.bin* and the processed data will be stored as *201811161500.csv*. Every new file is also stored on a backup system as soon as it’s created.

A separate folder will also contain all the config files for each sensor. Each config file will contain information about the sensor used to convert raw data during the processing and decoding phases.



Data Folder (Data)
1:n
Day Folder (YYYYMMDD)
1:n
Sensor Folder (<sensor name>_<sensor serial #>)
1:n
Raw Binary File (<Inst name><Inst Serial #>YYYYMMDD_hhmmss.xxx) xxx = raw file for sepecfic sensor
Decimal File (<Inst name><Inst Serial #>YYYYMMDD_hhmmss.csv)
Sensor Config (Sensor Conf)
Config File (<sensor name>_<sensor serial #>_YYYYMMDD.cfg)

Figure 5. File Structure

4. Requirements Matrix

Requirement	Components that satisfy the requirement
Record raw data	**Classes:** Sensor, File **Function:** read()
Data visualization	**Class:** WebApp **Function:** none
System clock synchronization	**Class:** File **Function:** adjustTime()
Control the switching system	**Class:** FlowControl **Function:** setAutomatic(), setFiltered(), setTotal()
Automatic Data Backup	**Class:** Backup **Function:** sendBackup()

Table 1. Requirement Matrix

Appendix A – Agreement Between Customer and Contractor

This document is an agreement between the Customer (Nils Haëntjens) and Contractor (Team Aqua) stating the requirements that will be included in the system. For the system to accepted all functional requirements must be completed and pass the test documented. By signing this both the Customer and Contractor agree on the work that must be completed and handed over to the Customer (Nils Haëntjens) before the deadline of December 19, 2018.

If changes of the system or requirements are requested from either the Customer or Contractor this document will be updated stating the changes requested and the reasoning behind the changes. For the requested changes to go into effect the document will need to be signed by the entirety of the team and the customer. From that point on the most recent System Requirements Specification will be the document referred to in future documentation.

Team Signatures:

Jacob Hall Jacob Hall Date: Nov. 16, 2018

Samuel Segee [illegible] Date: Nov. 16, 2018

Chi Anh Nguyen [illegible] Date: Nov. 16, 2018

Customer Signatures:

Nils Haëntjens [illegible] Date: Nov 16, 2018

Customer Comments:

Appendix B – Team Review Sign-off

By signing my name below, I certify that I have read, reviewed, and agreed to the information and requirements stated in this document.

Contractor Signatures:

Jacob Hall Jacob Hall Date: Nov 16, 2018

Comments:

Samuel Segee [illegible] Date: Nov 16, 2018

Comments:

Chi Anh Nguyen [illegible] Date: Nov 16, 2018

Comments:

Customer Signatures:

Nils Haëntjens [illegible] Date: Nov 16, 2018

Customer Comments:

Appendix C – Document Contributions

Sam Segee:

Sam wrote the Raw Data, Decimal Values, Synchronize, Split, and Merge sections of Data Processing. He also wrote section 3.1 and part of the requirements matrix. Overall, Sam contributed 25% of the final document.

Jacob Hall:

Jacob created all of the diagrams in this document. In addition he wrote the Architectural Design introduction, Web Application: System Status and Data Visualization, Data Processing introduction, Quality Control, Calibration and Corrections, System Failure introduction, Primary Computer Hardware Failure, Secondary Computer Hardware Failure, Network is Inaccessible, and Backup Hard Drive Failure sections. Overall, Jacob contributed 50% of the final document.

Chi A. Nguyen:

Chi is responsible for a part of Data Collection, Formatting, and Backup in Decomposition Description section, the creation of the Requirements Matrix, the Power Failure in System Failure section. Overall, Chi contributed 25% of the final document.

Appendix D – Version Control

Version	**Date**
Version 1	Sent November 10, 2018
Version 2	Sent November 14, 2018
Version 3	Sent November 15, 2018