Narrative for Software Design and Engineering:

Background and Purpose:

The final iteration of this project from CS-350 was completed in the middle of October 2020. For the class I had created a single python script that completed the requirements of the rubric by breaking most of the calls down into their own defined methods to be called as needed with in the script. Only one function was not removed from the main section of code into its own method, primarily due to limitations of time, the minor increase of complexity from the limitations of the python2 version of the code was built upon instead of the more familiar python3. The code itself uses a light sensor to test the brightness of an environment, if the brightness is high enough to decrease the resistance, as calculated by the program, it would then activate the measurement of a temperature and humidity sensor. Once the data was collected, the program converted the temperature measurement from Celsius to Fahrenheit, before storing the object pair in a list and passing it along to be written into a JSON file. It performed this process every 30 minutes as it would again check the brightness with the light sensor. If brightness were not enough to trigger a measurement the program would then count out thirty minutes before trying again. That is the primary function of the program, additionally there was conditions that the temperature and humidity data was checked against that would control external LED lights based on a prescribed nomenclature.

The purpose of selecting this program for inclusion into my portfolio was based on the possibilities that were available to build it beyond the requirement of the course. The program itself serving as the structural skeleton that functioned for its purpose, and by its own rights show cased the ability to refine a project that could have been performed in a large block code with

each instruction included in the main code section and difficult to review and refining it down to code blocks that allows for the reader to understand what is going on in smaller sections of code and then placing those pieces together. To improve upon this program, I wanted to take all of those possibilities that I had thought about for the program develop those out more by refining how the program terminated, cleaning up bits of code that were poorly named outputs and device connections to conform to various helper files from the class. Further, organize the JSON data by breaking the files into a meaningful set of time frame instead of one large file. The overall design improvement is to incorporate a system that will upload the JSON files into a MongoDB and will manage the JSON files that are stored on the system by removing older, successfully uploaded files, storing those that failed to upload for either manual upload at the next time of interaction by a human or automatically by trying to upload those files again once a successful upload occurs. A time measurement has been added to the to the JSON object that will come into play when the data is being reviewed in category three. The main section of code was further cleaned up, by removing the cluttering defined methods into a better organized library system of files for method calls, because of this the temperature and humidity data was refined from a simple paired list to a more established object that can be easily passed to the different parts of the code system.

Results of the Week:

Due to the number of changes for this program the amount of cross over between the two categories of software design and algorithms/data structures are so intertwined that I have trouble viewing one completed without the other part being completed as well. I found this even more true while trying to focus on just the design this week and consistently found myself trying to fill

in small holes, for the sake of completeness, of things I had planned to delve into during algorithms/data structures section. Some of these were unavoidable in order to maintain a functioning program, such as developing system to construct the weather data object to pass the information between the files, to those things that clearly fit into a later section like addressing the lack of default conditions within the LED light controls. "Have I completed enough to clarify my design plan", is the question that I frequently asked myself while trying to decide where one category ends and the next begins. To that end before moving my base version of the program into my python3 development program, I establish a rough set of individual files and quickly placed pseudocode their purpose so to anyone that was to look at the program while, in essence, functioning as intended for CS-350, does not have the completed features or decision tress in place to fully perform as intended for CS-499. Currently, I do not intend to change my outcome-coverage plans, there have been some hiccups and rough issues during this development phase, but the outlook still appears to be achievable.

Initially, my biggest challenge was staying focused on getting the things done this week in the program that I wanted to get completed, and not straying from the plan and working on all of it as I came to it. I did happen to get so far-off course with my plan towards the beginning of the week that I had to revert back to a backup, as I had moved, inserted, or deleted something, or I did not get finished with something that I thought that I had and instead been derailed onto another topic. As the week progressed and headway was made with the code more technical challenges arose. I had started work on the code using virtual box of raspberry pi, while I waited for a new SD card for my physical raspberry pi to arrive. Once I moved the code from the VM to the physical device, it no longer accepted how I had my imports setup for my methods located

in other files. I believe the issue came from changing IDE programs. In my research for a solution, I came across several people having this issue moving from VM to physical. So, on the VM I used something similar to “from path import file”, as my import command and then to call a method used, “file.myMethod()”, to access the method located in the different file. Once I was on the physical device my import changed to, “from path.file import myMethod, myVariable, etc.” After spending some time on reworking my imports and calls it became clear that just because both IDEs are using python does not mean that their interpreters are similar in what they accept. Other issues arose once the frequency of information I was passing between all the files began causing reference errors with the compiler. This was the point that I need to step back and rework how my data is begin handled by the program, moving from, what now seems more convoluted method, of creating a list of the data points, then breaking them into individual values to be passed and reassembled elsewhere. I originally had the list imported like a standard variable, but the issue was that I was only able to access the 0-index location and the index 1 was unavailable to the file. I began thinking how I would pass this information in C++/Java and I thought that I would do away with this list for the weather data and just pass the whole object instead allowing me to perform different methods as well as reference attributes in the object. As my experience with python is limited to the two or three class where compiled short simple scripts in python, nothing that encompassed creating and using objects let alone multiple files in python, it took a little research to see how to phrase what I was wanting to construct for python to understand, after the initial constructor that problem became a non-issue as well as improving readability of the code surrounding the use of the weather data. Currently, as I am in the grey area of what improvements belong to which category, with my MongoDB functionality of the program, that sends the JSON data to the database, was designated as part of the next categories,

but I have made some headway in establishing a base setup for it, that allows the program to connect to the database and request its status, but currently finding a way for the program to read the JSON file and then insert that as a document is my current issue, as at the moment the command json.load and json.loads both return the type of the variable that is used to house the information as a document instead of a list or an array that allows me to use insert_one or insert_many method to the database.