Narrative for Data Structures and Algorithms:

Background and Purpose:

The artifact that I selected to cover this category, as well as the previous category, was the final project from CS-350, Emerging System Architecture and Technology, named Embedded System Device Script. The final iteration of this project for that course 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. With this category completed, the overall program that encompasses category one and two is complete. With the refinement and expansion of the original program covering the first category, and the complex algorithms incorporated into the MongoDB and report control functions to stand for the second category. There were other refinements that occurred during this cycle that could go into either category, such as removing the need for have the directories for the filing system already setup prior to use, as well inclusions of weekly maintenance of the completed upload reports as well as attempting to upload failed reports. Encryption of the database login information for the program occurred during this past week allowing for more secure operation of the database information.

Results of the Week:

As opposed to last week, I feel a bit more upbeat about this week's work, as I don't have that lingering feeling that not all of design was completed due to trying to squeeze in as much as possible to last week. As far as my plans for what I wanted to achieve with this project, I would

have to say that I met my goals. I was able to take a simple python script and expand it to well beyond the intentions of the original project by including new features, creating a more modular code system within the program while improving readability of the main.py file as well as the other resource files as well.

This week the I was able to move forward rather rapidly, until I started dealing with datetime and datetime.datetime modules in python that led to needing to go backwards a day or two to figure out how I broke something using them. After getting the initial bugs sorted out with those with some conversion of date objects into datetime objects with use of the combine() function I was able to turn my attention to encrypting the database login information. I had originally planned to move the information into a nondescript file like I have the encryption key stored as resource1.ext and encrypt that file a few times with different encryption keys as it was decrypted. The issue I ran into so was trying to get python to allow me to reference things that I would be pulling out of the encrypted files without throwing IDE warnings telling me that a reference was made to a NoneAttribute when I set variables to equal None or just plain invalid reference one the file was encrypted. Eventually, I settled for encrypting the data seven times with a single key, for sake of brevity. Given more time, I would like to try and figure out how to multilayer encrypt the file key and allow python to still refence the information that will be plugged into that variable. Such as a list of keys with varying amounts of encryptions with the keys not being placed in sequential order in the list to add some complexity to it.