In this self-assessment, you will find my reflections from entering the Computer Science program at Southern New Hampshire University, progressing through the program's courses, and a brief introduction to the artifacts used for my portfolio projects.

## The Journey and Reflections:

I began my education journey into computer science field in March of 2018. I come into the field with thirteen years of experience in respiratory care, specifically in the Neonatal Intensive Care and Cardiovascular Intensive Care Units. During this time, I was fortunate enough to get to work with developers from Meditech to refine the charting systems that was currently in use, and then with EPIC charting system to construct the modules for the charting system that was brought in later. It was these small interactions that began my drive to investigate getting into a developer type role, initially to move from patient care into refining the charting system used for patient care. From early high school age, I was self-taught computer hardware enthusiast, I enjoyed building computers for myself and friends and tweaking their performance. So, entering the software engineering side of the computer science program was a dive into the deep end to learn and survive or fail and drown test for me.

The three most important skill sets that I feel that I gain from my time in the program would probably be the development of the mindset of computer scientist, specifically from a coding standpoint. Initially, trying to think about the assignments in a way to get the answers needed felt extremely foreign, and illogical to how I was trained to think as a Respiratory Therapist to see things I feared that I had made a mistake in my decision to return to school for this. Second, would probably be to be modular with my development. It took a while for me to understand that and some of my earlier assignments were long singular methods that performed

everything with the code that repeated itself when need to perform the action again, instead of breaking the code into sections with a modular method that could be called to perform an action when needed and moved into another program when a similar method was required. To get to an understanding of what was trying to be conveyed by that notion I ended up watching a few presentations by Sandi Metz around Object-Oriented Design for Ruby (which is a language I have not used, yet) but took her ideas from those presentation about cleaning up code complexity and trying to keep the code "DRY" (Don't Repeat Yourself) and looked at the issues I was having and things started to fall into place about how to be more modular and how to use methods in a more effective manner. The final concept would have to revolve around exploits and security issues, as just because the program runs as intended when the correct information is provided, does not mean the incorrect data does not break it, or that the software can be decoded to the point of revealing sensitive information.

The program itself had moments of ups and downs; some being complicated from life outside of school as well. Some of largest struggles came early in the program and I was still struggling to understand the process. As I recall a few instructors telling me that I was over thinking the problem, which was the result of trying to figure out the big picture from small simple scripts to provide insight into practices. This was the cause of many late nights, as I was not able to grasp how this would fit into a larger program and I could not understand how to go about creating my small programs. In a discussion board post a fellow classmate referred to it as I was, "trying to understand the purpose of a hinge from looking at the overall function of the desk." Things started to finally click, and I could start understanding how the pieces that I had acquired over the previous terms fit together in CS-260, Data Structures and Algorithms. After this class, the projects seem to evolve more and more with more robust requirements that

required a better understanding of techniques and philosophies that were beyond the programing language that was being used for the class. Another take away from the course was the importance of data structures and algorithms, and what they can provide to us when used properly allowing a place to store data and objects in a style to relevant to the requirements of that data and a means of time and resource savings by understanding how different algorithms will affect the system when used in certain situations.

Two other course that helped understand how my code works in a project came from the classes of CS-330, Computer Graphics and Visualization, and CS-360, Mobile Architect and Programming. Both courses had larger projects one constructing a 3D model with correct lighting and reflections, the other a functioning Android mobile app complete with social media integration, SQLite database storage, as well as user-interface design and implementation. Prior to CS-360 I had a course, DAD-220 Introduction to SQL, that I understood yet had trouble how it would function in the larger picture. When it was a required part of the CS-360 program to use SQLite, and I could see its implementation, I began to see how a relational database system works and how use it could be when applied to a program. In CS-330 it too was a larger project that brought other course into focus of how they applied to computer science, such as the advanced math skills and understanding of Physics concepts, Applied Linear Algebra, Discrete Mathematics, and Calculous 1.

In CS-310, Collaboration and Team Project, and CS-405, Secure Coding was another revelation of both workflow and complexity that I had fully understood the scope of the subject. Coming from a healthcare background, teamwork is not a foreign concept to me. How that would apply to the Software Engineering Concentration course of Collaboration and Team Project I was not exactly sure about, especially being a remote learning situation. Git was a new

system for me, and the course allowed me to see how, even when being in a remote work location, collaboration with a team across the globe is possible on programming projects. By pulling a branch of source code each team member can work independently of each other and then when completed push their changes into a new fork and then several of these forks, or individual forks, can be merged into the source code after addressing any issues that arise in the process. The process of working on a project, having a small piece that I was responsible for, pushing it back and waiting for my team members to look over my code as I looked over theirs in peer review before merging and submitting our project. Secure Coding was an eye opener, that I had taken with CS-310. The sheer scope of this course was amazing, I was expecting security policies along encrypting sensitive plain text information and proper methods of securing connection information through proper hashing. The course did cover these in projects, but as well proving understanding on some of the roots for memory leaks, buffer overflows and how check requests and returns communications, allowing for security issues like the Heartbleed Bug.

After my time in SNHU's Computer Science program I have arrived at the last point in the program, CS-499 Computer Science Capstone. It is the culmination of the various materials that were gathered and created over the last couple of years, allowing time to reflect on the projects, revisit those that highlight my strength and apply skills that were learned later in the program to flesh them out into a more robust and functioning program to be presented in a Portfolio.

## Portfolio Summary:

When deciding which artifacts that I wanted to include in my portfolio was not an easy choice. The limiting factor that I ran into with many of the options was the time constraint on for the modifications that I wanted to integrate into them. The choice that landed on was to take a simple python script from CS-350, Emerging System Architecture and Technology, used to continually measure temperature and humidity at a set interval, control some external LEDs, and store the results in a JSON file. It simple, with a lot of room that it could be expanded upon to highlight software design and engineering, data structures and algorithms, and database incorporation.

For software design and engineering, we took the original python script and finished the process that I had started in CS-350 of making the code fully modular. To that we began to design the process and procedures that were to be included to flesh out this script. To expand upon the complexity of the simple script I stripped as much code away from the main process organizing each method into its own file and those files organized into directories for controlling hardware of the device, and for controlling the files and information. To this this we also need to add access to MongoDB Atlas, and evolve the process of performing file maintenance during the inactive time of the device, as data measurements are only collected during daylight hours. Further details about this phase and a link to the original and final code is available in the grid below.

For evolved the data structures and algorithms in the original python script from CS-350 there was a lot of work that needed to be applied. As the device is a unit intended to operate for extended periods of time without human interactions, some type of file system would be needed

to ensure that storage space for the data is always available as well as when the processes should take place. For this we needed to flesh out the algorithm that was in place to detect sunlight and the paths that it took if there was or was not sunlight. In order to not skew data collection these processes would be better off to occur during the night, when the device is mainly inactive. Here we add an algorithm that checks the time and the day of the week. If appropriate it then performs the daily upload to MongoDB of the day's data, stores the JSON file in the appropriate directory(success or failed for upload) and if the correct day then it attempts to upload each failed JSON report and performs a clean up of the files removing any reports that are older than 7 days (unless the file has failed to upload then it is stored for manual upload). Further details about this phase and a link to the original and final code is available in the grid below.

The inclusion of MongoDB into the program was gateway that wanted me to create more than just a storage place for these files, but also to take the theories and philosophies I had gained at my time at SNHU and apply them to a structure system that I had not used before, HTML5, CSS3, and JavaScript within the span of a week to learn, design, and flesh out interaction from the MongoDB as a user-side interface to view the data graphically. This is an ongoing project, as full implementation as I originally plan was not achieved. Though the design is simple, it helped me establish skills that I was then able to apply to the design of this portfolio and a better understanding of these systems.