**Enhancement Two Narrative**

Tyler Blanchard

1/30/2025

For my second enhancement, I have improved upon a mobile application I had made previously by implementing advanced filtering options to very efficiently sort through a set of 100,000 data entries in addition to various improvements of existing features to enhance the overall efficiency of the application. Created in early 2024, this artifact is an inventory management application designed with the basic CRUD (Create, Read, Update, Delete) principle in mind to display a database of items in stock within a given warehouse or store, allowing for users (employees of the store/warehouse) to add and remove items from the app in addition to updating the quantity of items to accurately represent the inventory stock. The original artifact also included a basic search feature and notification feature to allow for users to find specific products and get notified when items run out of stock, however these features were implemented poorly and either didn’t work as they were intended to or not at all.

I chose to enhance this artifact and include it in my ePortfolio as while it did serve its basic purpose of displaying items for users to look through and manage, it was painfully inefficient to find items that were going close to being out of stock, especially if there were thousands or tens of thousands of different items within the database, which would make this application pointless in managing an inventory if you can even find the items going out of stock. Additionally, this artifact was the first big and (nearly) fully functional project that I had made in this program, which had plenty of issues that needed to be fixed, and I wanted to demonstrate my skills both in creating a mobile application such as this and in cleaning up and fixing issues to produce an even more polished and functioning application that can be better understood and maintained than the original. I have improved upon this application by including a filter option to sort data based on its name alphabetically and its remaining quantity (both low-high and high-low). With this filter option, I implemented three different sorting algorithms of various efficiencies and complexities (bubble sort, merge sort, and heap sort) to demonstrate different ways data can be sorted, and have documented each algorithm to explain their time complexities and their efficiencies (or inefficiencies) in sorting through very large sets of data. These improvements showcase my skills in understanding various common and complex sorting algorithms and their efficiencies/roles in sorting data of different sizes, and also in writing said algorithms to quickly sort data without using a database or external methods to do so. I have further improved this application by cleaning up functions, implementing security procedures for user inputs, and including error handling for uncaught exceptions, showcasing my skills in preventing unexpected crashes/behavior from occurring or data from being stolen (primarily via SQL injections) while still improving the overall functionality of this application.

From the work I have did for this enhancement, I have met course outcome 3, which I had planned to meet with this enhancement since the beginning of this course, and I have additionally slightly met outcome 5 as well, which both are stated below:

Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices (data structures and algorithms)

Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources

I have met outcome 3 as I have developed 3 different solutions in the form of algorithms of various efficiencies and complexities to sort data in 3 separate ways for better organization of data and to allow for easy identification of items close to getting out of stock. I have also partially completed outcome 5 as I have implemented simple security measures to prevent special characters in user input that may represent parts of commands/SQL injections from being processed by the database and have included more checks to prevent common exceptions such as integer overflows from occurring that could cause unexpected or undefined behavior within the application.

Throughout the process of this enhancement, I have learned quite a lot and have encountered and overcame a few challenges. One thing I have learned is the importance of organizing your code and implementing proper comments to ensure code is easier to maintain in the future, as certain code sections I had written before did not contain comments. I had to struggle a bit with figuring out what they did as I did not completely remember what each piece was doing, through fortunately most of the code was already organized neatly so I was eventually able to understand what each piece did and what needed to be changed so my enhancements could be used in this application. The biggest thing I have learned was the importance of implementing and utilizing the right data structures to efficiently handle the data you need. There are many different sorting algorithms that can be used to sort data in the same way, though they all have their own complexity to them and handle data at different speeds. Bubble sort, for example, is a very simple algorithm that was pretty easy to implement as it mainly uses two nested loops to swap data around until it is sorted, however it can take a very long time (relative to how fast computer processes usually take) to sort through a large set of data, with it taking over 3 minutes to sort through 100,000 data entries. Heap sort, on the other hand, is a much more complicated algorithm that was challenging to implement as it utilizes recursion and the concept of binary trees to sort data, but is vastly more efficient in sorting large sets of data, as it took barely a fraction of a second to correctly sort the 100,000 data entries. Based on these examples, it is always important to understand the data and the amount of data you have or will potentially have so you can choose the most appropriate algorithm to use.