**4-2 Milestone Three: Enhancement Two: Algorithms and Data Structure**

**Briefly describe the artifact. What is it? When was it created?**

**Category Two**: Algorithms and Data Structure

Artifact Name: Secure Coding

Origin: Coursework in CS 405 - Computer Science program

Category: Software Engineering/Design

The artifact is a C++ file that contains NumericOverflows.cpp.

The "NumericOverflows.cpp" file is a C++ program that shows methods for detecting and preventing numeric overflows and underflows. Template functions are for adding and subtracting numbers to check for overflow and underflow. The program tests signed, unsigned, and real numbers. It presents overflow and underflow findings to the user. The code handles arithmetic operations, data types, and numeric limit error detection.

**Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**

The selection of the artifact and the inclusion of it in my ePortfolio is to show my skills in the following:

Reduced duplication of code: I've developed the perform\_operation function to do away with add\_numbers and subtract\_numbers.

Error reporting improvement: The software has been modified to throw an exception when overflow or underflow is discovered, which will enhance error reporting. This makes it easier for calling code to handle errors.

Commenting: I've added a header remark outlining the goal of the program, as well as in-depth inline comments outlining the goals of each function and the main decisions made in the program. Additionally, I changed the comment at the file's top to include my name.

The artifact was improved in these key areas:

Improved Documentation: The improved code has explicit comments and comprehensive function headers that describe the goal, inputs, outputs, and behavior of each function. When other developers read or update the code in the future, it will be much simpler to understand.

Minimized Code Duplication: By establishing a single function (update\_value\_check\_overflow\_underflow) to handle both addition and subtraction operations, code duplication was considerably minimized. As a result, the code base became considerably cleaner and more DRY (Don't Repeat Yourself), making it simpler to read, comprehend, maintain, and test.

Updated Error Reporting: When an overflow or underflow is discovered, the updated code throws particular exceptions (std::overflow\_error and std::underflow\_error) as opposed to merely publishing a message to the console. As a result, errors are reported in a more thorough and unambiguous manner, enabling the calling code to handle exceptions effectively.

**Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

The course objectives are met by aligning the artifact with algorithms and data structure. Overall, the artifact shows how well a user can design reliable, adaptable, and efficient algorithms and how well they can employ a variety of data structures to support them.

**Reflect on the process of enhancing and/or modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

During the modification of the artifact, it could be challenging to understand the original code, what the code is meant to do, and how it operates. I learned to put all of these into proper perspectives before I could go on with the modification. I also learned to identify areas where code duplication can be implemented. In this case, since the add\_numbers and subtract\_numbers functions in the code have comparable structures but distinct operators, it was possible to combine them into a single, more flexible function.