**5-2 Milestone Four: Enhancement Three: Databases**

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

Category Three: Databases

Artifact Name: Secure Coding

Origin: Coursework in CS 499 - Computer Science program

Category: Databases

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 databases, data administration, and how to securely and effectively integrate databases into applications, while emphasizing the significance of secure coding practices. Below shows how the artifact is improved and the specific components of the artifact that showcase my skills in software development:

Language Proficiency: Understanding of syntax, data types, control structures, functions, and other language concepts are required for language proficiency in C++. The use of libraries, in this example the SQLite library, also fits under this category in this specific situation.

Database Interaction: This code exhibits a working knowledge of databases by using SQLite. Creating and running SQL queries, dealing with issues, and overseeing connections and statements are all included in this.

Error Handling: Correct error management is essential for any software application. In this code, errors from SQLite operations are detected and appropriately handled, such as outputting an error message, finalizing any prepared statements, and closing the database connection.

Understanding Overflow and Underflow: When doing addition and subtraction, this code is intended to check for overflow and underflow. This shows that the limitations and dangers of working with numerical data types are understood.

Code Structure and Organization: This code shows a comprehension of how to organize code in a logical and maintainable fashion. Functions are developed for specific duties such as carrying out SQL requests, adding, and removing integers.

Resource Management: A crucial component of resource management in programming is finalizing statements and closing database connections when they are no longer required. When done with a resource, it's critical to clean it up and release it to avoid leaks and other problems.

Comments and documentation: To describe what each section of the code accomplishes; inline comments are used all throughout the code. Code is simpler to understand and maintain when it has clear comments and documentation.

**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?**

1. The enhancement meets the course objectives by performing the following:
2. It serves as an example of SQLite library usage in C++.
3. It demonstrates how to make a table, insert data into that table, and create a SQLite database.
4. When performing addition and subtraction operations, the code looks for numerical overflows and underflows.
5. When an overflow or underflow happens, it logs these occurrences in the SQLite database rather than just printing them.

All the above aligns with the database category.

**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?**

The process of improving and changing the code can be quite instructive. During this process, I learned the following:

I learned that a greater comprehension of the original code is frequently required in order to improve or modify it. This covers each function's goal, how they relate to one another, and how each component affects the performance as a whole. Just as important as knowing "how" something works is knowing "why" it works.

Error Handling: As demonstrated in this artifact, effective error handling is an essential component of software development. It offers understanding into how the system responds to unforeseen circumstances and how to make the system resilient and dependable.

I also learned that working with SQLite can assist in understanding database interactions, such as building tables, entering data, and handling errors.

Possible difficulties include:

Understanding Existing Code: It may be difficult to comprehend what is happening if the existing code is complex or poorly written, which makes enhancement a difficult effort.

Debugging: There is a chance that a change will introduce bugs. Particularly for bigger code bases or when the error signals are unclear, debugging may be a time-consuming and difficult procedure.

Integration Problems: It can be difficult to make sure that updated or new components work properly with the current system. This can call for thorough testing and a thorough grasp of the overall system.