Ali Alshara

CS 499

11/17/24

Milestone 2 Narrative

The artifact I selected for my ePortfolio is a numeric overflow and underflow tester, originally created during CS 405. Written in C++, this program uses template functions to perform addition and subtraction while detecting overflow and underflow across multiple data types. The artifact outputs results for each operation and raises exceptions to indicate errors, showcasing core skills in software design, memory management, and error handling.

I chose this artifact because it highlights my ability to design modular and reusable code, addressing practical problems like numeric overflow and underflow. The template functions for add\_numbers and subtract\_numbers ensure flexibility and reusability for various data types. Additionally, the code’s robust error handling through custom exceptions demonstrates a focus on program stability and user feedback. As part of the enhancement process, I removed unnecessary dynamic memory allocation, simplified the overflow and underflow detection logic, and added detailed inline comments to improve clarity and maintainability. These improvements demonstrate my skills in writing efficient, maintainable, and professional-quality software.

This artifact and its enhancements align with several Computer Science program outcomes. By designing and evaluating a computing solution that addresses numeric overflow and underflow using algorithmic principles, I met Outcome 3, which focuses on managing tradeoffs to ensure reliability and efficiency. Additionally, by applying C++ templates and error-handling techniques, I demonstrated proficiency in using innovative tools and techniques to create a modular, flexible program, meeting Outcome 4. The planned outcomes from Module One have been met through these enhancements, and I aim to expand on these outcomes by incorporating security considerations, such as preventing potential vulnerabilities in memory usage.

The process of enhancing this artifact was challenging. Simplifying the code by removing dynamic memory allocation highlighted the importance of efficient memory management in ensuring program reliability. Refactoring the overflow and underflow detection logic taught me how to balance code readability with functionality, while adding detailed inline comments reinforced the value of clear documentation for maintainability. One challenge I faced was finding the right balance between efficiency and clarity when refactoring the error detection logic, requiring careful testing to ensure accuracy while reducing complexity. Ultimately, this process deepened my understanding of software design principles and prepared me to apply these skills to real world scenarios.

To further enhance this artifact, I plan to integrate a database for logging test results, enabling persistent storage and retrieval of data. This addition would allow me to demonstrate skills in database management and further align with Outcome 4. Additionally, I will explore implementing basic security measures, such as validating user input, to align with Outcome 5 and develop a stronger security mindset. These enhancements will make the artifact more robust and complete, reflecting my growth as a computer science professional.