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

Artifact 1:  
collection\_tests.cpp is a Google Test suite written in C++ that tests the behavior of a dynamic std::vector<int>. It includes a variety of unit tests that check how vectors behave when adding, resizing, or handling errors. These tests are grouped into categories and use fixtures and parameterized testing to make them organized and reusable. This artifact was created as part of CS 405: Secure Coding, during a module focused on learning how to test and secure software effectively.

Artifact 2:  
Form1.cs and Form1.Designer.cs are part of a Windows Forms application called TestRunnerUI, created using C#. This tool provides a graphical interface for users to run test executables (like the one generated from collection\_tests.cpp) and view the results. The application allows filtering of test output by passed or failed results and displays status updates to the user.

**2. 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?**

I selected these artifacts because they work together and show my full-stack skills, from writing secure and structured tests in C++, to building a front-end interface in C# that makes those tests easy to use. They demonstrate my ability to work with both backend logic and user-facing software.

In collection\_tests.cpp, I showed my understanding of test-driven development by using proper test structures, fixtures, and clear assertions. I improved the file by adding parameterized tests and even included an intentional failure case to better understand how error reporting works during testing.

In Form1.cs, I designed a clean and helpful user interface. I improved it by adding a label to show real-time status updates, a dropdown filter to sort test results, and output formatting that made the test results easier to read. I also included exception handling, so users aren’t left confused if something goes wrong. This project showed my ability to make functional tools that are also intuitive to use.

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

Yes, I met the course outcomes I planned to focus on in Module One. These artifacts directly support the outcomes by demonstrating my skills in testing, secure coding, and user interface development:

* Course Outcome 3: I designed and evaluated computing solutions by addressing both the technical testing logic and the user experience side of running and interpreting those tests. My software design choices ensured that the tool was functional, secure, and user-friendly.
* Course Outcome 4: I used well-established tools and techniques, including Google Test in C++, and Windows Forms in C#, to build a complete solution that delivers practical value. This showed my ability to apply different technologies appropriately based on the project layer (backend vs. frontend).
* Course Outcome 2: Through the GUI design and formatted test output, I demonstrated the ability to communicate technical information clearly. The interface was adapted for users who may not have a deep technical background but still need to interact with test results effectively.
* Course Outcome 5: I developed a security mindset by considering how the tool could fail or be misused and by adding error handling and feedback mechanisms to guide users and prevent confusion.

These artifacts clearly reflect my original goals from Module One. I do not have any updates to my outcome-coverage plan, as the work completed meets the objectives. That said, I’m always open to suggestions on how I can further improve the artifact or better align it with course outcomes.

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

While working on the C++ test suite, I learned the importance of breaking down functionality into small, testable parts. It also taught me how to cover edge cases, write reusable test code, and use parameterized tests effectively. On the C# side, one of my biggest challenges was figuring out how to display test information in a way that users would easily understand. I wanted the UI to feel intuitive and not overwhelm or confuse the user. I thought about other applications I’ve used or seen that handle test output or logs, and I used those as references for what works and what doesn’t. I also focused on keeping the original function of the test tool, to help users quickly understand if their tests passed or failed, while making it more visual and interactive. This is what led to features like the status label, output filtering, and a clean layout. Overall, this process helped me better understand how to balance functionality with user experience.