Edwin Jones

CS-405: Secure Coding

June 1, 2025

Process Summary

I implemented the body of each predefined test case according to its intended purpose, such as verifying capacity behavior, resize operations, clearing, and exception handling. Each test was written to be independent and deterministic, utilizing the SetUp() and TearDown() methods to isolate state between runs. I used ASSERT\_\* when a failure should halt execution and EXPECT\_\* when subsequent checks should still run. This ensured both robust test logic and readable failure output.

To fulfill the requirement for original contributions, I created two custom tests. The positive test, SwapContents, verifies that contents are properly exchanged between the collection and another vector. The negative test, AccessOutOfBoundsThrows, confirms that attempting to access an invalid index correctly throws a std::out\_of\_range exception.

All tests were compiled and executed using the Google Test and CMake. Each passed successfully, confirming that std::vector<int> behaves correctly under various operations. Throughout development, I followed secure C++ best practices including the use of smart pointers, clean memory management, and test structure that reflects professional industry standards.

A screenshot of a computer screen

AI-generated content may be incorrect.