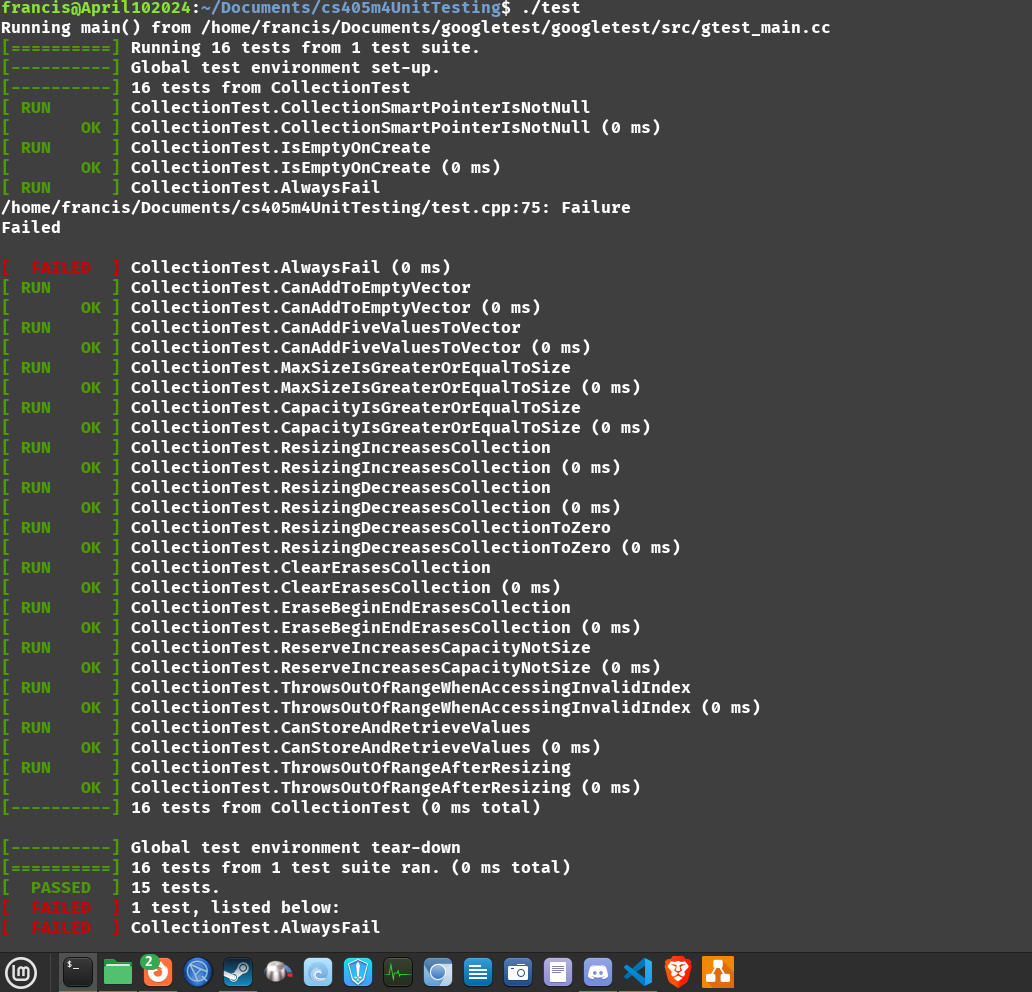
In order to guarantee thorough coverage of all possible circumstances, I concentrated on integrating both positive and negative test cases. Since the negative tests handled unusual circumstances like accessing out-of-bounds indices, they were especially crucial for confirming the system's resilience. This entailed making certain that the right exceptions were raised and dealt with appropriately, improving the code's dependability.



I made bespoke tests to cover cases that weren't initially thought of in addition to the established ones. The purpose of these tests was to confirm the std::vector class's functionality and edge cases. Tests were added, for instance, to make sure that actions like cleaning and deleting items worked as intended and to confirm the behavior while expanding the collection to different sizes, including zero.

Numerous bug kinds were found and fixed during this procedure. One prevalent problem was the improper application of claims. Whereas EXPECT was a better option, some tests first utilized the ASSERT macro, which stops execution upon failure. Tests might fail without pausing the entire test suite by fixing these assumptions, enabling more complete validation of later tests.

Managing edge situations constituted another category of bugs. Scenarios requiring caution included downsizing a collection to zero or accessing pieces larger than the collection's current size. To make sure the code behaved correctly and consistently in these situations, further tests were added to clearly check the behavior.

Another essential component of the debugging process was memory management. Efficient management of dynamic memory allocation and deallocation required careful use of smart pointers. In order to prevent memory leaks and make sure the tests were isolated and did not interact with one another, it was necessary to confirm that the collection was formed and reset appropriately before and after each test.

I added inline comments and descriptive test names to make the code easier to understand and maintain. This made it easy to troubleshoot and make future changes, in addition to aiding in the comprehension of each test's objective and functioning. It was evident from the remarks why each test was conducted and what it was intended to confirm. They also gave context for the test settings and expected results.