CS 405: 4-2 Milestone: Unit Testing

# Unit Test Names

Each unit test was named clearly to indicate its purpose, such as `CanAddToEmptyVector` or `ThrowsOutOfRangeWhenIndexTooLarge`, to reflect the operation being tested.

# Unit Testing

A total of 13 unit tests were implemented within the Google Test fixture framework. The tests included both positive and negative test cases to validate collection behavior, such as insertions, capacity, size after operations, and handling of exceptions.

# Negative Unit Tests

Two negative tests were created: one checks for an `std::out\_of\_range` exception when accessing an invalid index using `at()`, and the other ensures that calling `erase` on an empty collection does not cause undefined behavior.

# C/C++ Program Functionality and Best Practices

The test suite uses consistent naming, in-line comments, and assertions (`EXPECT\_`, `ASSERT\_`) that clearly communicate intent. The code follows RAII principles with smart pointers for memory safety.

# Process Summary

Initial challenges included confirming the project setup with Google Test and interpreting the intent of partially implemented tests. The debugging process included verifying all tests executed, fixing boundary condition errors, and using try-catch blocks for exception validation in negative tests.

# Console Output (Screenshot):

A screenshot of a computer

AI-generated content may be incorrect.