Final Report - Lachlann Lees, 23367810

This report reflects the development process, git usage, testing practices, and the implementation details of the Temperature Sensor Project. The goal was to create a system for generating, validating, analyzing, and alerting temperature data.

LinkedIn Learning Courses

Part one was to complete two LinkedIn Learning Courses to prepare us for the assignment. The two courses I completed were **Learning Git and GitHub**, and **C# Unit testing with xUnit.** These courses were pretty time consuming but gave me great skills to make life easier when completing part B of the assignment. I tracked the completion of these courses in my **Kanban board;** each issue was a course topic and added the contents I learned in the description of the issue. This helped me track my progress going forward and showed the things I learned in the courses.

Development Process and Git Usage

The application was built using an iterative and feature driven workflow with git branching

Git Strategy: a strict feature branch was used. Every new feature was developed in its own isolated branch (e.g., fault injection was developed in the branch fault-injection) before being merged into the master branch. This kept the primary codebase stable and allowed for safer, parallel work.

Commit Frequency: Commits were made frequently with descriptions that match; this practice provided a clean, transparent history clearly describing what was happening in each commit.

Feature Implementation

Core feature implementation

The initializeSensor function sets up the system using configurations from a file, establishing the standard operating range and the alert threshholds. SimulateData generates readings between 22 to 24 Degrees Celsius, intentionally adding small amounts of random noise. To counter the noise, SmoothData method calculates a 5-point moving average, providing a clearer trend. Monitoring is handled by two alert layers:

DetectAnomaly provides an early warning by noticing sudden spikes in temperature relative to recent history, while CheckThreshold issues a critical alert when the temperature crosses the hard safety limit. InjectionFault was implemented to simulate a cooling unit failure, causing a rise in temperature to test the alert systems responsiveness.

Testing Practices

Unit testing was applied using the xUnit Framework. Tests were written for all major features.

Examples from my work:

Integrity and Exception Testing – A core focus was input validation using Assert.Throws. This included testing of InitializeSensor to ensure it fails gracefully for edge cases such as empty names, minimum values exceeding maximum values, and temps being below zero, tests also confirmed that SimulateData cannot run before the sensor is properly started.

I also did tests on calculation and data integrity and Anomaly and fault testing.

Documentation

I created README with setup instructions, required packages, overviews, and a function guide. I also added the expected outcomes and advanced features. It included alot of key information that will be useful for the program.

Conclusion

At the start of this assignment, I wasn't really keen on LinkedIn Courses, but I am glad they were included as they helped a lot. I was pretty stressed during the completion of this assignment because I had three other exams to work on, but I got it done in the end. I am happy with my progress and how it turned out