What I wanted to do:

Plan/Objective

A separate front end GUI allows the user to clone repos through git into the program’s file folder. Then, they go through a preferences screen, where they specify at most 4 executables that they want to run as processes. If the user selects only one executable, the executable will be duplicated into process one and two, and scheduling algorithms dependent on differences in time or priority with processes will not be an option to select. When two or more executables are specified, the following scheduling algorithms can be selected (with a max of 3 at a time, and a minimum of 1) : first in first out (FIFO), Shortest job first (SJF), Shortest time remaining (SRT), Round Robin, and Multilevel. Then, the user selects how many times the program is repeated. Scheduling algorithms that use expected finish time as a baseline will benefit from more executions. Some programs may require user input. In order to simulate this, the backend will execute each program requiring input, and allow the user to answer the inputs. The answers will be recorded and used for executions during the testing step. All non-user I/O will be sent to another microcontroller with physical I/O present. After all preferences have been set, the user will select “start”. If any programs require input, that portion mentioned earlier will run, and inputs will be recorded. Then, each scheduling algorithm will run in its own respective thread. If there is only one scheduling algorithm chosen, it will run x amount of times across 3 separate threads if possible. During the run state, the GUI will show each thread’s state and estimated completion time. Once complete, the GUI will show a graph of time took to execute vs CPU usage, details on each scheduling algorithm, such as percentage of time the CPU spent waiting, total execution time, etc. , as well as a recommendation for which scheduling algorithm would work best for the user’s application.

Implementation:

A raspberry pi B3+ can be connected over ethernet, so a user can remotely log into the pi and clone their repos. Automatically running at startup, the GUI is written in C++ in order to minimize the storage and processing needs of the host program, allowing as much of the pi’s memory and processing to be dedicated to the user’s programs. Using a C++ file that targets .NET framework, I created a CLR project and started out