CMSC 335 Final Project

James Ford

In this project, I used multithreading to run a GUI-based program. The program called for multiple threads, all communicating with each other and ensuring they are sequenced properly to prevent thread starvation, race condition, livelocks, etc. Some of the major things I learned during this project was how to use threads, and how to update data in a GUI with binding. I had dealt with databinding previously while learning iOS app development, and I have briefly touched on multithreading using the FreeRTOS library on ESP32 processors, however not nearly as in-depth as this. I still don’t feel 100% confident that I’m using them properly, but the program I created does work. I also gave CSS a try when adding color changes to the light color, which it the absolute limit of my CSS knowledge. I also dealt with a major hardware problem when my laptop fan failed, but that is outside the scope of this project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Process | Expected Result | Actual Result | PASS/FAIL |
| 1 | Run basic program | Timer counts up. Car positions change. Traffic light cycles. Cars stop at red lights and continue on when lights change. | All processes work | PASS |
| 2 | Pause/Resume | Pause button stops timer, cars, and traffic lights. Clicking pause or start resumes travel. | Buttons work | PASS |
| 3 | Stop | Stop button stops all threads | All threads stop | PASS |
| 4 | Add car | New car added. While cars are running, new car will start. While cars are stopped, new car will be stopped and will begin travel when simulation is unpaused | Car adds and functions correctly | PASS |
| 5 | Add stoplight | New traffic light added. If simulation is running, light cycles. If simulation is paused or not started yet, light waits to cycle | Light adds and works properly | PASS |







