COSC 310- Project Final Report

Group Name: Pirates of the C
Group Members:
Kyle Keim (46335485)
Einar Schiele (64296734)
Mehul Raisingh (46168746)
Ryan Tschritter (47341862)

Project demo link: https://youtu.be/YZl0WcoFLdM

General Development:

1. *What did your team build? Is it feature-complete and running?*

For this project we developed an internet of things (IoT) sensor monitoring system for a large-scale commercial building; in this case a hospital. This system was designed to sense and monitor the changes in temperature of the environment over a period of time. Based on certain parameters entered by the user, the system will respond differently at different temperature levels and different levels of temperature fluctuation. The final product is feature incomplete but several sections are running independently.

2. *How many of your initial requirements that your team set out to deliver did you actually deliver (a checklist/table would help to summarize)? Were you able to deliver everything or are there things missing? Did your initial requirements sufficiently capture the details needed for the project?*

We did manage to fulfill some of the initial requirements including:

- Implementing a PID controller for the monitoring system.
- Developing a database for sensor reading to be logged and retrieved.
- Users will be able to monitor temperature sensors in real time through a web browser client
- System will alert the users when the temperature exceeds the threshold
- System will store past temperature data

We were not able to deliver all of our initial requirements and our initial time estimates were off by quite a bit. The initial requirements that we developed probably would have encompassed the production of a very basic IoT monitoring system but could easily be expanded in more detail to produce a better end product.

3. *What is the architecture of the system? What are the key components? What design patterns did you use in the implementation? This needs to be sufficiently detailed so that the reader will have an understanding of what was built and what components were used/created. You will need to reflect on what you planned to build vs what you built.*

We wanted to create a system that would distribute data between different containers with different purposes. The data would be collected through sensors, and then passed to a database. Users would be able to access data using a webpage that would display the current and historical data and alert if a threshold had been exceeded. The data would be transmitted using MQTT, where the containers would publish data on a topic and containers subscribed to the topic would acquire the data. Individually we were able to create all the components, however we lacked the sufficient time to fully integrate them together.

4. *What degree and level of re-use was the team able to achieve and why?*

The control system implementation consisted of many different modules. For the bandpass filter module we used pre existing libraries such as Scipy (scientific calculator for python) as well as other libraries such as numpy and matplotlib.

5. *How many tasks are left in the backlog?*

There are a significant number of issues still remaining in the backlog however I believe that some of them are much more insignificant tasks than others. With the objective difficulty of the project and troubles interconnecting various containers, I believe we still would have required a significant level of time investment for this project that many of us couldn't afford at the end of this busy semester.

CI/CD:

1. *What testing strategies did you implement? Comment on their degree of automation and the tools used. Would you (as a team) deal with testing differently in the future? Make sure to ensure that your testing report is updated to reflect what's actually been done.*

In order to implement continuous deployment into our system, we looked up a python-app.yml to allow us to utilize Pytests to automatically test our Python code before we made any pull requests to the main branch. In the future, it would be a good idea to look up various patterns of code testing to help in more uncertain areas of coding since developing tests for specific code tests can be surprisingly difficult before the code is layed out.

2. *How did your branching workflow work for the team? Were you successful in properly reviewing the code before merging as a team?*

We developed over 15 branches for this project for various purposes and we merged quite a few of the branches into the master branch but not all of them. We made sure to have someone else check over any pull request into the master branch and comment on most pull requests, even if they were minor changes.

3. *How would your project be deployed? Is it docker ready and tested? Provide a brief description of the level of dockerization you have implemented and what would be required to deploy.*

Optimally, our project would be deployed on a single system using various docker containers that are continuously side by side as new temperature values are inputted into the temperature monitoring system. Various parts of the system are docker ready and tested but not all of them. Most of the implemented sections are containerized and capable of running on their own but a lot of the docker networks remain unconnected. More work would be required to get input values moved from one container to another and the various MQTT broker related containers properly publishing/subscribing.

Reflections (Comment on the following items as a team):

1. *How did your project management work for the team? What was the hardest thing and what would you do the same/differently the next time you plan to complete a project like this?*

We all contributed to weekly meetings and even development of the project as well as managing our own time accordingly. We mostly worked independently of one another during the development process and followed a Kanban framework to develop the project.

The hardest part of the project was probably getting the various systems to hookup to one another over a docker network from varying addresses and varying containers/running processes.

If I were to do a project like this again, I would probably attempt to implement more code earlier into the production cycle as it is difficult to put together a large product in a shorter amount of time. Also having more frequent team meetings will work better in a project where less experienced software developers are trying to learn how to develop new systems they haven't tried before.

2. *Do you feel that your initial requirements were sufficiently detailed for this project? Which requirements did you miss or overlook?*

In terms of the control system, the initial requirements were not specific enough. Through the development, we realized that there were more details to consider such as the implementation of the PID controller, how it was going to be implemented, how it would be tested, how would the ki, kp and kd values be calculated as well as how this would communicated with other parts of the system. Although we did consider all these questions, a more detailed requirement would have been more beneficial.

3. *What did you miss in your initial planning for the project (beyond just the requirements)?*

We decided to cut out various parts of the project we had planned on at the start. We were planning on having a python backend server that would help translate different requests to different parts of the process but we decided to have the MQTT broker as the middle man to be published/subscribed to. We also cut out functionality for having user login/forgot password implementation.

4. *What process did you use (ie Scrum, Kanban..), how was it managed, and what was observed?*

We followed a Kanban framework as far as managing the implementation of project issues and development. We used a project board on github to create the Kanban workboard and used it accordingly to create work cards. We observed taking various cards from a to-do column to an in-progress column to a completed column depending on where the implementation of the task was at.

5. *As a team, did you encounter issues with different team members developing with different IDEs? In the future, would the team change anything in regard to the uniformity of development environments?*

For the most part, we all used python as a programming language and utilized vscode as an IDE. Python was a great choice as it is a very flexible programming language and vscode had very easy to use extensions when it came to github integration. We had very few issues when it came to using different IDE's and we wouldn't change our IDE usage come a future project of this nature.

6. *If you were to estimate the efforts required for this project again, what would you consider? (Really I am asking the team to reflect on the difference between what you thought it would take to complete the project vs what it actually took to deliver it).*

We probably would have needed better resources when it came to docker networking, and a considerable amount more time to implement a finished product. This project probably should have been started earlier in the semester and a better sense of total project time required from the get go. We are still probably weeks out from completion and it certainly took more time than we thought we would need as we started to run into walls of programming problems. It really doesn't take very much code to get a project like this running, but having the experience of knowing *what* to write and overcoming programming problems has to be over 90% of the time spent during our implementation of the product.

7. *What did your team do that you feel is unique or something that the team is especially proud of (was there a big learning moment that the team had in terms of gaining knowledge of a new concept/process that was implemented).*

I feel like even though we didn't end up with a finished product, we learned some very important industry standards in software engineering. Learning more about unit testing, especially in an automated fashion is a very helpful thing to learn. Also dockerizing segments of our product code presented a very unique challenge but allows for the product to be deployed on most common place platforms. Also using Github from the very start to end of this semester was an immensely useful learning opportunity as it can be a daunting topic to learn about especially for newer users.