

Personal Reflection Journal #1

Sprint #: 1

1. Plan: Planning sprint outcomes (200-300 words)

Here, you need to describe what was planned for the sprint (goals, activities, artefacts etc.). Support your answers with evidence.

- What was the planned (based on sprint) outcome?

Sprint 1 is the starting point for the software development of the project, where the first lines of code are pushed to the repository and software architecture diagrams of the system are created.

I was assigned the role of software developer however because I had greater experience in designing and coding web applications I took on more of a tech lead role, working in both the frontend and backend. This also meant I had significant involvement in the planning of the sprint for the development team.

Our initial goal for the sprint was to start the development phase of the project and have a baseline foundation of the web application. This was not an MVP but just a solid idea of the tech stack and an easy way for the developers to implement new features.

To accomplish this we outlined our plan for sprint 1 utilising Jira and Confluence. Within Jira we created relevant tasks from the user stories we outlined with the client earlier in sprint 0. And set the period of the sprint between March 22nd and April 4th.

Our first developmental step and number one priority was to create a baseline full-stack web application that consisted of a basic frontend (UI) and backend. We also wanted a basic graph to be shown with some static data points. The second priority was to understand the specific JavaScript packages we were going to use and start work on the system architecture diagrams.

— Jira tasks are listed below —

Here is our backlog of Jira tasks for sprint 1:

2023-03-26	CIR-26	Generate "charts/graphs" using library and display them	✓ Task
2023-03-26	CIR-27	Add dynamic scaling of "charts/graphs" based off a user's scroll wheel	✓ Task
2023-03-26	CIR-28	Connect editing of Graphs UI with api	✓ Task
2023-03-26	CIR-29	Connect delete existing graph UI with api	✓ Task
2023-03-26	CIR-30	Create application logic flowchart/user flow diagram	✓ Task
2023-03-26	CIR-31	System architecture diagram	✓ Task
2023-03-29	CIR-32	Implement React router and integrate with the Sidebar	✓ Task
2023-04-03	CIR-42	Research Recharts (JS Library) for "charts/graphs"	✓ Task
2023-04-03	CIR-27	Add dynamic scaling of "charts/graphs" based off a user's scroll wheel	✓ Task
2023-04-04	CIR-45	Use Case Diagram	✓ Task
2023-04-04	CIR-46	Data Dictionary Diagram	✓ Task

2. Execution: Executing the sprint plan? (200-300 words)

Explain your contribution to the sprint outcome according to the sprint plan. Support your answers with evidence.

- What did I contribute to the outcome?

My contribution was split into two main areas the first being the developer side and the other being the management side.

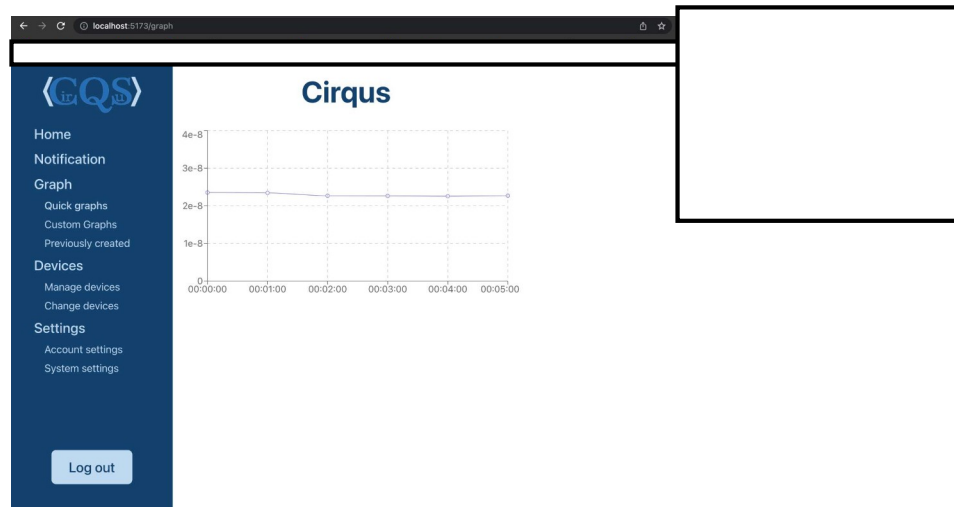
Regarding the developer side, because I was the only one who had knowledge and previous experience designing and building web applications in JavaScript, it was my responsibility to create the foundational web application. I set up the entire full-stack web app both the frontend component which we chose to do in ReactJS and the backend component being ExpressJS (NodeJS), as well as a simple local database. I pushed all this code to the repository, ready for the other developers to start their work implementing other features. As the sprint progressed I spent most of my time developing pushing code to the repository, however, I also spent time reviewing code from the developers on my team. I reviewed all merge requests as I was the only developer who had knowledge of both the frontend and backend.

Merge request that I reviewed:

The screenshot shows a GitLab merge request interface. The title is "[CIR-32] Implemented React Router and integrated it with the Sidebar". The status is "Merged". The merge request was requested to merge "CIR-32" into "develop" 1 week ago. The overview shows 1 commit, 7 pipelines, and 12 changes. The description states: "Created Router for the website and new pages. The graph has been transferred to the graph page. Edited 1 week ago by [redacted]". The merge was approved by [redacted] and merged by [redacted] 1 week ago. The merge details show: "Changes merged into develop with b07d2dbd." and "Deleted the source branch." The right sidebar shows the assignee, reviewer, labels (frontend), milestone (None), time tracking (No estimate or time spent), lock merge request (Unlocked), notifications (checked), and 2 participants. The reference is "mKQuantum/Projects/..." and the source branch is "CIR-32".

My next objective was researching potential packages to use for the graphing of data, this was to be a key feature of the web application so it was important to identify this early on. We initially chose to use a package called Recharts which I thought would work sufficiently however there was some unexpected functionality for our specific use case so we ended up switching to PlotlyJS at the start of sprint 2.

Here is basic web application with the original recharts graph:



Regarding the management side, I conducted and led many meetings with the development team discussing the project, which tasks we were working on, and assigning each member tasks according to their interests and preferences through the Jira backlog. Initially the developers on my team had some trouble as they weren't too familiar with using version control software such as git, so I spent some time teaching them how to use it properly and also showing them the best practices for merging to the codebase using merge requests on gitlab.

3. Analysis: Analyzing sprint using retrospectives (200-300 words)

Explain the following to capture your personal reflection for the sprint:

- What went well?
- What didn't go so well?
- What have I learned?
- What still puzzles me?

Support your answers with evidence

In sprint 1 a lot went well especially on the development side of things. I felt like I got a significant amount of code pushed to the repository in a short amount of time. Because key functionality was not a priority yet it made development progress quickly.

There were a couple of areas of sprint 1 that definitely could be improved upon for the next sprint. The main one being around the creation of developer tasks within Jira. The issue being that most of these tasks were really vague and didn't have a lot of detail because they were created by the team lead / co lead who didn't really have a good understanding of specific technical details. This issue meant it was harder for the other developers on the team to progress quickly as they struggled to understand the task. It meant that we had to create a lot of tasks on the fly throughout the sprint, which is far from ideal. This is highlighted in the sprint burndown chart which should be a line graph going down but instead it steadily increases over the sprint (some tasks were so vague they were never completed and had to be re-thought for sprint 2):

Sprint burndown chart ... going upwards 🤔



I understand that I'm partly to blame for this issue as I neglected reviewing through each task in the Jira backlog, however my time was tied up doing the bulk of the development. Looking back at the sprint I've actually learnt significantly more about management and have so much more respect for managers. It's much more difficult to stay organised than I originally thought it would be. I think for the next sprint I must spend more time creating quality tasks for the developers as it should make the sprint run more smoothly and successfully.

I'm already thinking ahead and one thing that puzzles me is how we're going to deploy this application. I think the best way to approach this problem is to ask the technical advisor for some advice.

4. Improvement: Improving for better outcomes (200-300 words)

Explain the following based on the analysis. You need to list and prioritise actions after the retrospective session and action those points. Make the improvement. Support your answers with evidence.

- How could I improve (action plan) for the next time I encounter a similar scenario?

The main point that could be improved upon, highlighted from the analysis above, is the quality of Jira tasks created for the sprint. If the tasks are detailed and thorough it should increase developer autonomy that being, they have a better understanding of what is and needs to be completed for the sprint and can quickly complete it by themselves. This results in the sprint running successfully and smoothly.

The action plan I could take to improve would be as follows:

1. Create quality developer tasks in Jira at the START of sprint 2

If I or another developer take some time at the start of the sprint to thoroughly review each task and create quality tasks in Jira. This should improve the sprints overall success.

2. Getting the Team Lead / Co Lead more up to speed on technical aspects of the system.

Another issue that was identified was that the Team / Co Lead created vague technical tasks at the beginning of the sprint. A way that I could potentially solve this issue would be to give greater insight into the technical aspects of the system and outline in more detail what needs to be completed.

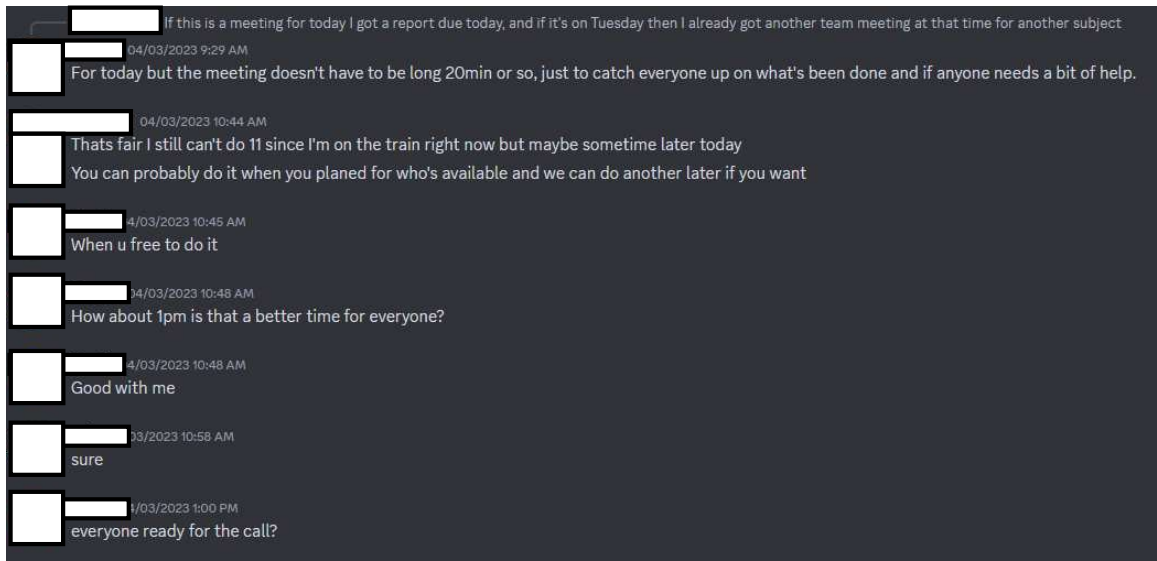
3. Have better structured meetings with developers

Being disorganised especially regarding the management side was an issue identified in the analysis. I need to do a better job at uplifting the other developers so that they can complete their tasks on time. One way I thought I could improve this would be to have



more up-to-date and structured standup meetings (just with the dev team) to encourage task completion, so that for the next sprint the burndown chart should be trending downwards.

Organising developer check ins / general meetings:



— **Personal Reflection Journal 2 below** —

Personal Reflection Journal #2

Sprint #: 2

5. Plan: Planning sprint outcomes (200-300 words)

Here, you need to describe what was planned for the sprint (goals, activities, artefacts etc.). Support your answers with evidence.

- What was the planned (based on sprint) outcome?

Sprint 2's focus and main goal was furthering the development of the web application by starting the implementation of specific features, one being websockets, allowing us to use real time data. The sprint was set over the period of April 5th to April 26th.

Using the learnings from sprint 1 I made sure this time to spend more time at the start of the sprint creating quality tasks for the developers. Because the majority of the UI had been completed in Figma I thought it would be best to create as many development tasks for the UI with the hope of finishing it this sprint. This would be a significant step forward in the project and would put us in a great position to complete development.

I planned to implement many features throughout the sprint, the main one being websockets allowing for real time data to be synced from the server-side to the client. This is a necessary core component of the web application as the data will need to be displayed in real time as it is monitored within the device (fridge). Before I could implement this feature I first had to test it, so one of the first tasks for the sprint was to investigate websocket implementation for Javascript.

Another core feature of the monitoring tool we're building is the valve map which is a diagram of valves for the fridge device running the experiments. This map has to be synced with real time data coming for the fridge as it needs to show which valves are open and closed. Creating this valve map is also a very high priority task for this sprint.

6. Execution: Executing the sprint plan? (200-300 words)

Explain your contribution to the sprint outcome according to the sprint plan. Support your answers with evidence.

- What did I contribute to the outcome?

My contribution was focused on development this sprint as we needed to start implementing core features of the web application and finish pages for the UI. I focused my time on investigating ways to implement websockets as I believed this was the most important feature we needed.

To accomplish this I created a basic websocket server using python that scraped data log files and sent a websocket message of each line, which was emitted every second. This was emulating the data produced from the fridge device we're monitoring. In our frontend application on the client side I used the native browser websocket api to listen for the messages sent by the python server and used the received websocket message as the data for the graph. This allowed the graph to plot data points in real time.

Graph plotting real time data points using websockets:

Photo A - 10 secs

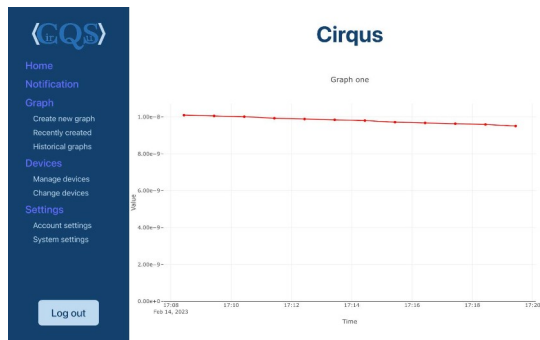


Photo B - 1 min

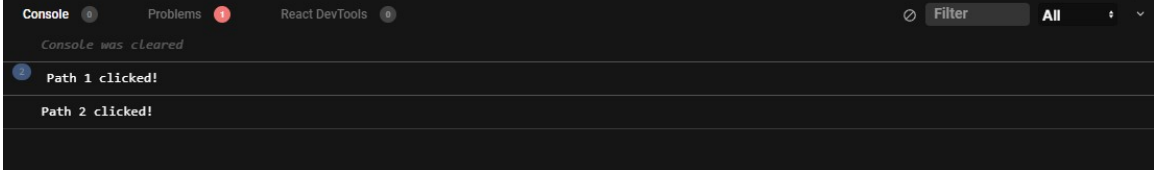



While this solution worked, to fully complete this task I needed to use the real data and network ports from the fridge. So I set up a meeting with the client to understand how this could be done, and they gave me the all clear to come down to the lab and test my solution using the real fridge.

I also worked on other features such as the valve map, trying to figure out how this could be created. I ended up creating an SVG, implementing this within React and using onClick handlers to make the SVG paths clickable. This was a smart solution and a great step towards completing the valve map, however I did not get around to integrating this within our current codebase.



Clickable SVG test for the valve map:



Apart from that I continued to do code reviews for every merge request providing feedback and advice when necessary for the other developers. I also attended a meeting with the technical advisor George who was very helpful in providing insight on how to deploy the application. I also quickly went through our architecture which got the green light which was a big relief for me.

7. Analysis: Analyzing sprint using retrospectives (200-300 words)

Explain the following to capture your personal reflection for the sprint:

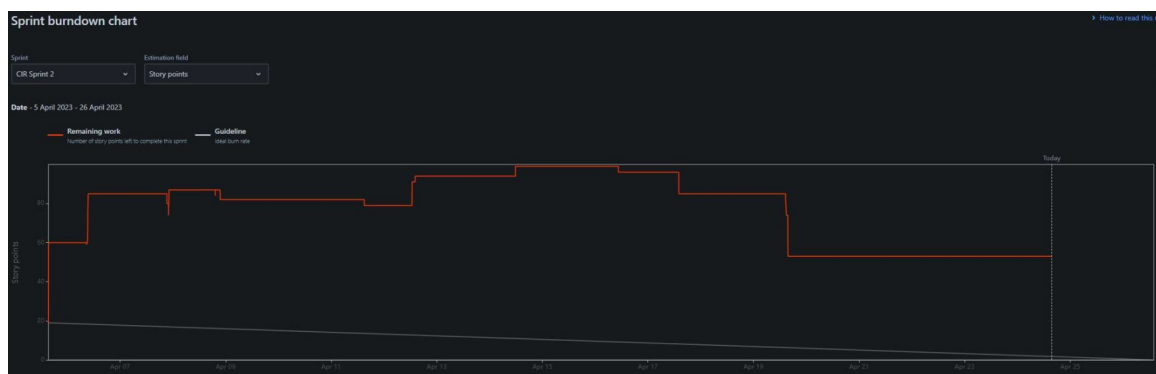
- What went well?
- What didn't go so well?
- What have I learned?
- What still puzzles me?

Support your answers with evidence

A lot went well this sprint, however I felt that the "honeymoon" phase had completely worn off and with StuVac slowing things down, I felt that motivation was at an all time low...for me atleast.

What went well was communication. I felt the team's ability to communicate effectively had greatly improved from sprint to sprint. Whether that's just discussing things in the group chat or giving each other feedback. Our communication as a group is fast and effective which is awesome. Something that I personally thought I did well in was utilising and executing on the learnings I had from sprint 1 such as "creating quality tasks at the start of the sprint." This helped make the sprint run a lot smoother. As a result the sprint burndown chart now actually goes downwards at the end of the sprint so that's a big win.

Sprint Burndown Chart going downwards finally 🎉



What didn't go so well was my ability to stay motivated throughout Stuvac. I was hoping to get a lot more of the development done as it would've been the perfect time to do so. However I felt a little burnt out and didn't do as much as I would've liked. That's not to say though I didn't do any work, I did a lot of the investigation and research work for both the websocket implementation and valve map throughout this period.

I'm feeling pretty confident about the project. The only thing that may be puzzling in the future is the websocket api for the fridge device. As it is quite specific and there are many data points we have to read so it may take some time to get the correct values.

8. Improvement: Improving for better outcomes (200-300 words)

Explain the following based on the analysis. You need to list and prioritise actions after the retrospective session and action those points. Make the improvement. Support your answers with evidence.

- How could I improve (action plan) for the next time I encounter a similar scenario?

In the analysis I talked about how I was struggling for motivation. I think this issue stems from the fact that I'm stressed out about finishing the project on time as my workload for this project has been, and continues to be extreme. This is not an understatement, I've spent a significant amount of extra hours each day putting in work to finish the project's most challenging developmental tasks.

After doing this reflection on the work I put in, I think I deserved a little break but this has inadvertently exacerbated the problem of not finishing on time. I can't do the whole project myself and perhaps one of the major mistakes I made was neglecting to give the other developers harder or longer tasks in the beginning. The reason this was the case was because it took a while for the other developers to be on-boarded onto the project and learn things such as git. I think it's finally time to assign much larger tasks for the other developers in an attempt to really push to complete the project.

I think I need to improve putting a larger amount of trust in my teammates even though I have doubts on their technical ability, however obviously I'm still going to be there supporting them. I really need to put greater trust and responsibility in them with the hopes that they will make a significant effort and contribution to finishing the project.

The action plan I could take to improve would be as follows:

1. **Create and assign much larger tasks for the other developers to really push to complete the project.**

I need more time concentrating on the harder problems of the web application such as the websocket implementation so I really need to encourage the other developers on my team to really push and complete other parts of the web app so we can finish the project on time.

Reference:

PDCA (Plan Do Check Act). https://www.mindtools.com/pages/article/newPPM_89.htm
The 4 Questions of a Retrospective and Why They Work.
<https://www.infoq.com/articles/4-questions-retrospective/>

Appendix:

Include any appendix. Evidence etc.

Solution Architecture document: [Google doc link](#)

I've attached relevant documents and pictures to support my reflective design journal inline within the paragraphs themselves.

Reflective Design Journal

1. Plan: Articulating the design problem (200-300 words)

Here, first, identify and articulate problem or opportunity. Explore the information available in full. Generate and screen ideas, and develop a plan or steps (design activities) to address the problem. Be sure to state your success or quality criteria and make them as measurable or testable as possible. You'll return to them later in the Check stage.

The design problem was quite simple. The quantum mechanics division at UTS had two quantum fridges that they use for experiments. They monitor these experiments in the lab, however they don't have the ability to monitor them outside of the lab. This makes it difficult identifying problems that may arise from the fridge when running an experiment when all the researchers are at home.

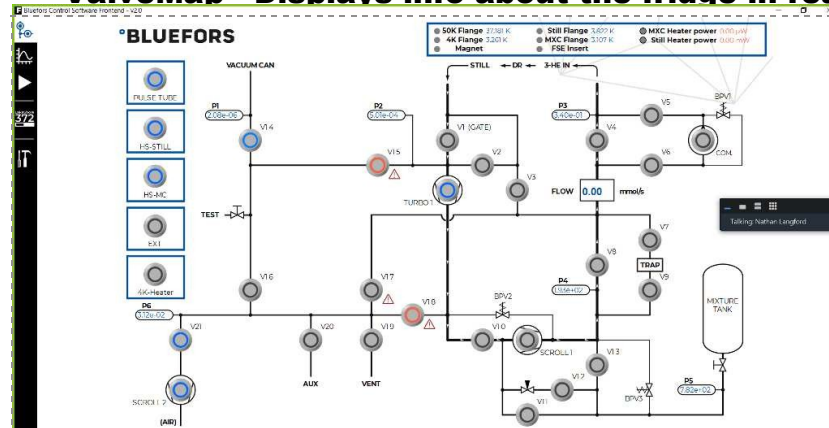
Our task was to create a monitoring tool that extended the features of the main one used in the lab but allow it to be viewed anywhere from both laptop and mobile devices. It will also have a notification system that will notify certain email addresses of a potential problem that may be occurring within the fridge so that the researchers could rush down and fix it.

The design problem I spent the most time with was related to the data aggregation for the web application. Firstly we needed to somehow obtain or make accessible the data from the device (fridge). And then secondly capture and show that data within the frontend / client-side (ui) of the application.

The device was currently outputting this data locally onto a laptop that was running the proprietary lab software. So the mission was quite simple: get the data from this laptop and make it accessible to the machine hosting the web application.

Sounds easy right? Well...not really...a big requirement of the design was to have a working notification system. This means the data will have to be in real time as we can't have a critical alert happen well after the crisis has already occurred. Also data that's being monitored in graphs or the ValveMap (diagram of fridge), also needs to be updated in real time otherwise what's the point of viewing them if the value is from 5 minutes ago.

ValveMap - Displays info about the fridge in real time:



Design Solution criteria for success:

Priority	Must be	Why?
1	Sufficiently real time	<ul style="list-style-type: none"> - Notifications must be sent within one minute of an issue occurring. - Viewing graphs and the valve map must be worthwhile, i.e. have correct values. - Data state must be within one minute.
2	Easy to implement	<ul style="list-style-type: none"> - We only have a limited time to develop the web app.
3	Easy to understand	<ul style="list-style-type: none"> - The client and other developers have to understand what's going on.

2. Execution: Designing the solution to address the design problem (200-300 words)

Proposing, designing, and questioning the possibilities (solution options) of solving the problem. Once you've identified a potential design solution, share it with your team or client and get feedback. Maintain a record or evidence of documented design and feedback data.

After visiting the lab in person and reading the proprietary software manual, I noticed there were two ways to access the fridge data. The first being a HTTP API that had access to all the data from the fridge. And the second being a WEBSOCKET that also had access to all the data from the fridge, it uses similar endpoints to the HTTP API however it also has the ability to LISTEN. I got this information from reading the software manual. This means in theory, we can receive the latest value of data from the fridge when its value changes.

I've worked extensively with both before and I know from experience that websockets support a bidirectional connection meaning we can both receive and send data, back and forth continuously. Using websockets we should be able to continuously listen for the latest value, receive that data and update it on the client-side, allowing for real time data flow. This is perfect for our use case.

The other potential solution is to poll the HTTP API endpoints over a set interval (one minute) to get the latest data values every minute or so. This won't fulfil the real time design requirement, however it still may be sufficient for our use case, emulating real time data flow to a certain degree. I do believe it's an option worth considering as it may be significantly easier and quicker to implement than websockets. It's also a good backup solution to fall back to if websockets don't work out.

This is an api request from the software manual that receives system information about the fridge device. As you can see, HTTP is nice and simple.

HTTP API Example:

Example 1 Requests system information in pretty printed format.

- **Request type:** GET
- **Request URL:** <http://localhost:49099/system/?prettyprint=1>
- **Request content:** <None>
- **Response content returned by the server:**

```
{
  "data": {
    "system_name": "Test system",
    "system_version": "1.4",
    "api_version": "1.4"
  }
}
```

You can see in the websocket example you need to first send a JSON command. You then receive multiple JSON responses each with their own state, this makes it significantly more complicated to implement as you need to filter through each response and pick the correct one with the value inside it.

Websocket API Example:

Example 1 Connecting to a secure port in local host with specified API key and using custom ID for command

- **URL:** <wss://localhost:49098/ws/system/?key=0352ebaa-6de5-4f1d-9091-17678d11dfd6>
- **Communication**
 - **sent**

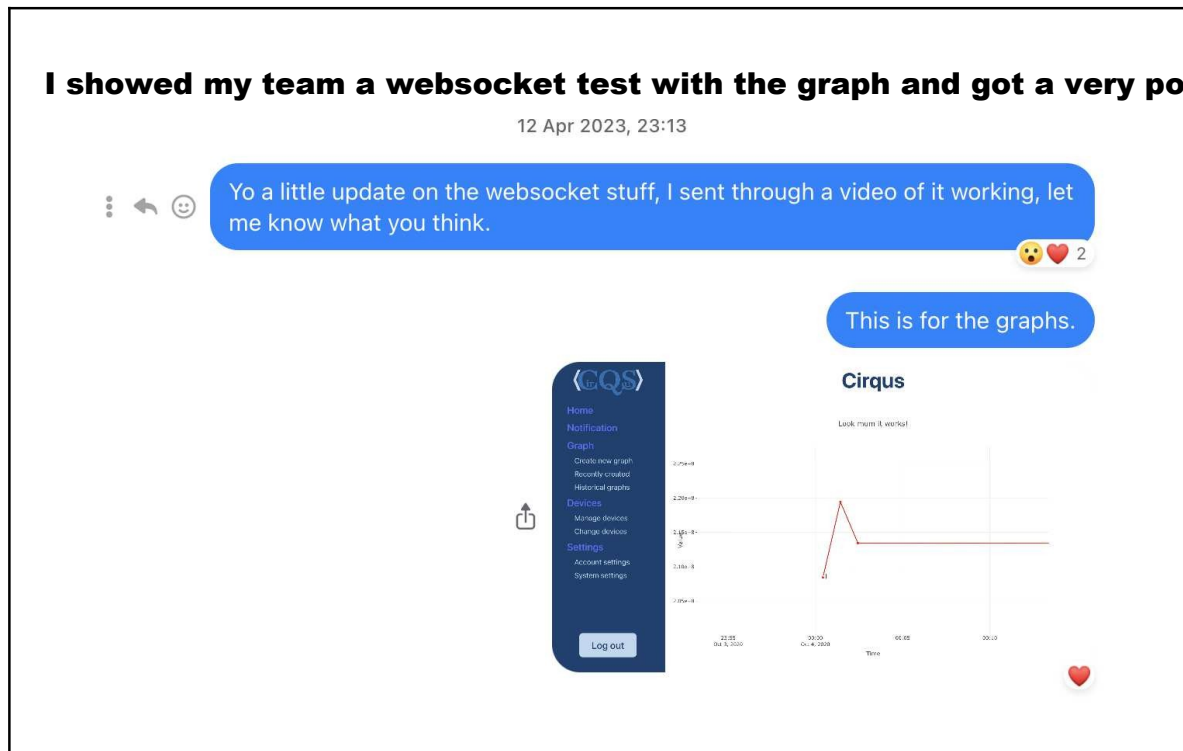
```
{
  "command" : "read",
  "id" : "2b64707c-17b0-11ec-827e-14dae904baea"
}
```

- **received**

```
{
  "id" : "2b64707c-17b0-11ec-827e-14dae904baea",
  "status" : "RECEIVED",
  "data" : {
    "command" : "read",
    "id" : "2b64707c-17b0-11ec-827e-14dae904baea"
  }
}
```

- **received**

```
{
  "id" : "2b64707c-17b0-11ec-827e-14dae904baea",
  "status" : "SUCCEEDED",
  "data" : {
    "system_name" : "",
    "system_version" : "1.4",
    "api_version" : "1.4"
  }
}
```



3. Analysis: Evaluating each possibility and feedback (200-300 words)

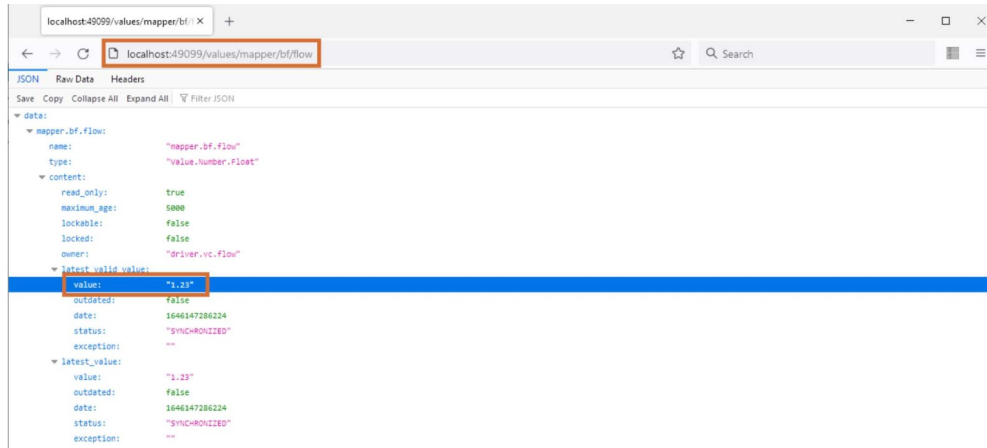
Analysing the feedback against the expectations that you defined in Step 1 (Plan), to assess whether your idea was a success. Describe how you evaluated which pathway is best. Show evidence of the evaluation process (e.g. diagrams/whiteboard doodles depicting different approaches, screenshot of discussing the issue with team members on Slack).

Using the design solution success criteria outlined in the first section, websockets is the clear favourite. This is because it is perfect for real time data flow, it's easy to implement within the development time frame of the project, the only criteria it isn't too good in is the last one, as it can be a little complicated to understand. However I think this is a fair trade off for accurate real time data.

To truly evaluate each solution I needed to test it with the real data coming from the fridge in the lab. So I organised a time to go to the lab and try both solutions, however the priority was to get the websocket api to work.



When I got to the lab, the researchers had already got the HTTP API working, outputting information about the fridge.



It worked. It required no additional hardware. It could be implemented on a Raspberry Pi.

After a few minutes, the fridge started to output data every second. This was great, it was working as intended however it raised a problem I initially didn't account for, related to storage. A researcher in the lab also asked if "using websockets would require significant storage space if we're saving each data point," at the time I didn't have an answer for him.

In my design I'm saving each data point within the database as it is received to allow for the ability to review archived or past data. This was a design requirement requested by the client. By requesting the HTTP API every minute it would produce one data point per minute, that means we are saving 1440 data points per day. Definitely manageable. However this can't be said about websockets as it would be producing one data point per second, resulting in 86400 data points per day. That's significantly more data being saved to the database each day.

But running the numbers and using a bit of mathematical wizardry I don't think storage size should be too much of an issue. Using CHAT GPT I asked it to estimate the amount of storage space taken by a small JSON object in a postgres database. It gave the answer 20 bytes. I used this number in my calculations.

Storage size comparison:

HTTP	1440 / day	Daily (20 bytes x 1440 = 28.8 kb) Yearly (28.8 kb x 365 = 10.5 mb)
Websockets	86400 / day	Daily (20 bytes x 86400 = 1.7 mb) Yearly (1.7 mb x 365 = 620.5 mb)

As you can see from the results 620.5 mb per year for the websocket implementation is easily manageable.

4. Improvement: Improving the design for desired outcome (200-300 words)

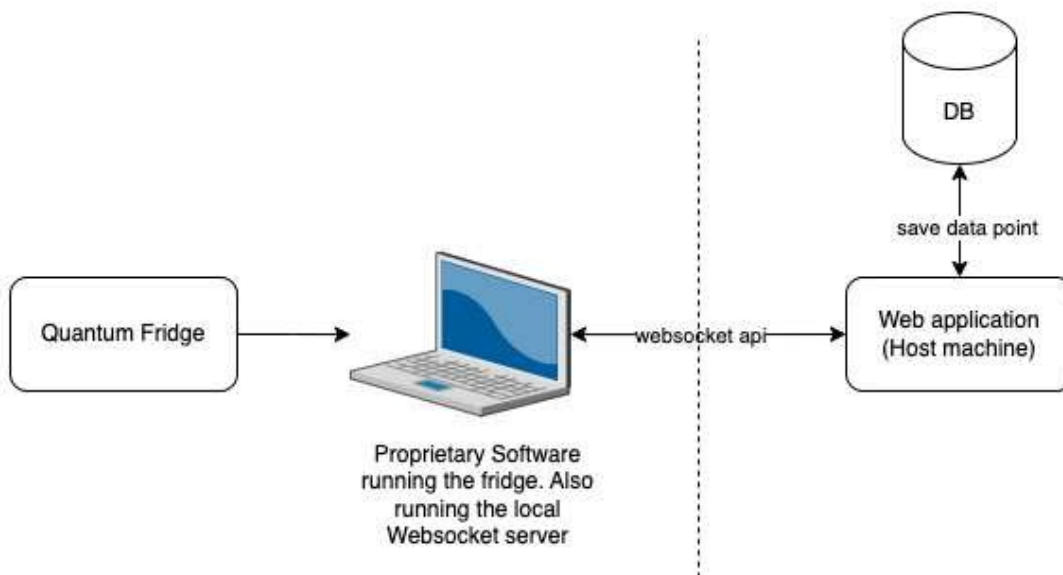
Explaining the outcome of choosing the solution or pathway and improvement (actions) made in the final design based on the feedback analysis from Step 3 (Analysis). This is where you address or action the feedback (reflection in action). Show evidence of improvements.

Considering the two options I had, websockets was by far the most ideal solution for real time data aggregation for the fridge device. When I showed the researchers in the lab that I had got the websockets api to work they were very excited, so this was a very strong positive indicator that I was picking the correct solution.

As for improvement, obviously storage shouldn't be an issue so there's no need to optimise the solution when saving the data into the database. However I did come up with a potential backup idea if it actually becomes a problem. We could average the data points gathered over a minute and save that value into the database. This would drastically save on storage space while still being somewhat accurate. It's not a perfect solution by any means but more so just a backup in case storage space becomes an issue.

When reflecting on improvement it just occurred to me that I don't actually have to pick one solution over the other I could use a mixture of both. Perhaps data that isn't too critical and doesn't need to be updated every second but still needs to be incorporated into the web app might be better suited to use the HTTP API.

As a final conclusion I've created a diagram of how the websocket solution will function within the overall system:



Date: 29/04/2023

Session: Wed 02

Reference:

PDCA (Plan Do Check Act). https://www.mindtools.com/pages/article/newPPM_89.htm

Appendix:

Include a link to solution architecture document and any other items to support your reflective design journal.

Solution Architecture doc [link](#) doc link

I've attached relevant documents and pictures to support my reflective design journal inline within the paragraphs themselves.