*GitHub has been an invalueable tool for this project. Each sub-task has a individual card, where I have been able to add notes, images, tables etc, all in markdown so they can be copy and pasted directly into my word processor, Zettlr. This has and pulling in information for the TMAs more straight forward. The project board with column: Todo, doing and done makes it easy to see the current and upcoming state of work. Card can be moved, and therefore priotisied easily, as well as grouped into "Milestones". In my WaterScrumFall process, these have acted like stages. (appendix 30) Finally, cards can have labels, which has been useful when assigning sizes to cards, as these can be filtered easily.*

*Mocking up the HCI was also a useful activity. LucidChart (ref) was easy to use, and provided in built mobile components. Drawings could then be replicated and small changes made. They also helped me identify which components could be reused, for example, the Vegetable Details page. As React Native is a component language, the mock-up elements map quite closely to what the code components would be. For example, "screens" are the other most component, with smaller "child" components such as lists, and list elements.*

*Before starting implementation, I should have added a story to complete the research tasks such as those resources identified as high risk. I left researching tools, or APIs, such as the weather API or gathering vegetable data until starting the story, at which point it was almost too when, as my schedule has only considered implementation timings.*

*Learning React Native was also harder that I expect. I have worked using Vue previously, so assumed they were more similar then they are. For each story, specificially earlier stories, alot of time was reading React Native documentation, slowing down development. I should have dedicated a few weeks to focus on learning React Native, instead of learning on the go. Although, I did still enjoy this process and after the initial learning curve, I could start to identify similarities with Vue, which made it easier.*

*Contrastingly, I did instead spend alot of time creating and assessing user stories, adding mock ups, testing criterial, acceptance criteria, texual description etc. This meant when it did come to executing a story, I had a clear idea of what needed to be done, which definitely influenced rate of development; 10 stories in around a month.*

*Postman was a tool which also helped during development of the Rails API. (ref) It allowed me to test the API before any front-end implementation, easily adapting parameters, URLs, and checking what response and status are returned for different methods. As the Rail API was implementing a JSON API ([ref](https://jsonapi.org/format/)) the body and responses were relatively straight forward. Postman also provides automatic documentation of the requests, and therefore API, which will be used an a way of evidencing work done. The API is also a successful interface, and I like how the front-end and back-end are completely seperated. This gives me more piece of mind when I am working on a piece of code, that changes are in isolation. Additioanlly, becuase I have experience in Rails, I found it quite quick and straight forward to get the API up and working. Allowing me to focus more time on the front-end.*

*Throughout, I would have liked to do more testing. When estimating story times and my project schedule, I should have factored in an extra hour for each story to implement tests. I did some basic test-driven development when I was struggling on something particular, however I will now go back and add some more.*

*Finally, I appreciated having friends and family who I could send the questionnaires, mock-ups and MVP too and get quick feedback. I enjoyed that I wasn't required to prioritise stories, as I feel like I would prioritised different features to them, making the application less useful. For example, the UX improvements are not something I usually am concern about improvement, however they are the majority of the users feedback which means I have learn't more about UX that I expected when starting the project.*

*If I were to do the projcet again, I would first schedule time upfront to learn more about the chosen language. I would also create a story for each of the research tasks, and understand the importance of these more. I would make stories for UX, if I had known what an important aspect this would be. For example, I hadn't a story about creating the navigation bar at the bottom of the screen, becuase I hadn't given it much thought. Although it appeared in the mock-ups, my stories were created from user requirements, where they understandly, assume there is some sort of navigation, and therefore don't state it. It also took a few hours work to test a few different navigation bar component libraries before choosing one and a story could have tracked this work more transparently.*

*Although an app was prefereable from initial requirements, I could have spent more time looking into other options, For example, web apps are now websites just displayed on decive in a more effective way, which could have worked. However, I wouldn't have got to learn React Native. Finally, I would aim to have a better plan before the project start date, because in the time between the start and TMA01, I wasn't really sure what to do. Although I did get two working prototypes, comparing Vue Cordoa and React Native, I hadn't really a plan and so felt my project actually started after TMA01, and so some development time could returned if I had been organised before the start date.*