Assignment: Riding on Rails

By Zhu Hanming A0196737L

Computing for Voluntary Welfare Organisations (CVWO) AY2019/20

Final App: https://www.hityourtargets.xyz

Final Submission

As I'm writing this report, I feel both happy and sad – happy that I've learnt so much along the way, but sad that I did not **hit** many of the **targets** I had initially set out for.

Prior to this assignment, I had zero experience with backend development. I had some experience with frontend, but it was accumulated through a few personal and rushed projects, many of which I had done without much guidance. As such, my understandings and skills were quite shaky, my foundations weak.

This is why I saw CVWO as an opportunity – it is a programme I really want to be part of, and it is also a chance for me to revise and refine my skills. I had also been considering exploring backend development, and this project was a good launchpad.

And I'm very glad I took up this assignment! I learnt so much from it. I'm also glad I took the time to think through what I wanted out of this project before I commenced, which allowed me to get the most out of this project.

Let us start by taking a look at what went well.

The Good

I've always loved UI/UX design – it amazes me what good design can bring to the world. Oftentimes, what makes the difference between a good app and a great app is how it feels to the user. I love designing such experiences and had quite a fair bit of UI/UX design experience. However, what frustrated me was that my frontend skills were often not enough to bring the app of my dreams into fruition.

This is why my goal for CVWO was clear – to design a beautiful app and code it out myself. I wanted the app to have a seamless user experience, delightful user interface, and have it built out in clean code for both frontend and backend. In most of my projects so far, one thing I've always been guilty of is messy, hacked-together code, especially in terms of CSS. I didn't enjoy reading my own code.

With all of the above in mind, I set off and managed **to-do** the following:

- **Design**
    - Design mock-ups of my ideal product before building. This design process is to be done without being limited by what I am capable of building, i.e. design first, then think about how to build.
        - Mock-ups were done first via Adobe Experience Design before building.
    - Experiment with a new CSS framework besides Bootstrap and learn to use frameworks such as SCSS or LESS.
        - Used Bulma SCSS.
- **Clean Code / Refine my Frontend development**
    - Learnt to use SCSS as mentioned above → way cleaner code than just CSS!
        - Utilised flexbox fully for this project

    - Set up eslint and prettier for the project, which helped to enforce coding standards at all times and stop me from committing any atrocious code.
    - Learn React Hooks and React Contexts.
        - Practiced by not having a single class in the code, only functional components (which is actually easier to build than classes!).
        - Created and used a total of 6 contexts! This is excluding contexts from third-party libraries.
    - Improve my React Redux
        - Prior to this, my code for Redux was very messy. Having no clue about Redux Persist, I manually persisted everything using LocalStorage.
        - In this project, I researched more into the different supporting libraries out there for Redux, and decided to use Redux Toolkit.
    - Learn and try Typescript.
        - Did it!
- **Backend**
    - Pick up backend development.
        - Learnt Ruby on Rails.
    - Try out JSON Web Token authentication.
        - Did it!

One of the above points I'm really happy to have achieved was that I manage to build what I thought I couldn't. I had designed the List view to be very similar to another to-do list app out there in the market – TickTick. I loved how intuitive their user experience was, and I wanted to adapt it and make it better. Similarly, I wanted a Trello-like clean drag-and-drop experience for my Kanban view and a very techy dashboard layout for my Main view.

I had no experience building any of those types of interfaces and was very hesitant initially about how much I can achieve. I'm very glad to say that I manage to achieve my goals (almost) completely! I've also successfully made all views – Main, List, Kanban – to be mobile responsive (which is pretty much a bare minimum in today's web dev industry). It felt very liberating to do what I thought I couldn't, and I really learnt a lot through this process of figuring out how to make things work, and how to do it cleanly.

This leads me to my next point on code style. As I mentioned, I wasn't very happy with how my code ended up during many of my projects (partially due to the boilerplate code needed for React class components). I saw major improvements in my coding style during this project, and I'm very glad about that too. The code I have written is also much easier to build upon than before.

This list of positive takeaways is long, but perhaps we should take a look at the (perhaps longer) list of shortcomings for this project. As much as I took a lot away from this project, there were many things that I had failed to achieve.

The Bad

I was very ambitious at the beginning. In my initial rollout plan, I aimed to have features such as mailing services, social media integration, notifications, gamification, Cron jobs, Docker set up, possible PWA expansion etc. Most of these features I mentioned here were, unfortunately, not implemented. Some of these are not easy to accomplish e.g. Docker, gamification, but some of them are low-hanging fruits.

I personally feel that there are two reasons as to why this happened – firstly, I underestimated how busy I would be once school started, and secondly, I was being limited by my backend proficiency. For the former, I heavily overloaded this current semester and ended up with little to no time to spend on this project. I could only squeeze out bits and pieces of time to work on Hit Your Targets and never reached the efficiency I had during the holidays. Having evenly spread out the workload allocation at the start, I started to see my rollout plan fall apart.

However, I feel that the latter reason might've played a bigger role – I was not sufficiently proficient in Ruby on Rails. Many of the logic I could visualise happening at the backend could not be implemented fully due to my inexperience. For example, I never got that far with the social media login – I wasn't sure where to start. Paired with the limited amount of time I had, I logically invested my effort on where I could bring about greater results – frontend. Thus, a lot of the logic for the app is actually handled on the frontend side, with the backend acting merely as storage, with minimal, pre-existing logic.

That being said, it's not as if I had no takeaways from the time I spent on the backend. My lack of experience spurred me to read up on Rails quite a fair bit during the holidays, and I learnt a lot in the process. I also did up the JWT authentication, albeit with the guidance of a tutorial, and lots of other necessary logic, all of which helped to make my app what it is currently. These experiences all helped to build on my previously non-existent knowledge of backend development. I most certainly took a lot away from this, just that the lack of practice and experience in applying said knowledge made me very slow to progress.

The Summary (Ugly)
As this project comes to a close, I'd just like to express my appreciation to the CVWO team for this opportunity, regardless of whether I ultimately get in or not. It can often be hard to start learning – especially when it comes to something completely new – but the existence of some guidance, no matter how detailed, can bring about comfort along the way. The CVWO assignment document helped me to clearly visualise what I can and should work on, and helped to show the way, and I checked back constantly to make sure I was on track.

It's also quite interesting how many people seem to look down on such "small-scale" projects like a to-do list. After all, it doesn't seem like a difficult or sophisticated app to build at all. Everyone can build a to-do app these days. However, it's only when we actually start building that we realise even the simple can be very complex – building an app that does just one thing is easy, but building an app that does that one thing very well is not.

Hit Your Targets is far from completion. I intend to see this project through – to build the remaining Calendar view and implement all other features required for an app like this today. I look forward to bringing the rest of the app to life.

Let us hit our targets together.