CVWO 2019/2020 Final Submission

By: Andre Wong Zhi Hua

This journey of my CVWO assignment have been a rough but fulfilling one. Not only do I get exposed to technology stack that I have not tried before, like Ruby on Rails, I also get to deepen on knowledge on areas that I am strong in, such as React. I certainly felt that my problem-solving skills have improved after this assignment.

Backend

For ruby on rails, one of the few difficult tasks I faced was implementing the database. Originally, I created by rails project with SQLite3 because I just started out using rails and did not know what to expect. However, as I progressed, I realised that Heroku does not support using of SQLite3, and thus I had to migrate over to PostgreSQL. It was a tedious process but a necessary refactor.

Furthermore, I also implemented the login feature, which required my database to store credentials of users and to link each user to their to-do items. I also focused more on the database side so that when I query information, the data is filtered in the database side using Active Record Query Interface, ensuring it does all the heaving lifting.

The login process was done using omni-auth in rails. This uses google authentication, which I believe is much easier for users when they want to access the app and easier to implement as compared to using normal email and password logins. Another reason why I implement google account login is because in future implementations, I would also like to add in the feature of adding task to their google calendar, adding reminders and so they this to-do app can be synced to their phone.

Frontend

For react, I chose semantic-ui-react for my components because cvwo also uses it. From there on, it was a matter of matching and check if all the component works as intended. I also added redux integration to react. Since I had previous experience in developing a react app with redux, this was relatively easy for me.

I added react-router-dom to handle the routing aspect of my front end. Because I have to check if user is signed-in, there was the main page that everyone can access, and there were the protected pages, which only people who are signed in can view. I took me a while to get the hang of routing pages but in the end I still managed to set it up and works perfectly.

Implementing typescript was a nightmare for me as I have never done it before. When I decided to migrate from JavaScript to Typescript, it was not an easy process for me as I struggled to figure out how to fix all the errors on how the .ts and .tsx files were not being processed into .js and .jsx files. But eventually, after a few days of

searching for a solution, I found someone facing the same problem and fix this error that was bugging me.

In addition to Typescript, I also implemented es-lint, which helps to check for syntax errors in my code. This helps me to ensure that my code is more consistent and more readable.

One feature that I am proud of is the tagging feature. I feel that the adding of tags is now much more user friendly as compared to my mid-term assignment's progress.

After finishing up most of my features, I then went to style my react app. I tried a few color themes that will look nice for the app.

Further thoughts

One thing I am still working on is implementing rails testing using cucumber and react testing using jest. However, due to time constraints and difficulty of this task, I decided to not push it in to GitHub yet.

One improvement that I would like to do is making this to-do app available on mobile browser. Right now, it can only be displayed nicely on the desktop screen but not on mobile browser.