CVWO Final-Assignment Report

Name: RUPPA NAGARAJAN SIVAYOGA SUBRAMANIAN
Matriculation Number: A0217379U
Assignment: Riding on Rails

Final Website: https://just-to-do-it-website.netlify.app/
User Guide: https://github.com/sivayogasubramanian/just-to-do-it/blob/main/final-assignment-submission/just-to-do-it-user-guide.pdf

Final Submission:

I am writing this report on the day of submission (30 Jan 2021) and I feel happy that I managed to pull off something that I previously thought was very difficult to do it by myself. This is my first ever website and any project of this scale that I have worked on. It was extremely tiring and painful to learn completely new things on my own without any guidance. However, this aspect of the project enabled me to learn some of the basics of web development and I can say that today I know much more about web development than I knew 2 months ago. This I consider as a victory.

Before this assignment, I did not have any experience in frontend or backend development. I had never worked on a project of this scale. What I knew was basic HTML/CSS which I learnt when I was doing CS50 edx module before university started. This meant that I started off the project with almost no knowledge in any aspect of the project. I had to learn everything from scratch which proved to be extremely challenging.

Looking back, I would like to thank Hanming for recommending this project to our CS1101s tutorial group. This is because without his introduction to CVWO in class, I would have never thought about doing this project. I must say that this project has taught me a lot of the basics of web development that I am sure will be relevant in the future as I progress through my computer science journey. I would also like to thank the CVWO team for this opportunity and the assignment document which helped me better organize my learning.

My accomplishments and shortcomings:

- **React and Redux:**

  During this difficult project, what inspired me to carry on through setbacks was the fact that when I add a feature, I can see it working with my own eyes. This meant that I naturally liked frontend development because it allows you see your application working and changing as you implement new features.

  Therefore, I spent a lot more time and effort into learning react. I must say that I have learnt about react more than the other components of this project. I managed to use many react hooks and even made custom ones to implement new features. I also managed to stick to the rule of allowing only one level of prop passing throughout the application. This means that a parent component can pass props into the child component, but the child component is not allowed to pass the same prop down again. In my opinion this made the code cleaner in general as it is more readable and scalable in the future.

  This was done using Redux. Redux was extremely powerful because it had a central store that can be accessed by any component which meant that there was very little “prop drilling” through the components that I created. However, I regret that I did not spend as much time as I should have because I started implementing Redux too fast and came to realize that it had a lot of unnecessary boilerplate code that made the code look messy and longwinded. On hindsight, I should have spent more time in exploring better alternatives that solves this problem like using Redux toolkit which would be more effective in maintaining code readability in the long term.

- **Rails:**

  I came to realize that Rails is extremely powerful very quickly. This is because I was able to have a REST API running in less then 1 week because a lot of the code can be generated using terminal commands. I am proud that I have a working API that uses JSON web token to authenticate users which I learnt using the help of an online tutorial.

  However, Rails efficiency meant that I spent very little time in exploring more advanced use case of Rails. I initially wanted to have 100 percent test coverage using rspec tests but was not able to carry on with it because I had to focus on frontend development. Furthermore, using JSON web tokens was done by following a tutorial online when I should have done it manually by myself just so that I can learn.

  Even though, I spent a lot of time learning Rails in the beginning of this project, I felt that throughout the project, my backend knowledge was very weak. In the past 2 weeks, I was exploring the idea of have user preferences set up, but I was unable to find a solution and therefore the feature is still not implemented. Even though this can be easily done by adding an additional model, I was hesitant to do so because it was nearing the project deadline and I did not want to complicate things especially towards the end. I have much more to learn before I am comfortable with Rails, but I am happy to learn the basics of backend development as it has enabled me to make a working website today.

- **Design (UI/UX):**

  I did not have any prior experience in UI/UX development but fortunately there are great UI frameworks that are widely used for React frontend development. I chose to use Material UI because I was always a fan of Google's material design concept and was very happy that Material UI is one of the more popular React UI framework. I had a great experience working with Material UI as it was quite simple and easy to use given that their documentation online is extremely detailed with demos. Not only that, but Material UI also gave an elegant and a simple way to edit/override the styles using CSS properties which meant that I was able to customize the design when needed easily.

  Using Material UI has allowed me to say that Just To-do It is fully mobile, and tablet optimized. I spent a lot of time making sure that the mobile and tablet equivalents are just as good as the web version because let us be honest, everyone is on mobile nowadays and having a todo manager accessible in your pocket is necessary to ensure that people use the application.

- **Others:**

  I am also happy to say that I managed to solve a personal problem that I had with todo applications that I have used in the past. Todo managers that I used previously lacked the ability to organize todos.

  Just To-do It solves this problem using tags. For example, if you create a todo with the tag "CS2040s", a dedicated view will be introduced in the navigation drawer which filters all todos with that tag. This is extremely good for organizing todos in general and my personal favorite in terms of features.

I also managed to host the Frontend and API successfully on Netlify and Heroku, respectively.

I also used Eslint and Prettier for my project using the Visual Studio Code extension which helped to enforce strict coding standards throughout the project.

What could have been done better (a lot):

My main goal for this project was to meet the assignment requirements and add extra features. This means that with my relative lack of experience in projects of this scale, my code in this project tends to be messy and could be improved so that it is more readable. Even though, I tried my very best to improve the code quality of this project towards the end of the project, I am sure that it might not been enough. I have learnt having beautiful code is also as important as having a lot of features because we will rarely be working alone in the future and having code that is messy will greatly hinder teamwork and progress.

When semester 2 started and I was exposed to Java which is a statically typed language, I realized the power of using TypeScript instead of JavaScript. I should have learnt and used TypeScript instead to make the code future proof and bug free.

I managed to implement all views that I stated in my mid assignment submission except one which is the calendar view. This view was extremely challenging to implement, and I was not able to implement a calendar view that works well by the project deadline. However, I will continue to work on this project after the deadline to introduce this view because it will greatly help in organizing todos which is the core philosophy of Just To-do It.

A lot of other goals that I set out for myself in the mid assignment submission such as having an email verification system, third party authentication, notifications were all not implemented. I am extremely disappointed in myself because I was not able to implement these features because these are features that should be present in a well-made todo manager. I was extremely productive during the winter break as I had enough time to learn and implement features but when semester 2 started and work started to pile up, I failed to dedicate enough time to this project which meant that these important features are yet to be implemented which I deeply regret.

I also did not have enough time to explore the usage of docker and cron jobs in my application. I had an idea of using cron jobs to automatically have completed todos deleted after a certain timeframe that will be set by the user in their preferences, but I was unable to implement this feature. I also did not learn as much as I should about docker.

Summary:

This project introduced me to web development which helped me greatly. I also got to experience something that I would likely be doing in the future in whatever company I might be working in. I think the most important takeaway of this whole project was learning how to learn quickly and effectively. This is due to size of things you had to learn just to start the project and the limited timeframe to complete it. Once again, I would like to express my appreciation to the CVWO team for this opportunity. Ultimately, it does not matter if I get into CVWO this year. What matters, however, is that I had a great time working on this project and the fact that I was able to learn a lot while doing so.