Ritesh Kumar CV Final Submission A0201829H

I'm submitting incomplete work and that's quite shameful. I have seen the good work some of my friends and other candidates have managed to produce and I'm amazed by the possibilities. I'll spend the rest of this document reflecting on what I have learned through this assignment and on how I could have done it differently (hindsight is always 20/20, isn't it).

What my work can do:

- CRUD actions on tasks, with a side-bar that is a bulletin-style list that sorts the the tasks and has a search bar to search via task titles.
- A tags list that shows all the existing tags, clicking upon which aggregates the tasks that fall under that particular tag. I'm afraid I didn't manage to make the user be able to create tags via the new-task form, nor implement user-accounts.

This assignment was supposed to be my first foray into self-learning something, having written my first hello World in CS1101S. Found The Odin Project and decided to follow it because I wanted to have a proper course structure that I could follow, I've documented everything on my Odin Project Repository (notes, tutorials I've done and small projects in Odin Project).

Noteworthy things for which I can be proud of:

- I spent about a week on learning proper JavaScript and doing some tutorials, learned about how the internet works in the first place and learned to use vanilla JavaScript to manipulate HTML DOM elements. Vanilla JavaScript is really powerful compared to Source and doing the tutorials helped connect some dots from Semester 1 and improve my confidence in writing programs.
- I'm now confident in my cli-fu and can help my friends on our schoolwork thanks to that. I got exposed to the Open Source community (the etiquette, the healthy learning culture...).
- I learned how to version control (at a basic level) and it's a godsend (though I'm using it more as a drive since I'm not too skilled at it yet).
- I spent about 2 weeks learning Ruby. Was mind-blown by some of its syntactic sugar (erg. spaceship operator, ruby crops, mixing and modules and the like). I learn OOP principles through the tutorials I had done when learning Ruby. This has made me appreciate learning CS2030' design concepts so much more. Odin Project focuses on Test-Driven Development a lot and this seems to be a good practice, since CS2040S relies on test-writing too.
- I spent another 2 weeks learning Rails, the MIC model and how a website actually works. Cherokee and Deployment was covered.
- I'm comfortable enough with APIs, serialization and Rails servers to be able to use it for other projects (friends and I are working on a Telegram bot that works with a Rails back-end). I should be able to understand Databases when I need to work with them in the future.

- I learned how to read Documentation which is probably what people spend most of their time on when working on a project.
- Lastly, I wish I had started it sooner, but I spent about a week learning ReactJS. I've not understood it properly though, which is why my project didn't get completed.

How I could have done it better:

- Should have started learning React earlier. As far as I know, there are 3 different ways to create a Rails-React webapp and two of them involve using specific Ruby gems while the third one involves setting up Rails as a JSON API end-point. I chose to do the third one since (as per my understanding on serialization) it would prove to be useful in other situations (e.g. Telegram bot-making). I couldn't clearly understand the design practices to be followed when writing the React front-end. Currently, I have mimicked what I had learnt when using Rails views: I have multiple React Components that get rendered upon clicking of links. The HTTP request is all done via a common component that keeps things in state and passed the relevant props to whatever component is rendered (note the URL changes when interacting with my implements). The correct way to do it would have been to have the routing all set up in App.js (the root component) and keep rendering child components based on what's captured in state. A lack of proper design meant that I faced unnecessary bugs since I was trying avoid using boiler-plates for my React components in the hopes of learning by making mistakes.
- I should have just looked for boiler-plates for the standard react components instead. A todos app is supposed to be a basic/straightforward app to practice what's learnt, and so I thought I could write majority of it myself, but that resulted in dead-ends and bugs that I could not fix.
    - Bug 1: Since no tags can be created yet, when submitting a new task form, a helper function that I wrote to aggregate tags for that task will return a null error and show an error, the task will still be created though, just that the redirect fails.
    - Bug 2: I managed to create helpers that will package the tag-field of the form into the correct JSON object and wrap it nicely for the axios command. The axios command works but the back-end can't seem to create that nested attribute for the task object.
- I did ask for advice from my friends: Jeremy Tan (my TA for CS1231S last semester) and Tan Rui Xuan (we helped each other out) who told me how I could have made some improvements (e.g. properly routing things for React router) but I asked a little too late. Nevertheless I'm grateful for their input.

In general, better code design and not insisting on writing things myself to encounter bugs would have helped me produce the basic deliverables. I'm still grateful for attempting this assignment. It has given me a lot more confidence in writing programs and getting accustomed to failing and troubleshooting things. I won't abandon this project, will finish it after I get my other side-project done. Thank you CVWO team, for the assignment and I apologize for the incomplete work.