Name: Tham Rui Cong (A0234692Y)

CVWO Final Assignment Writeup

Self-Reflection

It has been a painful yet satisfying one-and-a-half months of creating and debugging my own personal task management application.

Painful in the sense that with my little to no prior knowledge on web development, it was daunting and overwhelming when faced with all the new terms such as API, Frontend and Backend. Furthermore, each language (for front and backend) will have different syntaxes and conventions, which makes it even more confusing and convoluted. Thankfully, the multitude of tutorials and guides online made it easier to transition into the basics of ReactJS and Ruby on Rails. Many manhours were spent scouring Stack Overflow for debugging issues and (hopefully) resolving it to the best of my abilities.

Yet, the time spent was also a satisfying one. By challenging myself to complete this assignment, I have allowed myself to broaden my skillset (through learning new languages). ReactJS and Ruby on Rails taught me how complex and multi-dimensional simple applications are. The Model-View-Controller (MVC) framework opened my eyes to a totally foreign method of operation, where the logic of the application is kept apart from the database and the client display.

I am still not quite yet satisfied with my application, as I feel that it is albeit underwhelming and aesthetically unglamourous yet, and I feel that many more functionalities can be added to further elevate the application (such as deploying on Heroku). However, I feel that it is still an abundant showcase of the culmination of my hard work and knowledge that I have invested in these short time span.

Things I have learnt

ReactJS

I was more comfortable with the frontend design, from the initial installation of dependencies using **npm** and **yarn**, to using **react-router** to display different pages based on the URL. The part that was the most fluid for me was the class component states and methods. The idea of "lifting the state up" was interesting, and I used the various class component (which I represented as pages of the app) to store important data that would be subsequently passed to its children components. The class methods act to manipulate these data in various ways and was mainly called upon interaction with the view. This was where JavaScript (through React) and HTML (through the view) mainly interacted to manipulate the DOM.

Ruby on Rails

This was a more foreign language to me as the syntax was different from what I was used to. By using models, controllers, views and routes, I was able to set up the backend with database. Models interacted mainly with the database, where I used

associations and **validations** to configure the data in the database. Also, controllers allowed for basic CRUD functionality and renders the data in json format for easy access of data. Routing allowed for HTTP requests for the controller methods. Also, the implementation of React on Ruby on Rails was done using a webpacker, where I used **javascript_pack_tag** to render my ReactJS frontend.

Axios

Axios was responsible for communicating between the front and backend. By using **GET** requests, I was able to retrieve data from the backend and display it on the frontend seamlessly using the **componentDidMount** special method. Also, the POST requests are responsible for creating new tasks and tags, the **PUT** requests altered the contents of existing tasks and the **DELETE** requests removed unnecessary tasks and tags.

User Manual

To view the app on your local machine, clone the repository from https://github.com/thamruicong/CVWO_Project and run ***rails server*** on Ubuntu terminal to start the server.

The application will be displayed on http://localhost:3000/ .

For the creation of tasks, task titles are compulsory and must contain at least 5 and at most 30 characters. Also, each task must be unique and belong to an associated tag. Other fields like date and time are optional upon creation, but in leaving them empty, these fields will be stored with an empty value in the database.

Tags are permitted to have a name of maximum length of 20 characters. Furthermore, user cannot create tags of the same name as a currently existing tag.

Additional details can be found on README.md of the source code as well as the homepage of the actual application.