# CVWO Mid-Assignment AY22/23 Write-up

**Name:** Chan Yong Quan Joseph

**Matriculation Number:** A0249075X

Link to deployed application: https://blog-forum-app-cyqjoseph.netlify.app/login

---

## Introduction

Having some experience in frontend development and a little experience in managing NoSQL databases such as MongoDB with some pet projects, I decided to take up this assignment over the winter break to help refresh my knowledge of the frameworks I worked with previously, as well as pick up some new frameworks along the way. Needless to say, I have learnt a lot while creating this project, and my biggest takeaway from this is appreciating the connection between the frontend and the backend.

The project is a simple blog/forum with user authentication and features basic CRUD functionalities on users, blogs and comments. The frontend is built using React and currently deployed on Netlify; the backend is built using Rails and is currently deployed on Heroku. I decided to store my frontend and backend code in a single monorepo instead of separate repositories.

---

## Frontend

The frontend is written in Typescript and built using React. I gained a deeper appreciation of type safety since completing CS2030S hence Typescript was preferred over traditional Javascript. For styling, I decided to go with bootstrap for my various components for its simplicity. I decided to integrate react-router-dom for frontend navigation, and redux as my state to store current user data as well as errors.

## Difficulties faced

Overall, the challenges faced from the frontend of this project was not as much as the backend, as the heavy lifting of user/blog management was handled by Rails. However, some problems faced was with the design of components, such as getting the flex properties to work as intended. In addition, I required a refresher on Redux as its usage with action creators and state were definitely more complicated, thankfully the documentation was very helpful in clearing any issues I had.

## Improvements

Certain improvements of this project would include a touch-up of the overall design of my project, as many components are built with simple bootstrap designs provided out of the box and using the bootstrap default colors. I also plan on using Redux to handle state management for the likes and dislikes of the blogs in the

future, as they are currently static. Most importantly is refactoring and reorganization of certain components, as I feel there are lots of boilerplate code in the form of JSX and function handlers which can be removed.

---

## Backend

I decided to use Rails for my backend after going through some tutorials and videos to familiarize myself with the framework. Nevertheless, I still found the Rails documentation the most useful in my understanding of the MVC architecture after running through the Rails guide. Though I have no prior experience in Ruby, its syntax made it relatively simple to understand and pick up on the fly. The project consists of 3 models, the User, Blog and Comment, where a User would have many Blogs, and a Blog would have many Comments. In addition, every Comment would have its own unique "commenterId" attribute, which references its unique User.

## Difficulties

The initial phase of generating the backend was problematic, as much configuration had to be done to link the react application with the rails application which admittedly required of copy-and-pasting of code online such as the "cors.rb" and the "session_store.rb" files to get the project to work as intended. Following that, problems arose during the association and migration of the models, which involved a lot of debugging and correcting of my models.
The tagging system posed a problem as I had already generated the Blog models and I was contemplating whether to include tags as an additional attribute in the Blog model. In the end I decided to use the acts_as_taggable_on plugin which saved me the trouble from re-inventing the wheel, which helped me smoothly integrate the tagging functionality without writing much code myself.

## Improvements

Currently user validation is not that well formulated, duplicate usernames are not allowed and password hashing is implemented. An improvement could include passwords being of a certain length or email inputs being valid. I have yet to consider whether to put this validation logic in the frontend or the backend. In addition, the tagging system is very rudimentary, only exact matches of the tag would yield a result, fuzzy matching has not been implemented. A more advanced search system can be implemented. Furthermore, a Comment model is indirectly related to a User who commented it, I resolved this by making the Comment model contain a commeterId attribute which matches to the correct user. I am unsure if this is the best practice to link both the Comment and the User, or whether I should let rails to the linking between the two models internally. Lastly, I have yet to fix the issue of user inactivity after deploying the application on Heroku, as the user would be signed out after being inactive for a short period which will cause the frontend to render incorrectly.