Before embarking on this assignment, I actually had no experience in web development at all. Thus, I realized that trying out for CVWO would not only be a valuable way for me to learn web development, but also a great opportunity for me to potentially contribute to an organization that is creating a positive impact in the world. This was what encouraged me to participate.

I started off by learning the basics of React with YouTube tutorials, including the use of basic hooks such as useState and useEffect. After this series of tutorials, I ended up making a basic CRUD forum that was deployed on firebase. Since firebase took care of the authentication and database work, I mostly learnt only about the basics of frontend design.

After feeling like I know enough to continue, I decided to start on my project in Ruby on Rails. This was when I realized I had no idea how to get the backend working. I then proceeded to learn some basic database design off of even more YouTube videos, learning about the basics of one-to-many relationships and their associated foreign key entries in databases. This allowed me to create my User, Post, and Comment models, of which my first attempt failed causing me to have to git revert and remake the new models.

Next up, I proceeded to learn about Redux and their concept of slices and reducers. I realized that I probably had to learn this concept well if I did not want my project to become a mess of global states with props being passed everywhere. This was however rather tough at the beginning since having to debug the reducers together with async API calls were rather difficult, and I did not really understand why my state was not updating.

After getting Redux to somewhat work, which allowed me to use dispatches to do API calls which updated the global state, I finally had the ingredients to start completing my forum. This was also a rather daunting task as I felt that I have been a little too ambitious in my project requirements, with many features interconnecting with one another. I also realized I did not really have the software engineering experience to do a project with more than a thousand lines of code, with my own projects usually being much smaller scale. In addition, I had already spent quite a few weeks doing tutorials and learning about the aforementioned features in general, leaving not that much time before the deadline approaches.

I also faced some difficulty getting the useEffect hooks to trigger properly and in the right order, such as fetching the comments to a post on refresh, which then called a method to count the number of comments in the specified thread itself. I eventually managed to solve this by having the post controller perform a join query with the

comment database and directly returning the count. This was also a difficult process as I did not have any SQL background before this, and had to wrestle with Rails to get it to properly send my query and respond to my requests.

I also decided to do some query parsing on the backend for the post and index fetch methods, as I wanted to be able to specifically search for posts by a certain id or posts made by a specific user, or comments under a specific post.

For authentication, I decided to use JWT to create a token for each user when they sign up. This token is then stored as a cookie in the browser, and automatically sets the authentication state even when the user refreshes or leaves the web page and comes back. This token is used to authenticate any create, update, or delete requests. I had a vision of a forum somewhat like Reddit when I started, thus I decided not to have the user log in to be able to see posts or comments.

On the styling side, I decided to use Bootstrap for styling as it felt rather convenient for this project, with its already configured settings for the many possible features I could have wanted. I also felt that it was very important for the UI to be intuitive for the user, and so I experimented with many different ways to help improve the user experience. In the end, I wanted to make posts in the home page and both posts and comments in the profile page clickable, and thus I decided to add some drop shadow transitions to highlight this to the user. I also added dropdown menus that were only visible to the owners of that specific post or comment, with options to edit and delete it. I also added a display for users to see how long ago a post or comment has been created, as well as if it has been updated.

Near the end, I also decided to implement a search system, so as to simplify the searching of posts and not making the user manually do a ctrl F command to find what they want to find. I accomplished this by adding a search bar, tokenizing the search and going through the stored posts state to count the number of matches for these tokens. I then created a new Redux state for the sorted and ranked results of the search.

User Guide

To initialize the database

```
cd cvwo-assignment
rails db:create
rails db:migrate
```

To start the backend server

```
cd cvwo-assignment
rails s
```

To start the frontend website

```
cd cvwo-assignment/frontend
npm run start
```

Using the website

The default home page contains a list of all posts. The user can then search for specific keywords to narrow down the posts, or click on any post to go to the thread page for that post.

In the thread page, the user can see the post itself, as well as any comments made under the post.

If the user is not logged in, they can only view posts and comments.

The user can click on the login page link in the navbar to be redirected there. The user can then create a new account by clicking on another link which redirects them to the sign up page. When the user successfully signs up, they are automatically logged in and redirected.

Once logged in, a user has access to the create post page, and can also add comments under posts. The user can see all their previous posts and comments in their profile page, and go to these posts and comments directly by clicking on them.