Final Submission

Preface

Prior to this assignment, I had no knowledge whatsoever of web development. This assignment has been quite the challenge, especially when I was first starting out. Because the entire software stack was new to me, I spent many hours fumbling around. It was only about halfway through the project did everything more or less click.

After having done this assignment, I now have a good understanding of the *React* and *Rails* frameworks. Despite starting this assignment quite late in end December, I've committed a lot of time towards making the final product one which I can envision myself using.

My application can be found at https://todo.ianyong.com/. Due to the page limit of three pages, the user manual below will not go into specifics. Instead, I've created a sample account with seeded data for demonstration:

Email – cvwo.assignment@gmail.com
Password – password

User Manual

Upon loading the application, the user is greeted with the authentication screen. Here, the user can either login using their email and password, or if necessary, create an account.

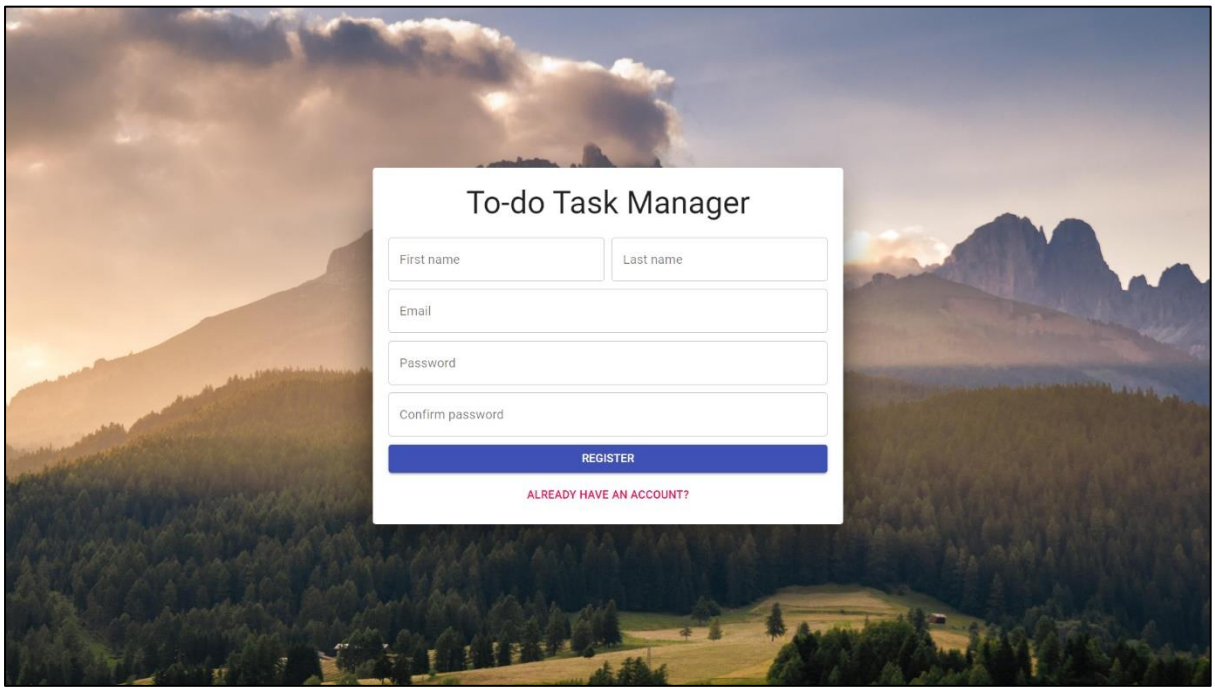

Figure 1: Registration Screen

Once logged in, the user is brought to the task list. The task list displays every upcoming task, as well as tasks which are overdue, sorted by date. At the top left corner of the page, the user

can press a button to expand the navigation drawer. Here, the user can perform the following actions:

- Filter by three pre-set date ranges (today, next 7 days, and all)
- Toggle between showing and hiding past completed tasks
- Filter by tags
- Edit account details
- Edit password
- Log out

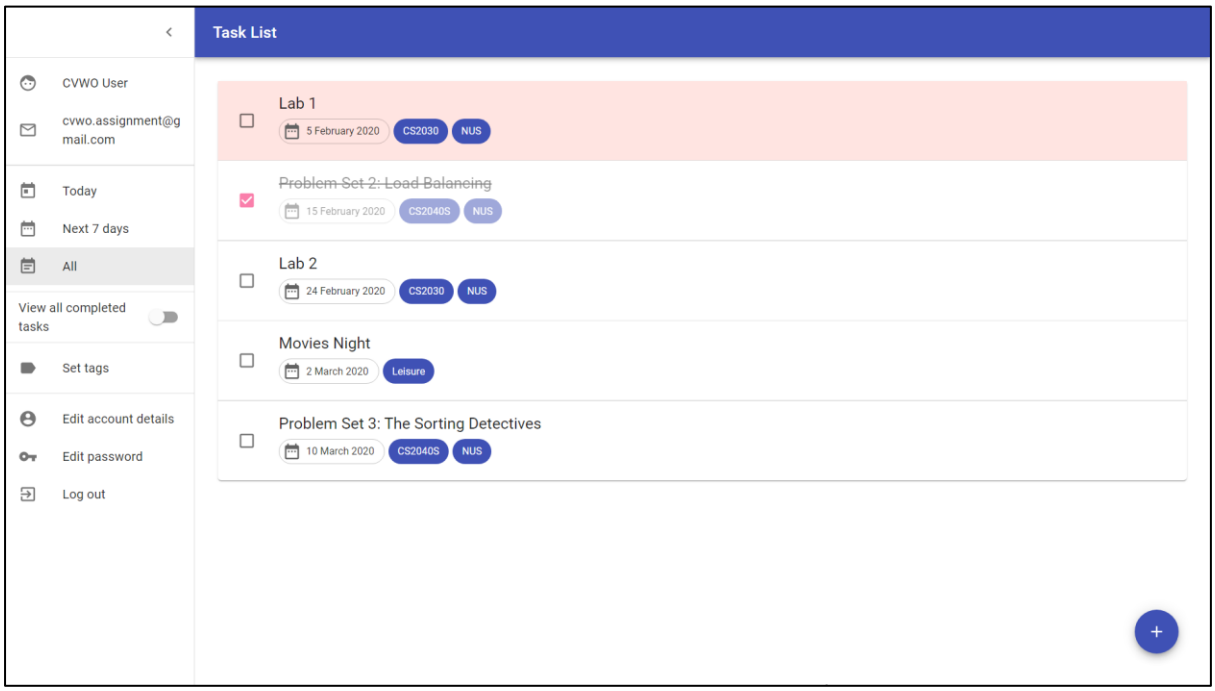

Figure 2: Task List with Navigation Drawer open

Tasks which are overdue are highlighted with a reddish tint. New tasks can be added through the floating action button at the bottom right of the screen. The user can click on a task to view more details, edit it, or delete it. When a task is completed, the user can click on its checkbox to mark the task as done.

Authentication System

The application uses JSON web tokens (JWT) for authentication. This eliminates the need for the server to keep track of user sessions. For security, the client is given a JWT that is valid for just an hour. Every single request other than logging in and registering requires a valid JWT to be present in the request header. Should the JWT not exist or be expired, the user will be automatically sent to the login page to login again.

Tagging System

Just like tasks, tags are user specific. Each task can have an unlimited number of tags. The *Rails* Active Record query interface makes the manipulation of data in the database fairly simple. In the application, tags are a standalone resource. Tags are related to tasks through taggings. This design of the database allows for the scalability present in the tagging system.

When filtering tasks by tags, the querying is that of an OR operation; the filtered tasks have at least one of the specified tags.

Material UI

The user interface was designed in adherence to Google's Material Design. This was achieved through the use of the *Material-UI* library. The reason I chose to go with Material Design is because I have some experience in Android application development, and thus am already familiar with the design guidelines.

Hosting & Docker

As suggested, the hosting of the application is on *Heroku*. Using *Heroku* has been a very pleasant experience. *Heroku* has great support for *Docker* images. Though it took me quite some time to get the configuration right, I am now able to deploy my application to *Heroku* simply by pushing to it. *Heroku* then builds my application in a *Docker* container before deployment. This has greatly sped up the deployment process as building the *Docker* image locally is far slower than having *Heroku* build it.

Custom Subdomain

I've always found it cool when I see seniors of mine with their own custom domain names. This assignment was the perfect excuse to finally get one of my own. Again, *Heroku* has great support for custom domain names. All it took to get my custom subdomain working was to add a simple DNS forwarding rule in the control panel of my domain registrar, pointing towards *Heroku*'s DNS.

Room for Improvement

Given my lack of prior experience, this point is inevitable. One major room for improvement would be the use of *Redux*. When I first started, I did take a look at *Redux*. However, I wasn't sure of how it would help in my application. On hindsight, using *Redux* would have saved me a lot of headache. One major issue I faced in the development of the application, was the synchronisation of state across components. In the spirit of modularity, my front-end is split across many *React* components. In order to "talk" across components, I have resorted to chain passing props. This is very clearly not ideal as it increases the complexity of the code and makes it much harder to maintain. While I have managed to make the application work just fine without *Redux*, I imagine this would very quickly become a nightmare if the application were on a larger scale.