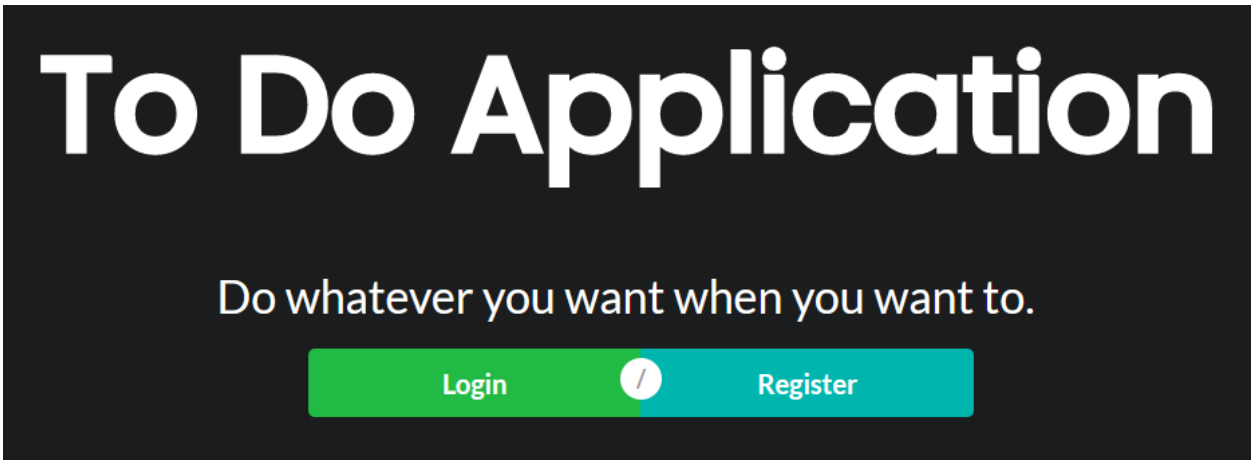
CVWO Final Submission



Introduction

I took a full 2 months to complete the assignment. I have no prior experience with any of the languages or libraries used beforehand. Coding the actual application took far less time than the 2 months, but due to my inexperience, I was unaware of many things that could have sped the process up.

For example, I initially used React libraries that were not popular or maintained, and eventually had to re-do certain components when there were certain unfixable bugs. Furthermore, facing certain bugs when self-learning without having anyone to consult added multiple days to the project.

However, it was struggling for 2 months that I have learnt much, had fun, built an application I'm proud of and have a relatively good understanding of most of the technologies used.

For this assignment, I have completed all the requirements and optional.

Github Repository: https://github.com/FergusMok/ToDoApp

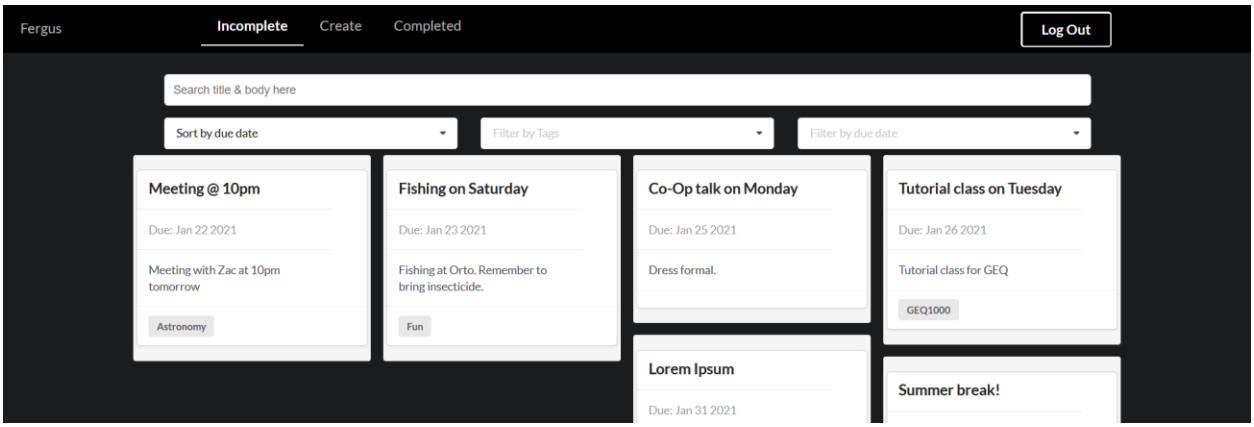
React: https://fergus-cvwo.netlify.app/

Rails: https://warm-escarpment-17665.herokuapp.com/

Requirements

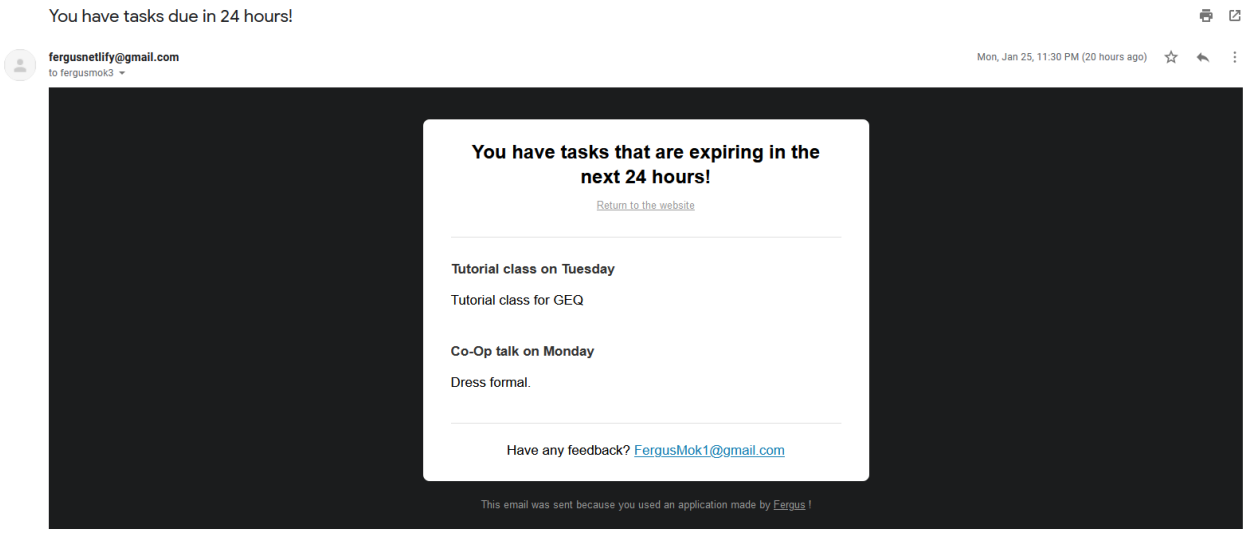
The application has a login system that operates based on cookies.
Once within the application, users are able to sort their tasks based on a “due date” and “update” criteria. Furthermore, they can filter based on tags, and a range of how long the due-date is away. Expired tasks are automatically added too for this filter.
In addition, there is an incomplete and complete page, which they can review old tasks.

For the user experience, I tried making the interface as friendly as possible, and an application that would be fun to use.
The grid system is animated, and users can mouse over to quickly edit or mark as complete. Furthermore, each task is compiled into a card, and users can quickly view many tasks at once. I’ve also included loading spinners while fetching and loading. Furthermore, all non-existent URLs are automatically caught to a PageNotFound page, and unauthorized redirection will automatically be redirected to the Login page. Also, the front page uses SVG animation.



Cron

For scheduling, I have implemented an email system sent at 11.30pm, that warns users whose tasks are due in the next 24 hours. Implementation was done by Heroku Scheduler instead, due to the hosting dyno switching off when not in use. If not hosted on Heroku, the rails application is able to utilize its implemented “whenever” gem.



Typescript

I've implemented to the strictest configuration, with "alwaysStrict" toggled to true.

RESTful APIs & Hosting

I chose to have 2 completely separated applications, with Rails hosted on Heroku and React on Netlify. This made the user-login implementation harder, but it paid off in the end. Scalability is/will be much easy.

Redux

Redux was implemented on React for certain states that was cross-component. For states that only existed within individual components, I stuck to normal state hooks.

Docker

The applications also come with simple docker-compose.yml files. This makes the building process much simple.

Additional Add-ons

Pinging

I was frustrated with Heroku's idling dyno, where initial logging in can take up to 10 seconds. I've since set it to ping every 5 minutes to keep it awake 24/7.

Improvements:

Security is an issue, as it is easy to simply post a request to Heroku for information.

I would also work on the implementation of Docker, as I am sure more can be done in that area.

Furthermore, I tried implementing registration via other platforms (e.g Github), but I eventually ran out of time.