CVWO Final Write-Up

Loh Jian Rong (A0252735A)
Working Application: https://d3mj3t330xelda.cloudfront.net

Notable Features

- Uses **OpenAI** (ChatGPT's AI model) to generate post content from title
- URL has **query parameters** for easy sharing
- **Rich text editor** support to edit/comment posts/comments
- **Search bar** for both tags and text
- **Sorting** system for posts and comments
- **Drag-and-drop picture from local machine** for image upload to S3
- **Toast notifications** for success and error messages
- **Responsive** layout
- **Account-based authentication** using JSON Web Tokens
- **Cron job** on EC2 to automatically backup PostgreSQL database
- **TypeScript** usage to improve maintainability and scalability
- **Redux** for global state management
- Application is **dockerized** and **deployed to AWS**

Reflections

Having built small-scale full-stack projects before, I initially thought this project would not pose much of a challenge to me. After completing the project, I believe that I was wrong and I am glad to have completed this project in a span of about 3 weeks. Going into this project, I wanted to maximise my learning as much as possible and pick up as many new technologies as possible to challenge myself.

Frontend

I previously had experience working with React and Redux at 99.co, however, I have only used it with JavaScript and never had to handle TypeScript at a larger scale. Despite having built small projects with TypeScript, there was definitely a learning curve to using TypeScript with React. It was very tempting to replace my types with `any` and move on to create the feature. However, I often have to come back and figure out the type. By then, I have lost the context and I have to remember what I did at that time. This leads to worse productivity and lost time. Hence, I gradually learnt to work with types as I developed the features. TypeScript has been helpful to the development process and the autocomplete features definitely sped up my development process.

Backend

For the backend, most of my experience came from working with Express.js and Python's FastAPI. With my knowledge of building a RESTful API with the MVC architecture, I wanted to pick up a statically typed, compiled language like Golang instead of Ruby. I did not have much experience designing a good database schema, so I had to reset my database a few times during development before I settled on one that was suitable and could work for my application. This made me realise the

importance of good planning. I think the database diagram I drew helped me with finalising my decision.

Deployment

The biggest challenge I faced was deploying on AWS. Even deploying the frontend on AWS was overwhelming because there were so many options to do so. Since I wanted to build the app in a Docker image and deploy the static files to S3. I decided to use the AWS CLI to push it to S3. Then, I used Cloudfront as a CDN for additional speed and HTTPS support.

Deploying the backend was more complicated. Firstly, I knew I needed a docker-compose file as I needed to dockerize both the application and the database. Setting that up was still okay, however, I had a lot of trouble configuring the ports on the EC2 instance. Even after managing to access the application through HTTP, I wanted the application to have HTTPS support. Many of the online demos made use of a custom domain, which I do not have and I did not want to purchase one. So, I found a workaround using Caddy. This allowed me to set up the EC2 server with automatic HTTPS support. Since I did not have a domain, I used nip.io as a DNS for the EC2 IP address.

Throughout development, I tried to keep my code as organised as possible. Keeping the abstraction barriers up, making sure that each layer interacts with each other independently. Outside of the database diagram, I tried to keep a Postman collection as well as keeping the READMEs in my repositories organised and easy-to-read.

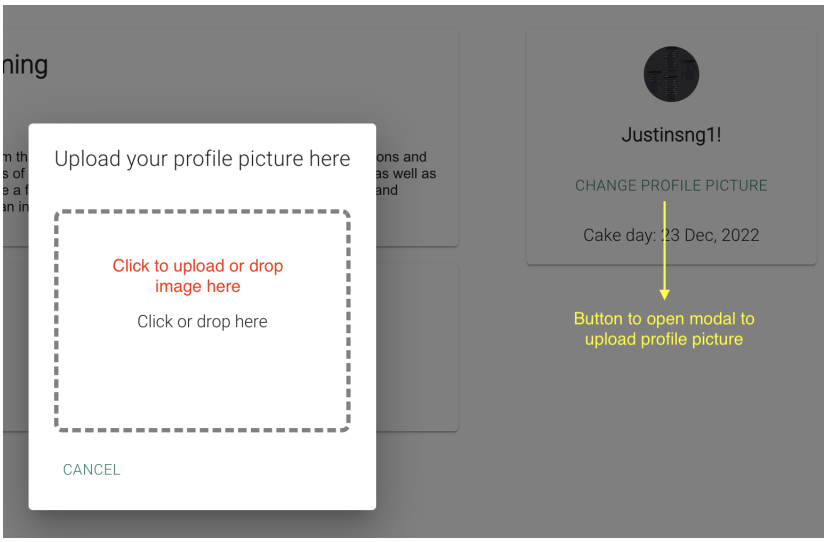
Generally, this assignment has been a great learning experience for me. I believed I managed the code quality of this codebase decently well and I made sure to document my code as much as possible. I also took a lot of inspiration on code structure from my CCA's codebase. For example, the idea of having a service layer in the frontend was taken from that codebase (Google Developer Student Club's Diabetes Singapore Project). Developing this application has been fulfilling, and I hope to be able to bring my skills to do more good for the public through participating in CVWO.

Some future improvements could include having a better app design and writing tests for my codebase. I am not the best designer and I made full use of MUI's components. Also, I developed this application as quickly as I could and this came at the cost of stability as I did not develop tests for my features. Lastly, I could also build a deployment pipeline which triggers every time a branch is merged to `master`. However, since this project is a solo project, I felt that the need for that is little and the effort used to set up that pipeline was probably better spent elsewhere.

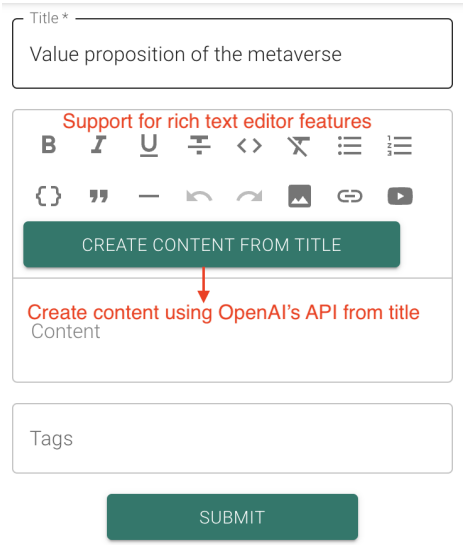
User Manual (Imgur Album for more details)

I have only included pictures for features which may be slightly more confusing. View the image album above for more comprehensive descriptions.

User page



Post submit page



Home page