CVWO Assignment Final Writeup

Deployed version: https://frontend-2w5qhaldaq-as.a.run.app/
User guide: In the same Github repo

Dear CVWO team,

I am really glad to have this opportunity to be able to grow my development skills. I hope to be able to learn from all of you and grow my web development abilities. I will be documenting some of my learning here, thus the sometimes informal writing. I have also created a User Guide, which is more formal, and will cover how to use the entire webapp. (but I think it's already pretty user-friendly :))

The link to my mid-assignment writeup can be found here(my submission did not go thorugh previously). A little fun fact: I only found out about the assignment on the 30th of last December as I had missed the email blast. However, with the experience I have had with web programming, my interest-driven motiviation led me to learn at a quick pace through online resources. Now, I am able to autonomously implement features such as *post liking*. Although I almost missed such a precious opportunity, this shorter timeframe really strengthened my time management skills, on top of learning and development skills. You can expect such dedication, learning and quality during the internship. (+*reading my emails more proactively*)

I finished the learning process in Week 1 of sem 2, and had finished the Minimum Viable Product (MVP) by Week 2. Afterwards, I tidied my code style and structure, and implemented some final touches. The final feature I implemented was *tagging*, which I manage to complete seamlessly (*it's quite amazing how much improvement can be made in a week*).

My 'mid'-assignment writeup contained the foundation of my tech stack, with the following changes:

- Instead of SQLite, I used SQL
- Much more features than listed

1. Tech stack

Backend
$Golang$ - $Go\ Fiber\ v2$ framework for routing etc., and $GORM$ to connect to a $MySQL$ database
Other packages: $bcrypt$ for hashing, and $jwt\text{-}go$ by `dgrijalva`

RESTful API is implemented i.e. each URL points to a single page, which has its own purpose. Requests & responses are handled using `JSON`

Frontend
$React$ - $Typescript$ ($tsx - typescript\ execute$, $ts - pure\ TS$)
(I used a lot of react hooks, instead of classes. However, I will like to learn more about them, and apply a wider range of methods in the internship.)
$Axios$ to request from backend via `AJAX` calls
Other plugins: $Remarkable$ for markdown-support, $Bootstrap$

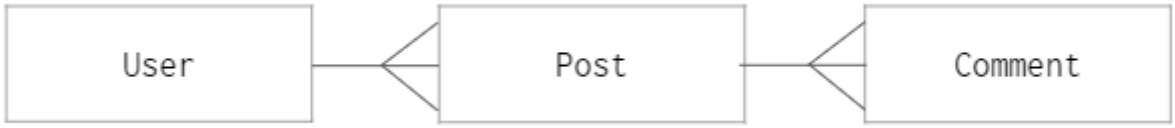
MVC

- Controllers are found in the backend
- Views are listed under `pages` in frontend
- Both front and backend has models to facilitate

Database

- $MySQL$ - Since I have experience writing $SQLite$ queries, I decided to use this (it was interesting to see how foreign keys work in $GO/GORM$ as it is not as straightforward as in $SQLite$. For instance, you can include a slice of comments `[]Comment` within a `Post` instance, then state the foreign key via `Gorm`. Another e.g.: You can have a `User` object within a `Post` object, along with a `UserId` attribute. This `UserId` attribute references the `id` of the `User` class.).

- However, I did not normalise my tables to `3NF` (i.e. $\exists$ non-key dependencies) as I was focused on learning, functionality and features. In subsequent projects, I will fully apply my knowledge on databases properly. I hope to be able to apply concepts such as ER diagrams too.



- The database will store all User, Post, Comment information.

> ⚠ **Note**
>
> You are required to preload nested objects using `.Preload` (when using `GORM`) for its contents to show. This is known as "Eager loading". Also, unsigned integers (`uint`) are used for `id` to lower memory usage.

Testing
Backend was testing using *POSTMAN*, while the frontend was tested using the *MVP*. I have also allowed real-users to test out the deployed app.

2 User requirements

Introduction
My forum is targeted towards NUS students, for the purpose of **sharing information on their CCAs and experiences**. Right now, information for CCAs are only largely available during Student Life Fair (SLF) and through word of mouth. This can be a monitored software where CCAs can officially host events. Althought other forum platforms like r/nus exists, this will serve as a centralised forum, solely for the purpose of CCAs. As the actual development time I allocated for myself is 1.5 weeks, I have yet to implement CCA-targeted features (i.e. the current product can be repurposed for anything). You can expect more features to be rolled out over the semester, as I prepare for CVWO.

Existing Features

- Account-based authentication (made possible using JSON Web Token (*JWT*) and *BCrypt* hashing)
  - Registration & Login*
  - Security Questions
  - Reset Password
- Restriction of certain webpages / functions for unauthorised users (Information Security)
- CRUD features for forums posts & comments + Liking
  - Create - Users are able to easily create posts from just about anywhere. commenting has also been made easy.
  - Read - Users are able to view *all* posts *or* their *own* posts, with the option of *sorting* posts based on likes & timestamp. **Each post also has its own post page.** Comments are displayed to users in a manner that reduces clutter. **You may also search for posts by their tag.**
  - Update - Users can easily edit posts & comments (from posts view or post page view). Profile can be updated too!
  - Delete - Delete function for posts & comments exist in the control panels for every post and comment! *(now you don't need to worry about accidental typos)*
  - Liking - This was one of my favourite features to implement. Likes show how popular a post is. *(This was challenging to implement. For instance, the buffer time due to updating the database on a like can cause errors.*

E.g. spamming the like button can cause the count to skyrocket. Therefore, I implemented a delay between each potential button click. This was effective, but I soon found out that like count starting at 0 caused issues. Thus, every post is initialised with 1 like, and the like count is decremented in my React .tsx files.)

- Markdown is also supported. (*hence, images can be posted*)

> **Future**
>
> A future feature I want to potentially add is a title field for each post, which can be implemented easily. (add a title field to the Post class, add an extra input field and display accordingly using React)

- Other features: Top Post view, Back to Top button, other UI features

3 Main Features & Techniques

I have documented *some* of my learning. I spent the bulk of the past couple weeks mainly programming the final product and deploying, so here are the bits and pieces of my thoughts on submission day :)

Fiber Framework

- (most) Functions in the backend will begin with `func <funcName>(c *fiber.Ctx*) error { ... }`. `Ctx` refers to context and it has many methods/
  - `c.BodyParser` is used to extract `JSON` body input
  - `c.JSON` allows objects to be transofmred into `JSON`
  - `c.Cookie` creates a cookie

JSON Inputs

- `c.BodyParser(&input)` is extracting data from the `JSON` body of `POST/PUT`
- input has the data `map[string]string` which stores key-value pairs i.e. `map[<keyDataType>]<valueDataType>`
- `BodyParser` will return `nil` if succesful. Otherwise, the `err` will be returned.
- The value of a key-vale pair from data can be accessed as `data["<key>"]`

useState, useEffect

- These two methods must be two of the most important ones in the entire project. `useState` helps create a variable of state that can be used to update the UI. It comes conveniently alongside another method beginning with `set`. For instance, I can use `const [content, setContent] = useState('')` to capture input from the user via an input box `<input type="text" onChange={e=>setContent(e.target.value)}>`.

Params

- URL paramteres / queries were a huge part of my routing logic. See `README.txt` under the `posts` folder in the React frontend folder to see the logic.
- They helped me redirect to the correct previous page, get information of what to display.
- With all this, they should still abide by RESTful

I would love to document my whole project once I get the chance to.

All in all, this pre-CVWO assignment has been a really fulfilling one. It gave me rigorous and independent training in full stack development. I will carry on honing these skills through the internship and self-learning. I hope to contribute to the CVWO effort greatly during my time there, and I hope that I can also use my teaching skills to help mentor the batches after me.

Thank you.