CVWO Final Write-Up

Firzan Armani Bin Fajar Masazman
A0184164A

Reflections

This has been an eye-opening experience for me as well as a personal challenge. I haven't dove deep into web development especially in back-end. Rails as a framework is actually, in hindsight, a very efficient way to get a idea into a prototype relatively efficiently and simply without too much challenging code being in the way of getting a basic idea out into a product. However, what I've also come to realise is that combining Ruby logic into the web page code has so much power to link both the back end into the front end with minimal code.

I've also become very familiar with the concept of Model-View-Controller framework. Developing with the framework in mind actually makes thinking about how to setup the database, simpler. It enables easier planning; a simple mind map or process drawing actually suffices to have a good idea on how to implement into the MVC framework and eventually into the creation of the back-end of the site. Inevitably, I've also learned what CRUD is. There are many tutorials on CRUD with many types of examples and applications, like blogs, social media, note-taking, chat applications. Reading through these examples show that Create-Read-Update-Destroy is actually everywhere when it comes to databases.

From that, I've learned how SQL databases work. At this point, I'm still not very proficient with SQL queries and such, but working with Rails and its MVC approach has exposed me to the concept and operations of SQL and how it works together with the controller to create and update the data. It's made me come to appreciate all the simple data operations that make up a full-blown website, to large scale ones like Facebook or Instagram with their use of posts.

Next, I've come to understand the use of RubyGems and in turn the internal structure of Rails. At first, I was adamant on relying on it as little as possible, but as time went on and I became more determined to improve the UI with minimal time left, I implemented the use of plugins or RubyGems. However, this did come as a cost. Some RubyGems have really poor documentation and customisations, so integrating it to look seamless with the current look and to make it function to exactly how I intended it to. The by-product of this, was an improvement in my understanding to how jQuery works and putting my JavaScript practice in CS110S into actual use, albeit not to a very major scale. As I made more tweaks to the code using jQuery and JavaScript, more bugs came popping up along the way, which kept breaking functionality.

This brought me to my next realisation, the importance of Git and version control. Throughout this experience, I realised that I placed too much importance on making big overhauls in code before making commits. Many times, there were bugs that appeared midway and I was unable to undo some of them since I made too many changes since the last commit. It's more of a habit that caused me to overestimate my accuracy in coding. The result was too much time wasting but regardless, I learnt a

lot about this and also did research by looking at other repositories to observe better versioning behaviours from more competent and experienced developers.

Lastly, if there was one thing I definitely overlooked, it has to be preparing tests. I definitely did not prepare proper test cases so I kept overlooking certain things and discovered bugs midway after stumbling upon them by complete accident. This shouldn't have been the case. With proper tests set up, bugs could have been discovered earlier and rollbacks could have been done quicker to avoid a lot of the resultant time wasting. I definitely lacked the foresight in this aspect.

Regardless of the many mistakes I made, I'm quite proud of my work. I've juggled 6 day part-time work weeks and other commitments, getting stressed about my Rails project along the way but self-studying this with an end goal, a project to work on, gave me a true purpose. I dare say, I've learned so much and achieved so much in this break than I had the entire semester 1. I can't wait to learn more. I'll definitely be trying to improve this project in my own time for my own practice and maybe even try to come up with my own project from my own ideas.

Features:

- Simple user authentication
    - Make your own private to-dos
- Tagging system
    - Tag your to-dos with your own tags such as ‘Work’ or ‘School’ for easy filtering
    - Filter out the to-dos with the list of tags that you’ve made for easier reference

Usage instructions:

Accessing your to-dos

1) Create your own account or log-in to the pre-made sample account
- ‘admin@example.com’ – ‘password’
2) Enter the portal
3) A list of all your to-dos will be shown.
- The checkboxes show if they have been completed or otherwise
- The checkboxes are non-clickable; if you would like to mark them as done or incomplete, edit the to-do
4) You may click on the tags if available, to filter out any tagged to-dos

Creating a to-do

1) Click on “Create a to-do”
2) Fill out the to-do item: a short but concise
    - Note that this must not be empty
3) Fill out any relevant details for your own reference
    - Not compulsory
4) Set a deadline if required
    - Choose the date using the picker
    - If the task requires a specific time, deselect the “All day” checkbox and a time picker appears. Fill out the time using your arrow keys or by typing out the time. Don’t forget the AM/PM at the end!
5) Set some tags if you’d like!
    - Choose some pre-existing tags or add more using the button “Add a new tag”
6) Create that to-do!

Editing

1) From the main page, select your specific to-do
2) You will be able to view all the details of that to-do
3) Delete or edit from this page!