CVWO Final Submission Writeup

1 Accomplishments

1.1 Journey with Web Development

Throughout the whole experience with CVWO, I have learnt the basics of the languages and frameworks required for the assignment(CSS,HTML, Ruby, Rails). I must say this is not easy as Ruby by itself has many new syntaxes which might arguably feel more "hackish" than Python, my first language learnt during CS1010X. Picking up web development with all the other supporting languages and tools just makes focusing on a single problem even harder.

Other than the conventional "learning to think by first principles", I improved my ability to "google effectively". This is useful for searching for existing code to make your life easier, such as gems and their documentation, or the commands required for executing an action, such as command line calls for git.

While these privileges exist, we are likely to not have a deep understanding in the code, and run into problems such as conventional programming bugs (syntax errors in learning new languages , logic errors in the interaction with files written in different languages) , as well as "command line" errors, which do not often happen in the formal education scene. In such a short time for completing the assignment, it would usually take me less than 5 minutes of attempting to code-trace any form of the above errors before I resort to googling and reading through multiple posts of StackOverflow for potential fixes and trying each one out. While the conventional approach is more important in the long run, I believe this is an essential skill as well, especially when under a strict deadline to complete something in a totally foreign field.

1.2 Addressing previous problems from Mid Submission

For the implementation, I deviated slightly from implementing an entire tag feature with a controller. Just implementing one with only model methods involved makes the application a lot more simpler.

Heroku deployment was successful eventually as I learnt that sqlite3 is not supported by Heroku. Switching to postgresql solved the problem.

2 Possible improvements

There are some things which I still have a huge room for improvement on, and wish I had more time for while doing the assignment.

2.1 Test Driven Development

This was something that really caught me off-guard, which was promoted strongly from a Rails Tutorial by Michael Hartl. This involves writing test cases first about

what the application expects to fulfil, and surprisingly there is a lot of things to consider and we tend to overlook those, like email formats, user authentications. Most of the time in the tutorial I was just trying to appreciate how they thought of the feature to test and how they actually tested for it. After I tried extending the application for more features, I realised I am not really confident at writing good tests. This really requires good grasps of the language and what exactly new features require, all of which I didn't really manage to achieve in this short period of time.

2.2 More user friendly features of a ToDoList

1. Some of them include a priority list which reorders the task based on the deadline of the task set upon creation. Many more ideas can follow like a calendar user interface just like modern ToDoLists. As of submission, I am using a simpler model of only displaying task in reverse order of creation.

2. Search bar and word recognition for any form of pre-existing tags

Most of these ideas can come about if I read into bootstrap more widely, but I decided to prioritse writing functional code rather than beautifying the application.

2.3 Refactoring and abstraction

The tutorial took care of most of the code duplication as they tried to reuse the files constantly. However, extending features beyond the tutorial was mainly done by improvising on other tutorials and hence much of the code wasn't well understood and opportunities for abstraction can't really be found.

3 User Manual

The aim of this to-do application is to keep track of tasks, maintaining user-friendliness and provide some motivation to complete tasks.

3.1 Sign Up

Fill in your name, email and password. There will be certain validations placed on each component where you have to pass to successfully create an account. You are automatically signed in.

3.2 Login/logout

The correct combination of email and password of an existing account is required for a successful login. You will be redirected to the home page where you can **create** a new task and see a feed of their tasks displayed in reverse order of creation, and **destroy** them when you are done.

When you are logged in, you can also change your password under Account Settings or Log Out, both through the drop-down menu on the top right hand corner of the page.

There is also a root account that is capable of deleting other accounts, read more in the README file in the repository

3.3 Profile

This can be accessed under Account Settings, where you can see your submitted name and email address. You can **update** any of name, email and password may be modified by clicking on Save Changes after making changes.

3.4 Motivation Factors

The users tab on the top right hand corner **show** the existing users and the number of pending tasks using the same application, as a slight form of motivation for completing your own tasks. This is accompanied by a famous baby meme picture at the bottom of each page.

3.5 Credentials and Contacts

On the footer of the page, there are About and Contact tabs. This is a simulated section in case there are some problems with the application and attention can be raised to the problem. Along the same line contains the source of where most of the code comes from.

3.6 Tasks

3.6.1 Create Task and Destroy Tasks

This is the main part of the application. You can **create** a new task only from their home page by filling in content and tags separated by commas. Keep the contents as concise and accurate as possible, preferably without subtasks, as task editing is also not supported in the application. You can only **destroy** your own tasks when you are done, thereafter a prompt will be flashed for you to confirm your decision as archiving tasks is not supported.

3.6.2 Filtering tagging system

Tags can only be created via creation of tasks to prevent the case of idle tags. Clicking on a tag can redirect users to a new page with only tasks under those tags. In the new page, you are only allowed to **destroy** tasks, but not allowed to **create** a new task.