CVWO Mid-Assignment Write-Up

(Kenny Hermawan / A0200758H)

Overview

The web application **CheckMark** (placeholder name) is an integrated task listing platform providing a way for users to conveniently and effectively manage all the day-to-day jobs and errands that they need to do.

Usage and Features

- **Basic Use Cases**
  1. Users can create, read, update, and delete items in their to-do list:
     - a) Items in the list are comprised of a single line of text / content.
     - b) New list items can be added with forms.
     - c) All items in the list will be shown in the app's index page.
     - d) Users can edit an item by clicking on it in the index page.
     - e) Items in the list can either be completed (✓) or deleted (×). The two options will have no difference in the first implementations (subject to development in future versions).
  2. Users can create, read, update, and delete tags associated with their items:
     - a) Tags can be used to categorize items in the list.
     - b) Tags are below items hierarchically.
     - c) An item can be assigned with (at most) 3 tags.
     - d) Tags can be created when users are editing an item.
     - e) The contents of a tag can be edited after it is created.
     - f) Tags can be deleted.
     - g) Deletion of an item will also result in the deletion of their tags.
  3. Users can search for items in their list (and the tags assigned to said items):
     - a) A search bar is provided at the index page for users to type in keywords.
     - b) The app will then only show the items that contain said keywords, or contain tags that contain the keywords.
     - c) In its first implementations, in order to search for items / tags, users will need to click the submit button next to the search bar (subject to development in future versions).
  4. A basic security authentication method (username and password) is present in the app.
- **Extra Features**

  The following extra features are to be considered and potentially implemented after the implementation of all the basic features described above.

1. Items in the list that are completed (✓) will not be entirely deleted, but saved and pushed to the bottom of the list. Items that are deleted (×) will be destroyed from the database.
2. A sorting mechanism to sort out the items based on certain criteria (tags, date created, etc).
3. Fast search (results of the search will be shown without the need to click the submit button).
4. Log in and log out feature (and a more secure authentication method).

Execution Plan

As the technology stack needed to build the web app is quite varied (Rails, React.js, Heroku, etc.), my plan is to work on the web app gradually. Before using and implementing a new technology (such as new frameworks) in my web app, I plan to gain ample knowledge and competency in said technology first instead of integrating it into the source code straight away. Below is my detailed execution plan:

1. **Initialization:**
    - Create new Rails app and initialize local repository.
    - Create remote repository in GitHub.
    - Initial deployment / testing through Heroku.
2. **Basic CRUD and Security Functionalities:**
    - Implement CRUD functions and React components for items in to-do list.
    - Implement CRUD functions and React components for tags.
    - Implement security authentication method.
3. **Basic Search Feature:**
    - Implement search function and its React components.
4. **Styling and Layout**
    - Configure the layout and styling of the web app using available UI libraries.
    - Implement Bootstrap (to be considered).
5. **Extra App Features:**
    - Implement extra features as described above (to be considered).
6. **Optimization and Extra Functionalities:**
    - Implement TypeScript, Redux, and/or other useful functionalities (to be considered).
    - Final compatibility check and deployment through Heroku.

Obstacles Faced (So Far)

The problems I have been facing so far are mostly related to Rails itself, a framework that was completely foreign to me prior to this assignment. I find it difficult to gain complete mastery and understanding over all of Rails' functionalities, due to the high amount of abstraction present in the framework, the so-called "Rails magic". However, I am keen to continue learning, as figuring out how this "magic" works is immensely satisfying and exceptionally useful in understanding the things that make Rails so powerful.

One obstacle that I have (unexpectedly) faced and (partly) overcome is Linux. As a lifetime Windows OS user, having to use Linux and its bash shell to do everything (installing Rails, Git, and all the other components of the web app) was exceptionally frustrating for me, mainly due to my unfamiliarity with UNIX-like OS and the various bugs and errors that regularly interrupt my work. However, I do find the journey and learning process (including all the debugging work) immensely satisfying as it has given me a deeper understanding of Linux's versatility and capabilities.