Mid Assignment Submission

Ezekiel Toh Fun Kai

A0218061N

Basic use cases

- Todo-list
    - Access to CRUD functionality for every todo item
    - Dashboard with list of all pending todos
    - Archive for all completed todos
    - Todos contain: title, tag, and a complete-toggle — will include 'deadline' if possible
- Accounts
- Search
    - Search by title or by tag
- Sort
    - Sort by deadline or date created/updated

Execution plan

1. Create Rails API //completed

- Build model for todo item
- Build controller for HTTP Methods for RESTful API

2. Create CRUD functionality on React client //completed

- Use React Hooks to manage state and fetch API with useEffect hook and axios.
- Build functions for respective CRUD functionality
- Integrate components with CRUD functions, user-friendly

3. Create search function //completed

- Build a search bar for users to search either by tags or by title

4. Create authentication for user accounts and login functions //to-be-done

- Establish devise on rails
- Build user authentication
- Create login page
- Create profile page/account management page

5. Build app into Material Design standard //in-progress

- Beautiful typography //incomplete
- Simple, minimalistic design //in-progress
- Choice between dark and light theme (bonus) //incomplete

6. Deploy on Heroku

Suggestions

1. A todo list is useless without a deadline feature — sorting the inbox/dashboard by deadline would be much better than what it is now
2. Adding additional remarks for each todo is a good idea, although not absolutely necessary in my opinion
3. Implementing Context API would be much more beneficial than what I am currently doing: putting props where they don't belong, just to pass it down a chain to be used by a component

Difficulties

This is my first web-app — in fact it is my first ever app. My only prior knowledge of programming came entirely from CS1101S, so picking up Ruby, React.js was very daunting at first. Learning about fetching APIs from the back-end to the front-end boggled my mind for a few days.

- Lack of web-dev experience
  - No idea what back-end, front-end, full-stack, APIs etc. meant.
  - Classes were confusing, and some blogs were using React Hooks, which confused me a lot more before I figured out what classes and hooks were. In

    earlier versions of my app, I included both hooks and classes into the same code

    - Didn't understand how RESTful API worked, couldn't grasp the concept of fetch requests and other RESTful methods
    - Took me awhile (days) to figure out how to run Rails on PostgreSQL (which I wanted for deploying to Heroku)

- Bit more than I could chew
    - Started to change my app into a TS app, ended up messing everything up with types and interfaces — maybe I'll pass for this assignment