Assignment: Riding on Rails
By Zhu Hanming A0196737L
Computing for Voluntary Welfare Organisations (CVWO) AY2019/20

Before Starting – Initial Analysis:

After reading through the requirements, below was my initial analysis (for me personally):

Challenging
- Ruby on Rails
- SQL
- TypeScript
- Cron
- Hosting

Difficult
- Docker

Personal Goals
- Apply React Hooks and Context in app

Problem Statement

As a frequent user of to-do lists, I have always found the lack of time-based planning lacking. It's easy to find to-do lists that help users plan out their list of tasks for the day, but there are barely any apps for long-term planning, i.e. setting an overarching long-term task and creating subtasks (with individual deadlines) under it. Some that are out there are habit trackers (which are not what I want), and apps that are close, such as TickTick, still ultimately lack the support for subtask curation. Furthermore, they are not free.

As such, I wish to create an app that helps users plan every single detail of their life down into a clear to-do list. This to-do list would thus support both small and large tasks, and help users break those large tasks down into manageable subtasks.

The core feature would be that users are able to view their tasks in list view, Kanban view, calendar view.

This app will also come with dark mode. I will also explore the option of a Progressive Web App, though there may not be time to implement it given the various views I wish to build.

Use Cases

The main use case, as mentioned above, will be for users to create, read, update and destroy (in both senses of the word) tasks and subtasks. As such, the app will need to support CRUD for these two objects/models.

There will also be a need for user creation/registration. Authentication will be done using JSON Web Tokens.

Core Paradigm

One important paradigm of this app is – if you have subtasks under a task, you cannot simply check that task off. In fact, when all subtasks are completed, that task would automatically be completed as well. This is to encourage users to break down tasks into subtasks, such that each subtask is a small piece of the larger task, and the completion of all subtasks would mean the completion of the task.

Now we will delve into the specific views, and the expected functionalities of each of them.

Views

Home/Main Overview

The Home page will act as a dashboard that allows users to gain an overview of their tasks. There will be two core sections – one that informs users on their incomplete/overdue tasks for today, while the other will let the user see the progress of their tasks, ideally those with subtasks.

There is also the calendar and fun fact section, which may be revised subsequently.

More can be found in the mockups under the mockups folder of this repository.

List View

This view is the most traditional view. With an interface that's split in half, users can see both the tasks and the tasks they're focusing on.

They can see the tasks they have categorised under due today (or overdue), due by this week (or overdue), all tasks, and completed tasks. The user is able to switch between the different categories and check the tasks off (if they have no subtasks). Else, they can click on the task, and view the details of that task on the right. The right half acts as both a "card" and a form – any changes made to any of the details of the task will update the task.

The goal is to have a beautiful and seamless feel.

More can be found in the mockups under the mockups folder of this repository.

Kanban View

Similar to Trello, users will be able to view their tasks in a Kanban board form. They can drag and drop the tasks, allowing them to push the tasks they feel are more important forward. Within each Task card/board, there are also cards/subcards which are the subtasks. They can similarly rearrange the subtasks as they like.

When the board is expanded, a form similar to that of List View will pop up as a modal.

Calendar View

This is the view I am most excited about. In this view, users can see their tasks and subtasks shown as events on a calendar. This is also the most complex view, and may require some engineering to achieve, since I will have to find a way to make the subtasks and their parent tasks obvious and intuitive. It is not yet confirmed whether they are able to interact with the tasks while in this view, but should they be able to, the details of the task should slide out from the right, similar to a sliding pane/menu.

This is also the view that I want the most, and is the reason why I am building this app.

Current Progress
As of the time of writing, the core of the Main View has been built, and the List View is almost complete, less the subtask creation. The basic CRUD features of tasks are in play, and soon that of subtasks will follow.

Moving Forward
The remaining two views will be worked on. There are also some other features/goals/tasks that I am looking at:

- Cron jobs
- Docker
- Touching up of the app – settings, main page, small things such as favicon
- Landing site and assets
- Hosting of API on api.hityourtargets.xyz as well
- Mailing services – possibly email verification
- Email templates
- Facebook Login
- Notification system on tasks
- Gamification – create achievements and records such as longest task, most subtasks etc.
- Comment and document code
- Look into integrations with Google Calendar, Calendly, Trello, GitHub

Challenges
The biggest challenge for me is that I'm still very new to the backend. Many things that I wish to do on the frontend side requires support from the backend, but I am still in the process of learning. For example, achievements and records would require some way to store statistics on users, something my database currently does not have.

Takeaways
Nonetheless, this process has been full of takeaways for me. It's only the halfway mark, but I've learnt a lot throughout this process. On the backend side, I've picked up Rails, though still very new. I've done my fair share of reading up as well on MVC.

On the frontend side, I challenged myself to write code cleaner than I've ever written before, and I'm glad it has worked out for the most part. This includes the use of SCSS/SASS, eslint, Contexts, Hooks, etc.. All of these are things that I've almost never used before. It was also a good opportunity for me to reinforce what I had previously learnt.

All in all, this process has been very fruitful, and I'm looking forward to making the rest of my app a reality.