Assignment: Riding on Rails

Zhang Ziqing, A0223145R

Introduction

In today's fast-paced world, it is common for people to perform multitasking which could be stressful because they have to remind themselves of every deadline in mind constantly. Even so, they may forget some tasks sometime, building up their stress level. A To-do List App is thus a simple yet useful tool for them to unload their mental burden so that they can just refer to the To-do List when needed. In this assignment, I aim to create a simple To-do List Web App to help busy people plan their daily tasks.

Use Cases

1. **Create new tasks:** Users with a lot of tasks in mind want to add new tasks by simply key in their tasks. (They may want to add more details once the task is created.)
2. **Read existing tasks:** Users want to have an overview of how many tasks they have in total in a list.
3. **Edit task details:** Users may want to add more descriptions to existing tasks.
4. **Track task progress:** Users want to mark tasks completed by ticking the checkboxes. They also want to view the list of completed / ongoing tasks by clicking on sidebar options.
5. **Delete unwanted tasks:** Users want to delete outdated tasks by clicking on delete option.
6. **Categorise tasks:** Users who have many concurrent projects want to tag their tasks by editing the existing tasks. They will create new tags, go to the task overview page and then edit the tasks to add tags. After finishing editing and going back to main overview, they will want to view filtered list of tasks by tags by selecting drop down options in sidebar.

Functionalities Required

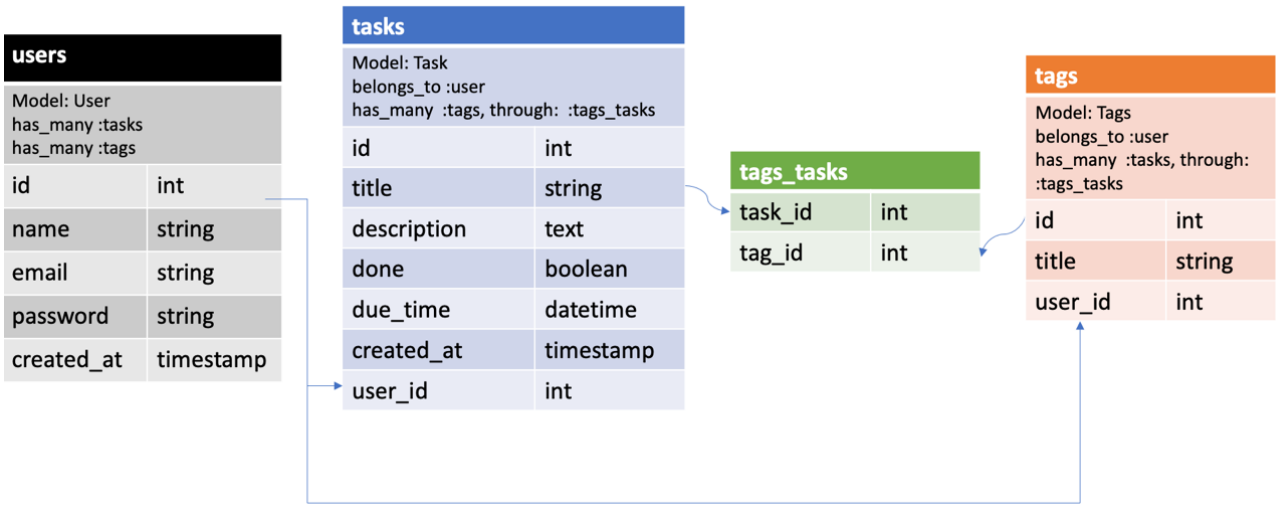
1. Create, retrieve, update and delete (CRUD) of tasks and tags.
2. User authentication including sign-up, log-in, updating profile and delete account.

Current Progress

1. Created backend using Ruby on Rails and frontend using React.
2. Basic CRUD operations on tasks and set up how to consume RESTful JSON API.
3. Some basic styling on task components using material UI.

To-do List

1. Add functions to edit tasks in React
2. Setup user authentication with JSON Web Tokens in Rails
3. Learn react router
4. Add tags: Setup relational database in rails (see below for draft proposed schema).



5. Learn styling: style the frontend using libraries.
6. Hosting of API

Challenges and Problems (in decreasing order)

1. **React:** Having the most difficulty understanding the structure of React app and how we should build it step by step due to lack of whole picture. It is simple to start but it gets confusing later. For example:
    a. **Basic workflow:** Should we build static version of React first? How to decompose each components into smaller components?
    b. **Hierarchy of Components:** What kind of code should be put into the same file? For one component, is it advisable to start with putting all codes in one file? (How can we learn the use of different files)
    c. **Hooks:** Is usage of hooks from the beginning advisable? Is it ok to use a mix of Hooks and class components?
    d. **State and props:** Having difficulty understanding when and why to use them.
    e. **Styling:** Is it recommended to do styling at the later step or along the way? (Having difficult learning how libraries such as material UI works across different components and how to override the styles as they use many props)
2. **Relational database for tagging**
3. **React Router** (Related to point 1)
4. Not very familiar with consuming RESTful APIs

Learning Points

The learning curves for both React and Rails are steep. My perception of the difficulty level is similar to a damped sine wave, which initially vibrates significantly and becomes closer to reality as I keep on learning and gradually gain the momentum. As I pick up new concepts and apply them through trial and error, it soon becomes interesting to see how powerful Rails and React are. The journey has taught me to be brave but not perfect. The knowledge comes from the mistakes I have made so far. I would like to keep on trying and taking on new challenges.