Introduction
Creating a to do app may seem simple, but when it comes down to the details it's difficult to create a system that users would love. I had to reflect on what I think was useful in the to do apps that I was using, as well as to think about why users will love a particular todo app.

Features in this app
My personal philosophy is that any app should have a simple, intuitive user interface, with a set of small but powerful features. Based on the requirements of this assignment, I have decided that my minimal viable product must have the following features:

1. CRUD features: add, view, edit, delete tasks.
2. Tag tasks
3. Search tasks
4. Filter tasks

The following are nice to have features. These are features that I enjoy the most in todo apps that I'm using or feature I wish to see implemented. Currently I have been using Habitica, a todo app that gamifies tasks.

5. I want to be able to add recurring tasks (tasks that happen daily or weekly)
6. I want to be able to set deadlines for tasks
7. View completed and uncompleted tasks

These are features in other todo apps that I might consider adding to my project:

1. Creating projects, where tasks or milestones in the project can be added as subtasks of that project
2. A gamification system, where users will earn 'experience' points for completing tasks and lose these points if they fail to do what they set out to do by the deadlines.

Difficulties faced
A major difficulty I had was in how to plan and complete a programming project, especially a full stack web app. It required me to pick up a lot of programming languages – HTML, CSS, Javascript, and Ruby, as well as to learn database management and a little UI/UX.

Surprisingly, another difficulty I had was to plan the user experience. For example, to indicate that an element (such as a button) was interactive, a UI change should occur to the element when the user mouses over the element. I spent some time thinking of what style changes would be more intuitive to users and looked at how other websites styled these changes.

Lastly, a difficulty I had was to resolve errors when using Rails. Some of these errors stem from using gems with incompatible versions. While these are not too difficult to troubleshoot, they take quite a long time to do so.

Execution plan
December:

- learn HTML, CSS, Javascript, and React.
- Create some HTML/CSS pages to visualize how the final app would look like. The pages can be seen in the mid-submission folder of the repository.

- Read up on Ruby on Rails and do some simple Rails apps by following online tutorials. The Rails apps are pushed to the ToDo list github repo as branches.

Planning

- Decide on the models in my ToDo List app:
    - List:
        - Key: string [will be mapped to a user, this will be randomly generated]
        - Title: string
        - Tasks: references [relationship: has_many]
    - Task
        - title: string
        - description: string [optional]
        - deadline: datetime [optional]
        - isCompleted: Boolean [default = false]
        - tags: string [has_many, optional]
- Decide on the routing and controllers
- Write React Components

January:

- First week: create a very minimal todo list that supports CRUD
- third week: add tagging features
- Fourth week: add searching and filtering capabilities