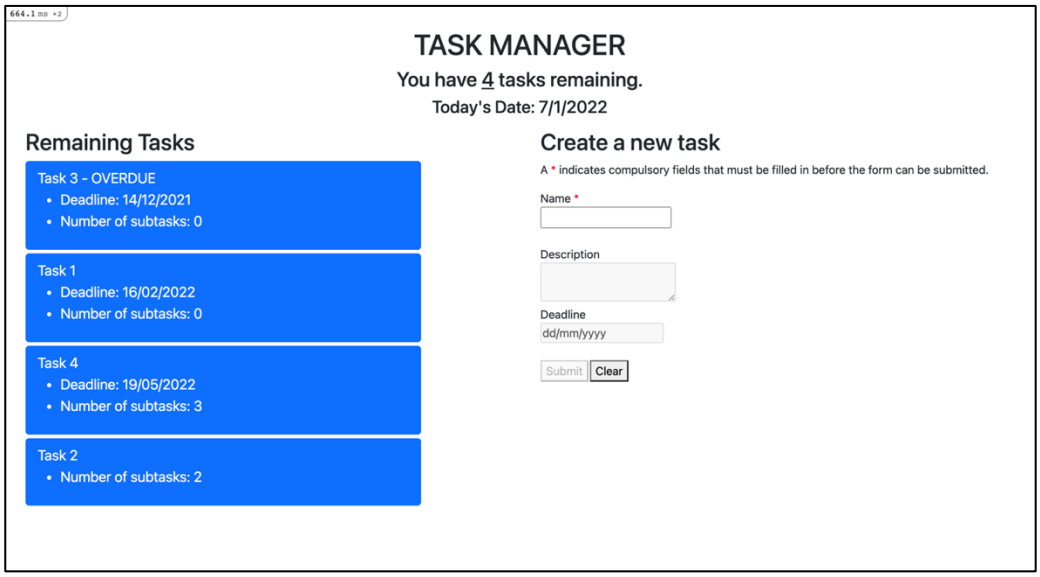
User Manual

Opening the App

1. Fork the repo
2. Clone the forked repo into a directory of your choice
3. Open your terminal and navigate into the cloned repo
4. Install yarn if you do not yet have it installed
5. Install dependencies for the project by entering the command “yarn install” into the terminal
6. Create the database and run migrations by entering the command “rails db:prepare”
7. Seed the database by entering the command “rails db:seed”, or skip this step for an empty database
8. Start the server by entering the command “rails s”
9. Open http://localhost:3000 in a browser of your choice to view the app.
10. You should see a page like the one below, in which case you have successfully opened the app

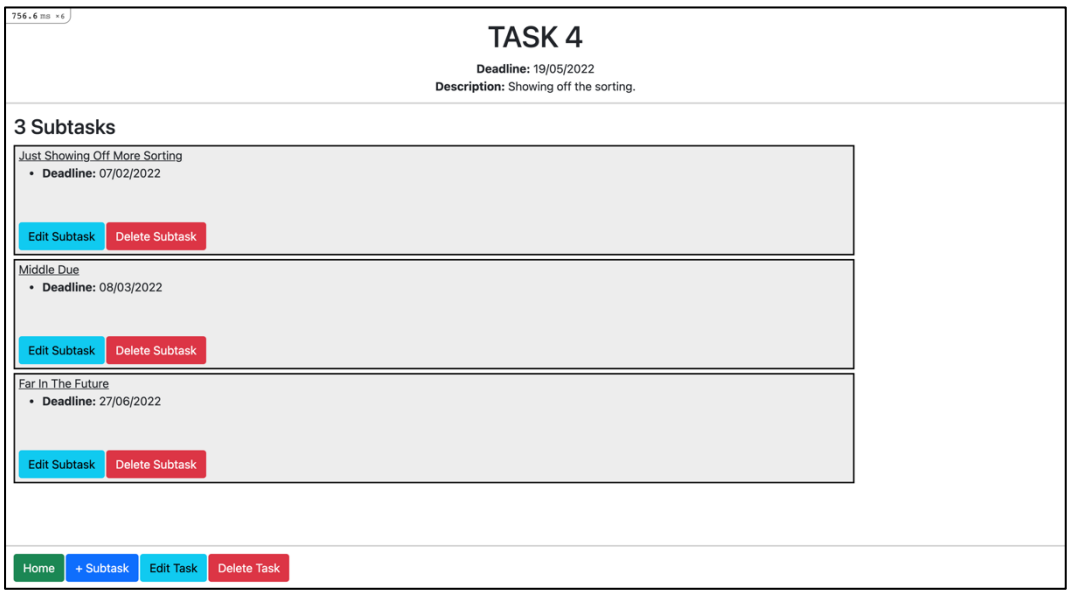


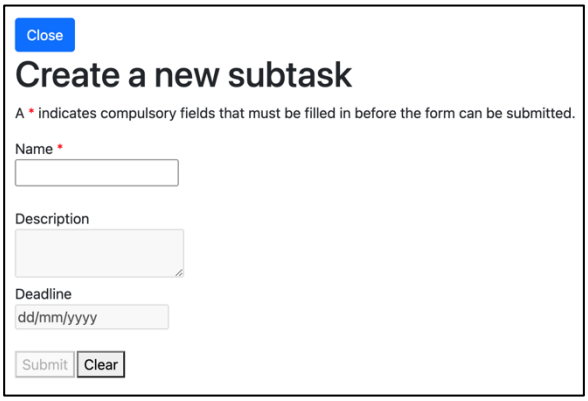
Home Page

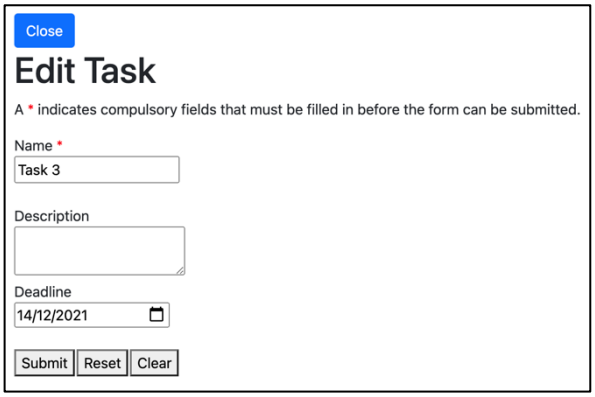
The app opens on the home page, where there is an overview of all tasks in the form of pressable buttons that directs you to the individual task page, and a new task form. All tasks are identified by a corresponding slug formed from their name, and the new task form does not allow repeat names.

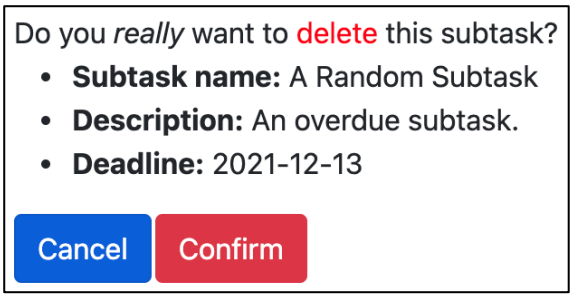
Individual Task Page

On an individual task page, you can see the details of the task and all subtasks under it. You can also perform actions such as editing the task, deleting the task, and creating new subtasks, and so on.









Code

The source code is in the folder **app**. Within app, the important folders are **controllers**, **javascript**, **models**, and **serializers**. The **controllers**, **models** and **serializers** folder hold the backend Ruby on Rails code to manage the database and send and receive JSON messages upon request. The **javascript/components** folder holds all the frontend React code that builds the app. The React code is written in JavaScript.

The database, migrations, and seed data are found under the folder **db**.

My Accomplishments

Regarding the backend, I am still somewhat confused about the syntax and grammar rules of Ruby on Rails. However, I now understand how the backend communicates with the frontend - through JSON messages that can be sent upon requests by the frontend. I also better understand how to add further functionality to modify the behavior of simple CRUD actions of the controller.

Regarding the frontend, I am a lot more proficient at React now, unlike how confused I was when I first did the online tutorial. Once I experimented with it more and figured out that I could use JavaScript, I was able to make use of what I had learned in CS1101S and twist that knowledge to fit with the new framework. I have learned how to break down an app into individual components and pass down properties from parent to child, as well as use generic components and functions across multiple components.

There are still many areas to improve in. I would like to implement user authentication such that multiple people can make use of the same app and see the tasks that they've created.

I would like to further clean up the react code, such as extracting the various form functional components into a generic functional component, since all the task and subtask forms share many similarities. I would also like to add error handling in the form of error boundaries so that some parts of the app can continue to work even if something small breaks.

I would also like to spruce up the presentation of the app, since I focused on that aspect the least and did not spend too much time on it other than making the app look presentable. From reading about CSS properties online, I learned what the :before and :after properties do. I would like to learn to customize scrollbars and implement simple animations, as well as find out what else CSS can accomplish, along with React effects like typewriter.

I gained a lot of knowledge over the holidays, very much assisted by the hands-on experience in creating the app myself, and I am very happy and proud about that. Even if I do not get into CVWO, it was a good learning opportunity to pick up new knowledge and skills that will help me in my future modules in NUS.