Being new to both React and Rails, I had to take the tutorials slowly and learn as much as possible about both programming languages. For specific requirements such as Create-Read-Update-Delete (CRUD) and filtering on React, I had to google for explanations and tutorials to further my understanding and knowledge. I implemented CRUD using axios libraries in React.js, which I considered to be following RESTFUL Application Programming Interface (API) calls. React has powerful filter and map functions, which I was familiar with from CS1101S, which I used to make my Read.js component, to filter out specific data that the user is searching for.

I faced multiple issues in the creation of this Task Management application. Early on, one issue that I faced was having issues using the sample-react-app and integrating it with a Rails API. I created a new React application and learnt about props and state to manipulate data within the frontend, to have a much easier time with merging React and Rails.

One major issue that I was facing in the middle of the development process was that I could not send GET and POST requests to my Rails backend, using Postman and from my React Backend. After hours of googling and testing, I found the answer in ‘rack-cors’ gem on Rails, which by altering parameters allows me to bypass the Control-Access-Allow-Origin header, allowing me to GET and POST data from two apps hosted on the same server.

Lastly, I deployed the application on Heroku. While the React page deployed well, I ran into issues with the Rails API. Heroku doesn’t allow SQLite databases, which my backend used, due to it’s file system. After numerous attempts to move my backend Rails API to use PostgreSQL instead, I ended up failing to migrate the databases and did not deploy the full application in the end. In hindsight, this boils down to planning at the start of the project, seeing which languages suit your requirements best. Had I checked from the start which database Heroku accepts, I could have avoided this entirely.

Overall, I feel quite accomplished, despite the mediocre-appearing webpage that I made. Having to learn programming languages and concepts from scratch, having to remake the backend Rails API multiple times to get it right, having to debug code and search for answers online, it was a challenge for me to create this webpage. However, I am proud of my first Web Application and what it can do, and I hope to produce better work in the future. I hope to work on more projects like this in the future.

**User Manual**

1. Fork this repository.
2. Clone the forked repository.
3. Open your terminal and navigate to the directory with the cloned project.
4. Navigate to the rails-api directory.
5. Run **rails s** to start the backend.
6. Open <http://localhost:3000>/ to view it in the browser.
7. Open <http://localhost:3000/api/v1/tasks> or <http://localhost:3000/api/v1/tags> to view stored JSON data.
8. Navigate to the react-page directory.
9. Run **npm install / yarn install.**
10. Run the app in development mode by running **npm start / yarn start.**
11. Type **y** when prompted to start on different port. It will start on port 3001.
12. Open <http://localhost:3001/> to view it in the browser.
13. Create new/delete tags using the **Manage** button.
14. Create new tasks using the **Create** button.
15. **Search** for tasks using the search bar, and **filter** according to tags.
16. **Update/delete** tasks using the buttons.