Tasks Manager

User manual:

Tasks manager would help users keep track of all tasks. Follow the readme instructions to have your tasks be stored locally in your device.

Creating tasks

Users can create tasks by clicking on the floating action button on the bottom right of the screen. The app should work with both large and small screen sizes including mobile devices, as it has been tested with multiple screen type. The user would be greeted with a new task page which prompts the user to create a new task. Only the name of the task and due date/time is compulsory fields. Save button would only show if compulsory fields are not empty. The due date/time is converted from users' local device time to UTC time and stored in database.

Viewing tasks

The tasks are presented in a responsive table form. Clicking on any table header would sort the tasks. The due date/time here is converted from UTC time to the users' local device time. This would allow prevent wrong due date/time should the user moved into another time zone.

Editing tasks

Clicking the checkbox for the tasks would allow the user to update the tasks as done or delete it. Clicking on a task would allow the user to edit fields in the tasks. As the tasks uses a unique identifier, changing the name of the tasks is also possible. The app always assumes the user know what is right.

Finding tasks/tag

The search bar allows the user to find tasks by name, description and tag. This would allow the user to filter through tasks which are tagged. Do note it is case sensitive.

Try it online https://task-manager-petery.herokuapp.com/

As Heroku would shut down processes after more than 30 minutes of inactivity, expect the cold boot time to be 15 seconds. After which expects only responsive UI since it is implemented with react.

Accomplishments:

Uses react with material-ui styling for frontend and rails for backend. The backend rails app serves data on localhost:3001 and manages database. The frontend react component follows typescript syntax and uses axios to get data in json format from rails. React components are used to properly present the tasks and display it nicely by following material ui theme. React also allows for the responsive transition between different pages of the UI. Lastly, both frontend and backend were hosted on Heroku with only one dyno allowing the app to be accessed from anywhere.

Challenges faced:

Right after mid submission when school is starting:

I have a full rails app which runs frontend with backend together. Before adding react, I fully implemented all the crud functions with filtering/sorting/editing/creating tasks, and thoroughly tested it. I wanted a perfect rails app before moving on.

Adding react:

To my dismay, converting rails to backend with reactjs frontend did not go well, no online tutorial seems to be working, they were outdated/had different configurations/not written detailed enough to work with my app. I end up rebuilding the app again. As I was starting from scratch, I chose Facebook's create-react-app as it is better documented. The different react components, class/functions/headless components really improved my JavaScript skills as I need to solve the wide variety of syntax errors that comes with these varied react component style. I also encounter problem by letting every variable be var, datetime was not string format in database and was throwing type error in the react side. Hence, I converted react to use typescript to reduce unexpected/wrong typing. Typescript's useful lint and intellisense also made it worth the conversion.

Hosting on Heroku:

Heroku was even less documented than converting rails to react. They only provided the most basic tutorial on hosting rails app in Heroku. I initially hosted my frontend and backend separately as 2 dynos. However, there was no useful instructions on routing axios calls from frontend to backend. I had to adapt multiple tutorials on hosting rails and react together in the same dynos which finally solve the problem.

Future todos:

Add login page, user authentication to allow different users to manage their tasks. Having frontend and backend complicated the implementation, hence this was tried without a successful attempt.