Name: Tai Jun Jie (A0239022N)

CVWO Mid Assignment Writeup

Github Repository: https://github.com/TJun-Jie/T-Jun-Jie_CVWO-Assignment-2022

Use Cases:

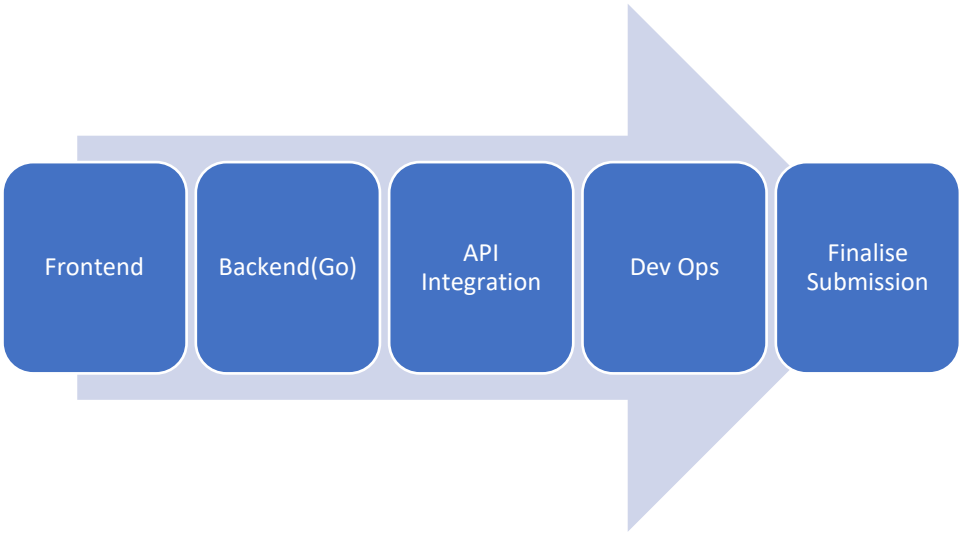
Basic Functionalities (Will be completed)

1. Users can view all the created tasks
   - The user would be able to see all the created tasks on the home page of the application.
   - If the user wants to view the tasks based on the task's priority, there will be a side drawer on the left, where they can click the priorities' which they will be routed to a new page where they can see only the tasks with that selected priority.
   - There is also another page where the user can see all the completed tasks. (The home page will not contain any completed tasks)
2. Users can add a task
   - There will be a button "Add Task" on the home page.
   - When the user clicks the, a component will pop out, and they would need to fill in the new task's title, description and priority.
   - Then the user would click on the "Add Task" button to confirm the action.
3. Users can edit a task
   - On the home page, the user would see all the uncompleted tasks.
   - When the user clicks on one of the tasks, a modal will pop out. The modal contains a prefilled form with the current values of the tasks.
   - Then the user would need to click on the "Confirm Changes" button once they edited the task.
4. Users can delete a task
   - In the same form as the editing action, there is a "Delete Task" button, where the user would be able to delete the task.

Extra Functionalities (Will do if there is spare time):

a. User Authentication
b. Allow user to create their categories
c. Search functionalities

Execution plan



Frontend

Libraries:

- React (Frontend library)
- Material UI (UI library)
- Formik (Form state management)
- React-i18next (translation library)
- Axios (Data fetching)
- Yup (Schema validation)

I aim to use typescript from the start as this would reduce the time needed to convert the whole project from JavaScript to typescript. Also, it is pretty easy to implement from the start.

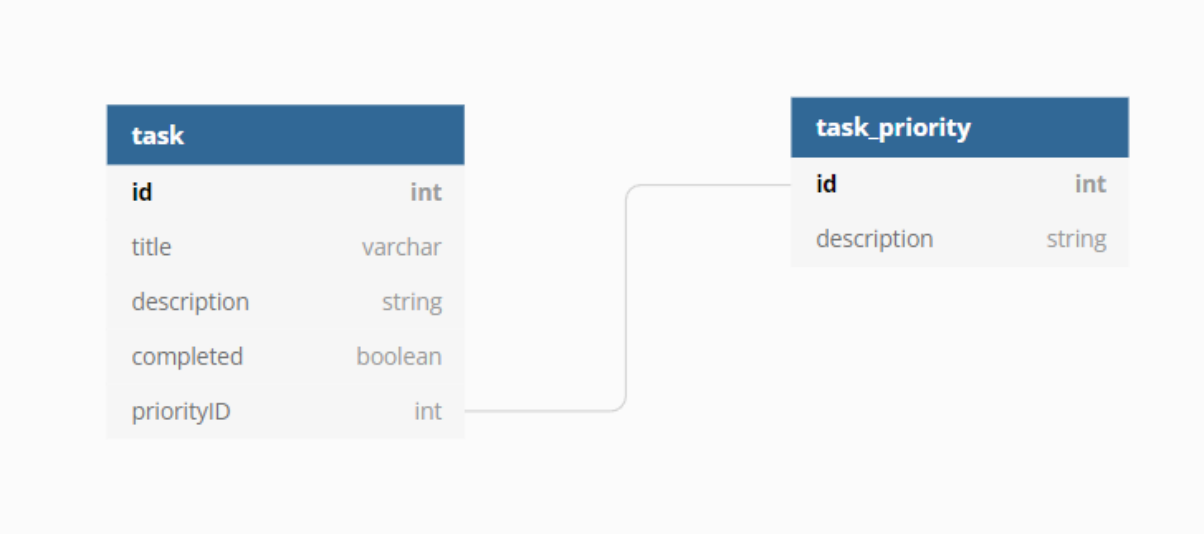
Additionally, I would convert the whole react state management system into Redux if there is extra time. For now, I would use React UseState and context API for state management first. Since I don't know which part of the application would need a global state, it wouldn't be wise to start with redux right from the start.

Backend

Database: MongoDB (App does not need complex SQL functions like JOIN)

Language: Go (Go syntax looks more uncomplicated compared to Ruby, and also it is suitable for creating simple APIs, which is perfect for our use case)

- Data Modelling



Dev ops

I would aim to use docker and AWS elastic beanstalk to deploy the app.

However, if I get stuck and fail to do that, the backup plan would be to use Heroku to deploy the frontend and Google cloud to deploy the backend.