Name: Marcus Pang Yu Yang
Repository link: https://github.com/marcuspang/cvwo

Use Cases

The task management application will allow users to create and keep track of their tasks, which the user can add a due date, some description, or label to each task. The tasks are stored in lists which they can rearrange freely using drag and drop within the list and between other lists. For a better overview of their tasks, there will be three different views of their tasks. Namely,

1. A "list" view where all the tasks are displayed in their respective lists,
2. A table view where every task is displayed row-by-row, with a sorting functionality available for the user to choose, and
3. A calendar view showing tasks with due dates assigned to them.

Additionally, lists can be shared to existing users for them to edit the list as well, adding a collaborative element to the application.

Main features to implement

1. Authentication
   - Users can register/login using traditional email and password through Go backend
   - Validation is done on frontend and backend
2. Tasks
   - Users can create/delete/update tasks
   - Tasks are stored in a list
   - Tasks have due dates and labels
   - Tasks can be dragged and dropped into other lists
   - Tasks can be seen on the calendar
3. Lists
   - Users can create/delete/update lists
   - Lists can be sorted based on creation date, labels, alphabetical order
   - Lists can be assigned colours
4. Calendar
   - Users can see all tasks with due dates assigned
   - Clicking on a date creates a new task
5. Miscellaneous
   - Board/List view

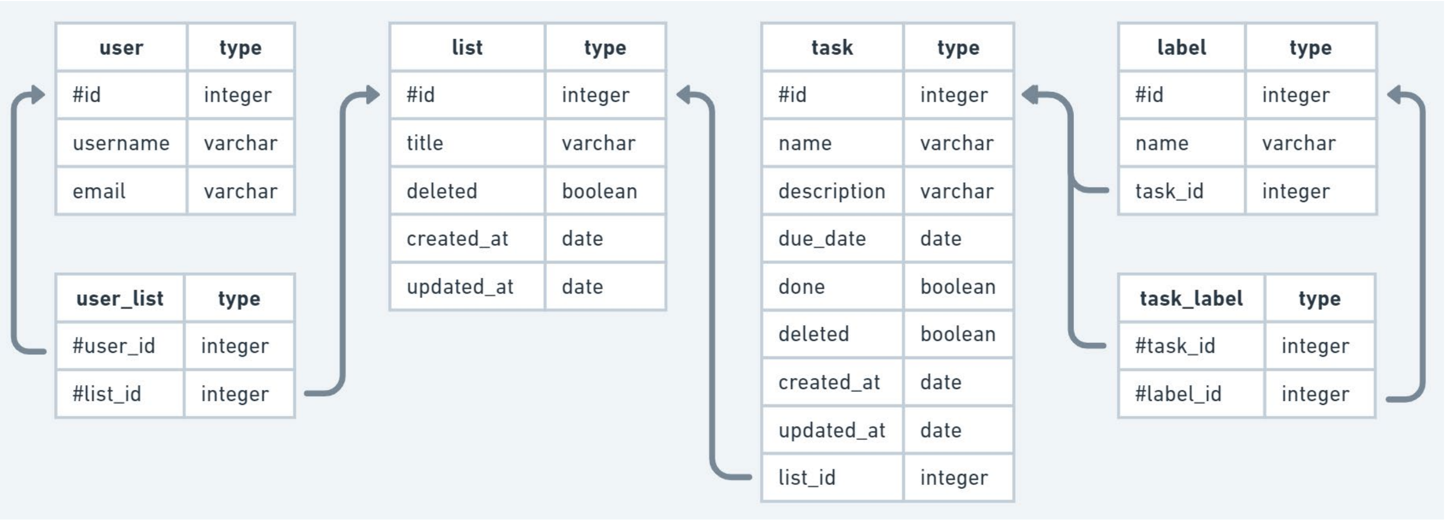
Name: Marcus Pang Yu Yang
Repository link: https://github.com/marcuspang/cvwo

- Trash bin to show recently deleted tasks
- Users can search through tasks and lists
- Notifications can be given based on the due date
- Tasks can be shared/assigned to other users (edit access)
- Lists can be shared/assigned to other users (edit access)
- Dark mode

Backend

For this project, I have decided to go with a Go backend as I have yet to try out Golang and I would also like to spend more time using a statically typed language (other than normal TypeScript).

Currently, my database tables will look as such, where the # indicates that the field is a primary key.



user	type
#id	integer
username	varchar
email	varchar

user_list	type
#user_id	integer
#list_id	integer

list	type
#id	integer
title	varchar
deleted	boolean
created_at	date
updated_at	date

task	type
#id	integer
name	varchar
description	varchar
due_date	date
done	boolean
deleted	boolean
created_at	date
updated_at	date
list_id	integer

label	type
#id	integer
name	varchar
task_id	integer

task_label	type
#task_id	integer
#label_id	integer

The user and list tables will have a many-to-many relation where each user can create multiple lists, and each list can be shared with other users after creation. Also, the task and label tables will have a many-to-many relation where each task can be assigned to multiple labels and vice versa.

Execution Plan

For this project, I will be making use of yarn workspaces to create a monorepo where the backend and frontend files will be located. With regards to my current progress, I

have created a Go backend following a tutorial, and a frontend making use of Next.js and ChakraUI for a mock-up of how the tasks will look like.
Over the next few weeks, I will finish up the database schema for my backend, and implement the various features outlined in the report.

After which, I will Dockerize my frontend and backend and subsequently deploy on Vercel and Heroku respectively. I have chosen Heroku for my backend and database solution as Heroku allows user to attach a PostgreSQL instance to the application. For the frontend, as I am using Next.js, I will be choosing Vercel due to its support and easy deployment process for Next.js applications.

Anything else to include in plan

If time permits, I would like to implement unit testing and end-to-end testing. Additionally, I would like to add OAuth for the frontend for better user experience.