CVWO Midterm Submission

Desmond Tan

December 26, 2021

1 Use Cases

The product is a web application that allows the user to organise their personal tasks and events so that the user would be able to prioritise them and complete them in a timely manner.

1.1 Features

- **User Authentication**
  - The user can login to the application using their email address and password.
  - The user can register for an account using their email address and password.
  - A jwt token would be generated for the user after the user has successfully logged in, and can be used to authenticate the user for a day before expiring.
- **Task Management**
  - The user can create a task using the task name, description, due date, and status.
  - The user can edit a task using the task name, description, due date, and status.
  - The user can delete a task using the task name, description, due date, and status.
  - The user can tag their tasks with a label.
  - The user can untag a task from a label.
  - The user can add a task to a list.
  - The user can remove a task from a list.
- **Event Management**
  - The user can create an event using the event name, description, start date, end date, and status.
  - The user can edit an event using the event name, description, start date, end date, and status.

  - The user can delete an event using the event name, description, start date, end date, and status.
  - The user can optionally link their google account to retrieve their events from their google calendar.

2 Tech Stack

2.1 Frontend

- React
- Redux
- TailwindCSS
- Graphql
- JWT
- Google Calendar API
- Vercel (Deployment)

2.2 Backend

- Golang
- Gorm
- Gin Framework
- Graphql
- JWT
- Google Calendar API
- gcloud app engine (Deployment)
- gcloud sql (Postgres Database)

3 Other Details

3.1 Journey

I started on this project with the intention of learning more about other web frameworks as well as implementing the best practices while developing a full stack web application, hence, I decided to step out of my comfort zone and work with frameworks that are relatively more difficult like implementing using Go, redux as well as deploying the app to google cloud.

At the start, I implemented the backend using a restful framework, but I realised that such a framework would be too limiting for implementing batch operations as well as for relational databases, such as creating a task that has several tags as well as a list that it belongs to. If implemented using a restful framework, the frontend would first has to create a task, the tags needed, as well as the list and then finally add all tags to the task and add the task to the list. This proves to be rather inefficient as the frontend would have to make multiple requests to the backend. On the other hand, implementing it using a graphql framework would be much more efficient as the entire process would be handled by the backend and the frontend would only need to send a single request to it.

Another goal that I wanted to achieve was to make the code very modular and testable, and to be able to achieve a 100% code coverage.

3.2 Roadmap

- **User Authentication**
  - Implement JWT authentication for the user.
  - Implement Google Calendar API for the user.
- **Task Management**
  - Implement a task management system.
  - Implement a task management system with a label system.
  - Implement a task management system with a list system.
- **Event Management**
  - Implement an event management system.
  - Implement an event management system with a google calendar link.
- **Testing**
  - Write unit tests for the backend.
  - Write unit tests for the frontend.
  - Write integration tests for the frontend.
- **Deployment**
  - Deploy the application to Vercel.
  - Deploy the application to Google Cloud.
- **Migration from rest framework to graphql**
  - Rewrite both the frontend and backend to use graphql.