CVWO Mid-Assignment Writeup

Chan Chung Loong (A0240830R)

Overview

This assignment aims to create a task management web application to help users keep track of their to-do items in a quick and efficient manner. The application is built using both Golang for the backend and React for the frontend. Golang was chosen due to its simplicity and readability when writing different functions required. The database used to store all information is MySQL, which is chosen for its simplicity and fast deployment using XAMPP locally in the development environment.

Basic Use Cases

1. User Authentication
    a. Any user should be able to register for an account and login to the application, ensuring that they can only see their own task lists.
    b. The session should be persistent even after closure of browser tabs and automatically expire after a certain time period
2. Task Management
    a. Create any new tasks easily with its own description, name and category label
    b. Read all tasks linked with the user account and display in a readable format to the user
    c. Update individual tasks by editing details and saving it at the click of the button
    d. Delete any unwanted tasks after the task is complete
3. Sorting
    a. Sort tasks based on the categories input by the user
    b. Display selected tasks only when user filters based on categories
    c. Sorting done in the frontend to prevent additional strains to the backend server
    d. Case-insensitive

Target Audience

The target audience are users seeking for a platform to create, update and delete any potential tasks as a way to keep track of their schedule

Execution Plan

1. Initialization
    a. Create a new Golang project and React frontend using the available scripts provided by React.
    b. Start a new MySQL database using XAMPP as a testing platform and connect the two ends together using the proxy feature in React
    c. Initialize the different tables and setup the env credentials required for connection to the database in the Golang backend
2. Database Connection
    a. Connect Golang backend to the database, and write functions to perform CRUD operations using SQL
    b. Setup the different API routes for the frontend to call and the different JSON outputs to be sent back to the server
    c. Tag each task with a specific user ID generated by the database
3. React frontend
    a. Use bootstrap to style the different modals, cards required
    b. Setup different functions to interact with the backend and interpret the retuned result
    c. Write a function to sort the different cards based on the category label and only display the selected tags when required
4. User Authentication
    a. Register a new user by inserting the email and password (hashed using Bcrypt) into the database
    b. Compare the user information when the login function is called and returned a Boolean to determine successful authentication
    c. Create a JWT and set the token ID into a cookie that is sent and persisted in the client computer
    d. When server receives a new request, check for token validity and read the content of the cookie, display the email in the page and retrieve all tasks linked to the particular user ID. If the validity of the token has expired, automatically redirect the user back to the login page.

Deployment

The web application is deployed using Heroku for its ease-of-use and fast deployment. The React frontend is deployed in production mode and served by the Golang backend as static files. The MySQL database is currently instanced using AWS RDS free tier. The entire project can be hosted on a single EC2

The code is currently available on GitHub, with Heroku set to publish the latest commits from the repository automatically.

Future Plans

1. Typescript
    a. Implement Typescript to reduce the frequency of type-related errors and ensure scalability in the future
2. Docker
    a. Containerise the whole application for easy maintenance when deploying on different environments. Since the environment that the assignment was created is in Windows, having Docker would be useful if the project was moved to a different platform such as Linux
3. React
    a. Rewrite parts of the frontend into separate components for modularity and scalability in the future
    b. Add form validation to prevent missing input to the backend
    c. Improve on other features, such as adding deadlines, completion status
4. Golang
    a. Add in form validation to ensure input is valid and not susceptible to SQL injection
    b. Check for unique emails when user register for a new account
5. GitHub
    a. Rewrite the README.md with details related to the assignment such as user guide and screenshots.
6. Overall
    a. Add in comments for readability purposes and ensure variable names make sense