CVWO Mid-Assignment Write-Up

Name: Jovyn Tan Li Shyan

Matriculation Number: A0225607H

Link to GitHub repository: https://github.com/jovyntls/task-manager-app

Link to working application: https://task-manager-jtls.netlify.app/

Basic Uses Cases

- User authentication
  - Users create an account with a username and password, so that tasks can be saved to their accounts.
  - Data validation ensures that usernames and passwords are appropriate
- Task creation
  - Users can create, read, update and delete tasks.
  - Tasks can be assigned different levels of priority.
  - Tasks can be marked as completed or uncompleted.
- Task management
  - Categories: To compartmentalise tasks
    - Because reaching a goal often requires several smaller tasks, users can organise their tasks into categories.
    - Users can create, read, update and delete categories.
    - When tasks are displayed, they are organised by category, so that users can easily see the bigger picture of their to-dos.
  - Tags: To show connections between categories
    - Users can organise categories by adding tags to them. This helps users identify similar categories, or even highlight important categories.
    - Users can create and delete tags.
- Task organisation (currently not implemented)
  - Users should be able sort tasks within a category, either by date created or by priority.
  - Users should be able to filter categories by tags
  - When a category (and along with it, all associated tasks) is deleted, users should be able to undo this action
- User experience (currently not implemented)
  - Users should be able to toggle dark mode on/off.
  - User viewing preferences (dark mode on/off, sorting, filters) should be cached, so the user does not need to manually toggle every time.
  - Users should stay logged in after page refresh, unless they sign out.

Execution Plan

Type	Done	Item
Authentication	✓	Log-in and sign-up pages
	✓	Data validation: unique and case-insensitive usernames, length restrictions of usernames and passwords
	✓	Keep user logged in, using JSON Web Tokens (JWT)
	✓	Sign out button
Tasks	✓	CRUD for tasks
	✓	Add and update priority
	✓	Add and update status (completed/uncompleted)
		Sort by priority (asc/desc)
Categories	✓	CRUD for categories
	✓	Menu dropdown for more options
		Undo after deleting a category
		Clear completed tasks from a category
Tags	✓	Create and delete tags
	✓	Data validation: tags should be unique
	✓	Add tags to and remove tags from a category
		Edit tag names
		Allow filtering of categories by tags
UI/UX	✓	Responsive layout based on viewport size
		Compatibility and testing with different browsers: Google Chrome, Firefox, Internet Explorer, Safari, Opera
		Dark mode toggle
		Fix accessibility issues
Code Quality		Organise CSS and JSX files
		Add Redux state management
Deployment	✓	Deploy front-end and back-end

Problems I'm currently facing:

- **Issue**: (Frontend) Passing of data through props, and triggering refresh of parent components, is messy
    - **Difficulties**: I am not familiar with the best practices to handle component states and data from an API.
    - **Planned solution**: I hope to add Redux to my application as having a single state may make the frontend neater without having to pass props and refresh data between parent and child components. As I am completely unfamiliar with redux, I hope this will not take too much time and prevent me from implementing other functionalities that I hope to add.
- **Issue**: Unsure of best practices on organising data
    - **Difficulties**: I am often not sure if I am implementing features the right way. For instance, I am unsure if I should call all the data from the API at once, and then sort and filter data in the frontend; or make several small API calls for specific data.
    - **Planned solution**: I will try to read up more on the best practices of web development to improve the implementation of my application.
- **Issue**: Messy organisation of JSX components and CSS files in the frontend
    - **Difficulties**: I am not familiar with the best practices of structuring and organising components.
    - **Planned solution**: After implementing Redux, I will try to restructure my project in a sensible and understandable manner. I will also try to look at other open-source React apps to understand how to properly organise my repository.
- **Issue**: Poor user experience caused by some lag in CRUD operations
    - **Difficulties**: As time is taken to call the API, there is a lag between an action (e.g. user creates a new task), and the action being completed (e.g. user is able to see the new task created).
    - **Planned solution**: I am not sure how exactly to fix this. I considered rendering a task before it has been created to optimise user experience, but this may lead to a discrepancy between the data in the frontend and backend, which may become buggy and confusing for users.
- **Issue**: Not sure how to test browser compatibility with Internet Explorer
    - **Difficulties**: In the past, I have tried and failed to download Internet Explorer for MacOS.
    - **Planned solution**: I will be exploring alternative ways to download Internet Explorer again, or ask a friend to help me test a deployed version of the app. In general, I will avoid any browser-specific CSS, and hopefully there will be no other challenges to compatibility with other browsers.