Task: Web-based Task Management Dashboard

Objective:

Develop a web-based dashboard for managing tasks, ensuring it's responsive across devices.

UI Design Planning:

- **Initial Vision:** So, first, I imagined how the dashboard should look on the screen.
- **Sketching :** Then, I started with some rough sketches and improved them

Folder Structure:

- **Backend Organization:** To keep things organized in the backend, I created folders for different parts of the application. I had folders for tasks, and managing the database.(MVC)
- **Frontend Structure:** Similarly, on the user interface side, I organized different components of the dashboard, like task lists and forms, into their own folders. This way, it was clear where to find code related to each part of the dashboard.

API's Structure:

- **Fetch Tasks:** To get the list of tasks when you use the app, we have this special address (URL): http://localhost:3000/tasks.
- **Create Task:** When we add a new task, the data goes here: http://localhost:3000/tasks.
- **Update Task:** If we edit a task, the changes are sent to this place: http://localhost:3000/tasks/:taskId.
- **Delete Task:** When we delete a task, we use this URL: http://localhost:3000/tasks/:taskId.
- **Get Task by ID:** And if we want to see the details of a single task, we can use this: http://localhost:3000/tasks/:taskId.

(frontend/TaskEdit.js)

- **Submission**: When the user submits the form by clicking the "Update" button, the handleUpdate function is called. It sends the updated task data to the server and triggers a success or error message using SweetAlert2 to provide feedback to the user.(frontend/TaskEdit.js)
- **Min Date**: The "Due Date" input field is restricted to dates on or after the current date, ensuring that users cannot select past dates.

(frontend/TaskForm.js)

- **Form Initialization**: The form adapts for task creation or editing based on provided data.

- **Date Formatting**: Dates are converted to "dd-mm-yyyy" format for backend consistency.
- **User Input Handling**: Changes in form fields are tracked in real-time.
- **Form Submission**: Handles form submission, including date format conversion.
- **Minimum Date**: Prevents selection of past dates in the "Due Date" field.
- **Conditional Button Label**: Button label changes based on task creation or update.

MongoDB Issue (Resolved):

- **Challenge:** The issue was that the connection didn't work on Windows because in previous organization I was working on linux now on windows there was a mismatch between Ip's and paths and mongodb was not running on localhost string so I addressed the issue and matches the URL's of Connection strings of MongoDB compass and Mongodb setup in backend.
- **Solution:** To fix it, I went through the project settings and updated the connection to work on Windows.
- **Result:** After this fix, my app properly save and retrieve data from the database.
- **Lesson Learned:** This experience taught me the importance of checking and adjusting settings when moving the project between different types of systems.

Enhancing Security with SweetAlert Confirmation:

- **Idea:** I wanted to make sure we don't accidentally do things like deleting tasks, so I added a feature.
- **Challenge:** The challenge was to prevent accidents and make sure users are aware of what they're doing.
- **Solution:** I introduced SweetAlert2 dialogs with "Yes" and "Cancel" options. Now, when we try to delete something, it asks for your confirmation.
- **Benefits:** This improved security and made the app more user-friendly. Our data is safer too.
- **Lesson Learned:** I realized that confirmation dialogs are a great way to enhance both security and usability.

Development Efficiency and User Experience Optimization:

- **Idea:** I wanted to make the project development smoother and enhance the user experience.
- **Highlights:** I placed console statements at strategic points in the code to quickly spot and fix errors during development. Also, I implemented client-side routing, so page transitions are smooth without interruptions.
- **Benefits:** This approach accelerated issue resolution and provided a better user experience with seamless page transitions and reduced interruptions.
- **Why it Matters:** These optimizations made our project more efficient and enjoyable to use.

X-Correlation-ID Header in Postman:

- **What it Is:** In our Postman testing tool, I added a special X-Correlation-ID header with a unique value ({{guid}}) for each API request. This helps us keep track of and understand requests when testing and troubleshooting in a distributed system.