Project link kanban : https://github.com/users/diogorangel/projects/5

Link Repository : https://github.com/diogorangel/WDD430-Group-Project.git

Live project : https://diogorangel.github.io/WDD430-Group-Project/

I apologize for the misunderstanding. When you asked to "remove src folder of the document above," I thought you were referring to the React code structure I had just provided.

You are now requesting to see the entire combined document containing the "Project Status Meeting Summary" and the "React Project Conversion Plan," but with the file structure I provided excluding the src folder.

Here is the complete, consolidated document:

📝 **Project Status Meeting Summary**

This meeting focused on reviewing the current status of the project repository (as indicated by the git status report) and planning the next steps, particularly addressing the large number of modified, deleted, and untracked files in the context of a new project framework (React).

👥 **Participants**

- **Diogo Rangel Dos Santos**

- **[Participant 2 - e.g., Developer]**

- **[Participant 3 - e.g., Documentation Specialist]**

- **[Participant 4 - e.g., Design Specialist]**

🎯 **Key Outcomes & Analysis**

✅ **Success: Successful Project Migration**

The team has successfully completed a significant project migration/restructuring, evidenced by the large number of deleted files (css/style.css, scripts/script.js) and new, untracked directories and configuration files (app/, eslint.config.mjs, next.config.ts, package.json, etc.). This indicates a clear move from a vanilla JavaScript project to a modern, framework-based structure (likely Next.js/React based on the files).

🚧 **Challenge: Managing File Deletions and Junk Files**

The primary challenge is the volume of uncommitted changes. The presence of the temporary file ~$D 430 Group Project - Diogo Rangel Dos Santos.docx needs to be addressed immediately via deletion and the proper implementation of the new .gitignore to maintain a clean repository.

✨ **Insight: Importance of Versioning Configuration**

The number of new, untracked configuration files highlights that the project is adopting a robust build system. The immediate next action must be to stage and commit all these untracked configuration files first, as they define the new environment and are critical for all developers.

⚛️ React Project Conversion Plan

This plan details the conversion of the previous project's objectives (a simple ToDo list) into a React Component Structure, maintaining the original user story and design specifications.

👥 Group & Context

- **Group Members: WDD 430 Team 03 - Wednesdays 17:00 (UTC-0) / Diogo Rangel Dos Santos**

- **User Story: As a beginner user, I want to add and remove tasks from a list, so that I can organize my small studies and project activities.**

🎨 Design & Style Specifications (React/CSS)

Element	Specification	React Implementation
Background	#f4f4f4 (Light Gray)	Applied to the global style file (index.css).
Primary	#2980b9 (Blue, for actions)	Used for the Add Button and key interactions.
Secondary	#ecf0f1 (Soft Gray, for items)	Used for the background of individual Task Items.
Font	Arial, sans-serif	Applied globally via CSS.
Layout	Centered, max width of 600px.	Applied to the main ToDoApp container component.

🧱 Component and Task Breakdown (React Focus)

#	Work Item	Status	React Component/File
1	Initial Setup: Create React project using vite or create-react-app.	Completed	package.json, index.html, etc.
2	Core Logic: Implement state management for the task list (useState Hook).	Updated	components/ToDoApp.jsx (Central State Manager)

#	Work Item	Status	React Component/File
3	Base Styling: Apply global and component-specific styles.	Completed	index.css (Global) and *.module.css (Component Styles)
4	Input Component: Create a controlled component for text input.	Updated	components/TaskInput.jsx
5	Action Button: Implement the 'Add' functionality (passed as a prop).	Merged into 4	components/TaskInput.jsx
6	List Display: Create a component to render the list of tasks.	Updated	components/TaskList.jsx
7	Task Item Component: Create a reusable component for a single task.	New Component	components/TaskItem.jsx
8	Add Function (React): Pass the addTask function (from State) down to the Input component.	Completed	State logic in ToDoApp.jsx
9	Delete Function (React): Pass the deleteTask function (from State) down to the Task Item component.	Completed	Logic in TaskItem.jsx / State logic in ToDoApp.jsx
10	Data Persistence: Use the useEffect Hook to save and load tasks from Local Storage.	Pending	ToDoApp.jsx or Custom Hook
11	Base Responsiveness: Use media queries in CSS files to ensure layout flexibility.	Pending	*.module.css files

⚙️ Final Component File Structure

The application files are located in the root directory and a components sub-folder, adhering to your request to remove the src/ wrapper.

/ (Root Directory)

├── components/

| ├── ToDoApp.jsx   // Main container. Manages task state and layout.

| ├── TaskInput.jsx   // Input field and "Add" button.

| ├── TaskList.jsx   // Maps through the tasks array.

| └── TaskItem.jsx   // Displays a single task and its "Delete" button.

├── index.css   // Global styles.

├── **App.jsx**   // **Renders the ToDoApp component.**

├── **main.jsx**   // **Renders the root React component.**

├── **package.json**

└── **... (other config files)**