# Task Management Dashboard

## 1. Design Choices

The Task Management Dashboard was designed with a focus on clean, maintainable, and reusable code. The architecture follows React best practices, using small, focused components and a global state management approach with React Context + useReducer for predictable state updates. TailwindCSS was chosen for utility-first styling to speed up development and maintain consistency. The application structure ensures separation of concerns, keeping components, hooks, types, and utilities organized.

## 2. Libraries Used and Why

* React: For building the component-based UI.
* TypeScript: For static typing, which improves maintainability and reduces bugs.
* Vite: For its fast development server and optimized build process.
* TailwindCSS: For utility-first styling, enabling quick and consistent UI development.
* React Context + useReducer: For centralized state management and predictable updates.
* localStorage API: For persisting tasks across sessions without a backend.

## 3. Challenges and Solutions

* State Management: Ensuring global state is managed efficiently without prop drilling was solved using React Context with useReducer.
* Persistence: Tasks needed to persist between sessions. This was handled using the localStorage API with proper serialization/deserialization.
* Accessibility: Making the UI accessible required semantic HTML, ARIA attributes, and keyboard navigation support.
* Advanced Patterns: Implementing Compound Components and Higher-Order Components to meet advanced requirements while keeping the code clean and modular.