High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **TaskFlow - To-Do List Manager**. TaskFlow is a modern, open-source to-do list application enabling users to create, edit, complete, and filter tasks via a clean React-based UI.

---

1. System Architecture Overview

Architecture Description:
TaskFlow is a single-page application (SPA) built with React and TypeScript. All logic and data are managed client-side, with persistent storage using browser local storage. The UI is styled using Tailwind CSS.

Main System Modules

Module	Role/Responsibility
UI Layer	Renders components, handles user interactions
State Management	Manages task data and UI state (React state/hooks)
Data Persistence	Stores/retrieves tasks in browser local storage
Filtering Logic	Filters tasks by status (all/active/completed)

---

2. Component Interactions

Interaction Step	Description
User Action (UI Layer)	User creates/edits/completes tasks via UI components
State Update (State Management)	UI dispatches actions to update React state
Data Sync (Persistence)	State changes are saved to/retrieved from local storage
Filter Application	Filtering logic updates visible tasks in UI

Sequence Flow:

1. User interacts with UI (e.g., adds a task)
2. State is updated via React hooks
3. Updated state is persisted to local storage
4. UI re-renders with new/filtered task list

---

3. Data Flow Overview

Source	Data	Destination	Purpose
UI Components	Task input/actions	State Management	Create/update/delete tasks
State	Task list	Local Storage	Persist tasks
Local Storage	Task list	State Management	Load tasks on app start
State	Filtered task list	UI Components	Display tasks per filter

---

4. Technology Stack

Layer/Function	Technology/Framework
Frontend Framework	React (with TypeScript)
Styling	Tailwind CSS, HTML, CSS
State Management	React state/hooks
Data Persistence	Browser Local Storage
Tooling	Node.js, npm/yarn (build)

---

5. Scalability & Reliability

- **Scalability:**
  Designed for single-user, client-side use; no backend required. Can be extended to use remote storage or APIs if needed.
- **Reliability:**
  Data is persisted in local storage for offline access. Minimal risk of data loss unless browser storage is cleared.
- **Security:**
  No sensitive data handled; local storage only. For multi-user or cloud sync, authentication and secure APIs would be required.

End of Document