- Final Demo Walkthrough

```
// Single-file React component (default export). Tailwind CSS classes are used for styling.

// How to use:

// 1) Create a new Vite React project: `npm create vite@latest my-todo -- --template react` or use CRA.

// 2) Replace `src/App.jsx` with the contents of this file and ensure Tailwind is set up.

//    (If you don't have Tailwind, the component includes a small fallback style block at the top.)

// 3) Run `npm install` (if needed) and `npm run dev` / `npm start`.

// 4) Open the app in the browser. To capture a screenshot: use the browser's screenshot or devtools -> run command -> Capture screenshot.


import React, { useEffect, useState, useRef } from 'react';


export default function TodoApp() {

 // Feature set:

 // - Add / remove todos

 // - Edit todo in-place

 // - Toggle complete

 // - Filter (All / Active / Completed)

 // - Persist to localStorage

 // - Keyboard accessible (Enter to add, Esc to cancel edit)


 const STORAGE_KEY = 'react-hooks-todos-v1';

 const [todos, setTodos] = useState(() => {

  try {

   const raw = localStorage.getItem(STORAGE_KEY);

   return raw ? JSON.parse(raw) : sampleTodos();

  } catch (e) {

   return sampleTodos();
```

```
  }
});
const [text, setText] = useState('');
const [filter, setFilter] = useState('all'); // all | active | completed
const [editingId, setEditingId] = useState(null);
const [editingText, setEditingText] = useState('');
const inputRef = useRef(null);

useEffect(() => {
  localStorage.setItem(STORAGE_KEY, JSON.stringify(todos));
}, [todos]);

function sampleTodos() {
  return [
    { id: id(), text: 'Buy groceries', completed: false },
    { id: id(), text: 'Finish React Hooks assignment', completed: true },
    { id: id(), text: 'Walk the dog', completed: false },
  ];
}

function id() {
  return Date.now().toString(36) + Math.random().toString(36).slice(2, 7);
}

function addTodo(e) {
  e?.preventDefault();
  const trimmed = text.trim();
  if (!trimmed) return;
  setTodos(prev => [{ id: id(), text: trimmed, completed: false }, ...prev]);
```

```
  setText('');
  inputRef.current?.focus();
}

function removeTodo(idToRemove) {
  setTodos(prev => prev.filter(t => t.id !== idToRemove));
}

function toggleTodo(idToToggle) {
  setTodos(prev => prev.map(t => (t.id === idToToggle ? { ...t, completed: !t.completed } : t)));
}

function startEdit(todo) {
  setEditingId(todo.id);
  setEditingText(todo.text);
}

function cancelEdit() {
  setEditingId(null);
  setEditingText('');
}

function saveEdit(idToSave) {
  const trimmed = editingText.trim();
  if (!trimmed) {
    // if text becomes empty, delete the todo
    removeTodo(idToSave);
    cancelEdit();
              />
```

```
            {editingId === todo.id ? (
              <div className="flex-1 flex items-center gap-2">
                <input
                  autoFocus
                  value={editingText}
                  onChange={e => setEditingText(e.target.value)}
                  onKeyDown={e => {
                    if (e.key === 'Enter') saveEdit(todo.id);
                    if (e.key === 'Escape') cancelEdit();
                  }}
                  className="flex-1 px-3 py-2 rounded border border-slate-200"
                />
                <button onClick={() => saveEdit(todo.id)} className="px-3 py-1 rounded bg-emerald-500 text-white">Save</button>
                <button onClick={cancelEdit} className="px-3 py-1 rounded border">Cancel</button>
              </div>
            ) : (
              <>
                <div className="flex-1 flex items-center justify-between">
                  <label htmlFor={`chk-${todo.id}`} className={`cursor-pointer select-none ${todo.completed ? 'line-through text-slate-400' : ''}`}>
                    {todo.text}
                  </label>
                  <div className="flex items-center gap-2 ml-3">
                    <button onClick={() => startEdit(todo)} aria-label={`Edit ${todo.text}`} className="text-sm px-2 py-1 rounded border">Edit</button>
                    <button onClick={() => removeTodo(todo.id)} aria-label={`Delete ${todo.text}`} className="text-sm px-2 py-1 rounded border">Delete</button>
```

```
      <small className="block mt-3 text-slate-500">Tip: press Enter to add a task. Double-click Edit to ... well, click the Edit button 😄</small>
    </div>
  </div>
 );
}
function FilterButton({ children, active, onClick }) {
 return (
  <button
   onClick={onClick}
   className={`px-3 py-1 rounded ${active ? 'bg-sky-600 text-white' : 'border'}`}
  >
   {children}
  </button>
 );
}
/*
 - Expected look: a centered card with header "Todo — React Hooks". The list shows sample todos.
```

- Project Report

□ Full-Stack To-Do App Summary

🎯 Objective

A clean, efficient To-Do app for managing personal, academic, or professional tasks using **React (frontend)**, **Node.js + Express (backend)**, and **MongoDB (database)**.

⚙️ Core Tech Stack