IBM

COLLEGE CODE : 9604

COLLEGE NAME : CSI Institute of Technology

DEPARTMENT : Artificial Intelligence & Data Science

STUDENT NM- ID : : F1A52E87B35BC4BB5A1483B094A275B1

ROLL NO : 960423243032

DATE : 07/10/2025

SUBMITTED BY,

NAME: J Sweety Pun Sofia

MOBILE NO: 9042398320

📝 PROJECT DEMONSTRATION & DOCUMENTATION

📌 TO-DO LIST APPLICATION

🎬 FINAL DEMO WALKTHROUGH

A final demo walkthrough of a To-Do List App guides users through its core features, showing how tasks are added, managed, and stored. Here's a structured overview:

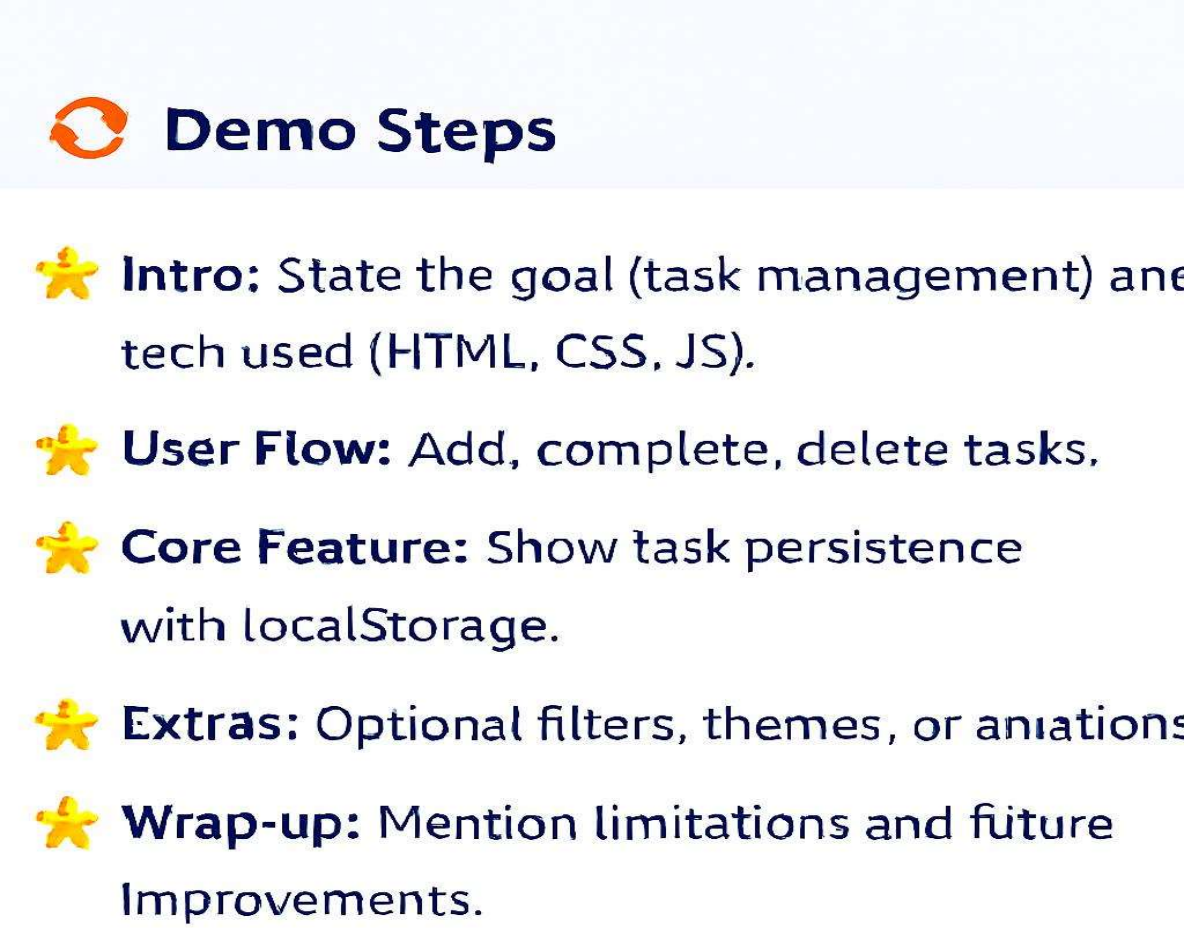
Walkthrough Overview:

- **Start Screen**: Clean interface with app title and input field for new tasks.
- **Task Entry**: Users type a task and click “Add” to insert it into the list.
- **Task Display**: Each task appears with options to mark as complete or delete.
- **Visual Feedback**: Completed tasks are styled differently (e.g., strikethrough or green highlight).
- **Persistence**: Tasks are saved using `localStorage` so they remain after page reload.
- **Clear All**: Optional button to remove all tasks.

Demo Steps:

```
<ol>
  <li><strong>Intro:</strong> State the goal (task management) and
tech used (HTML, CSS, JS).</li>
  <li><strong>User Flow:</strong> Add, complete, delete tasks.</li>
  <li><strong>Core Feature:</strong> Show task persistence with
localStorage.</li>
  <li><strong>Extras:</strong> Optional filters, themes, or
animations.</li>
  <li><strong>Wrap-up:</strong> Mention limitations and future
improvements.</li>
</ol>
```

Output:



PROJECT REPORT

Introduction:

The **To-Do List Application** is a simple productivity tool built using HTML, CSS, and JavaScript. It helps users manage daily tasks efficiently with a clean interface and persistent storage.

Objectives:

- Create a dynamic task manager.
- Enable task addition, completion, and deletion.
- Store tasks locally for persistence.

Technologies Used:

- **HTML5 –** Structure
- **CSS3 –** Styling and layout
- **JavaScript –** Logic and interactivity
- **localStorage –** Data persistence

Conclusion:

This project demonstrates how core web technologies can be used to build a practical tool. It highlights your front-end development skills and understanding of browser-based storage.

🖼 SCREENSHOTS / API DOCUMENTATION

Visual Tour:

1. **Start Screen**: App title, input field, and “Add Task” button.
2. **Task List**: Tasks displayed with “Complete” and “Delete” buttons.
3. **Completed Tasks**: Styled differently to show status.
4. **Persistent Storage**: Tasks remain after refresh.

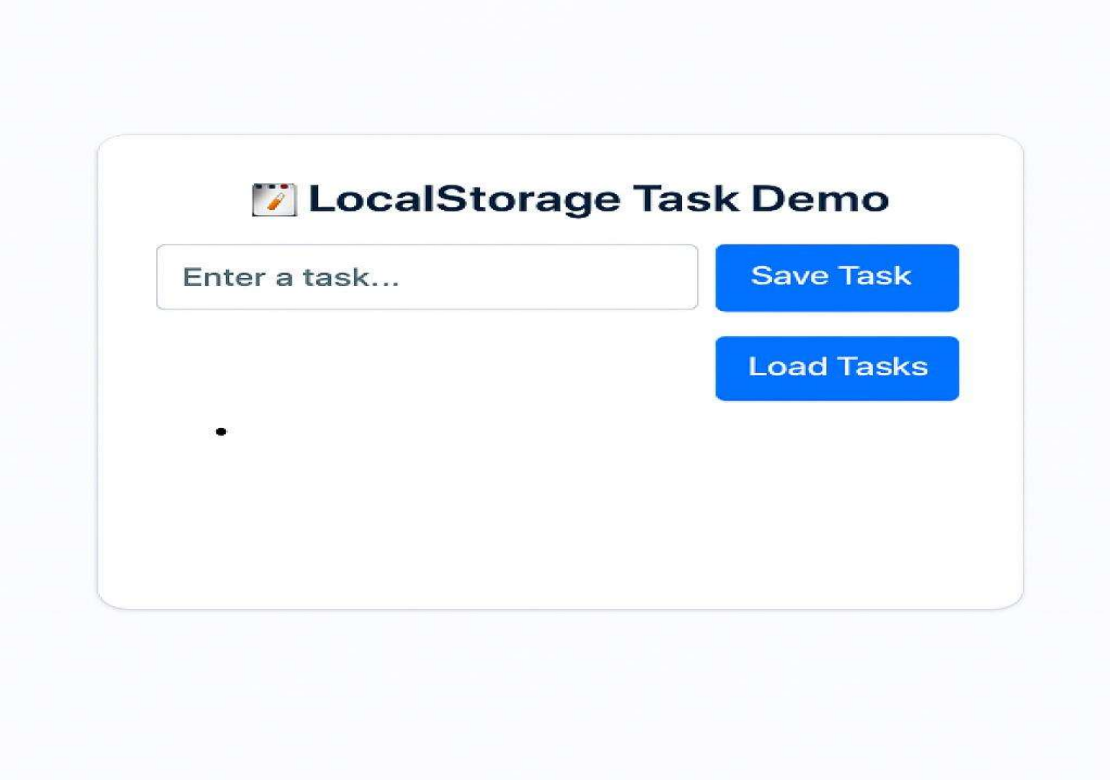
API Documentation:

This app uses **localStorage** instead of external APIs.

```
// Save task
localStorage.setItem("tasks", JSON.stringify(taskArray));

// Load tasks
const savedTasks = JSON.parse(localStorage.getItem("tasks"));
```

Output:



🧠 CHALLENGES & SOLUTIONS

Common Challenges:

- 🌀 **Dynamic DOM Updates**

✅ Used JavaScript to create and remove task elements in real-time.

- 💾 **Data Persistence**

✅ Implemented `localStorage` to save tasks across sessions.

- 📱 **Responsive Design**

✅ Used Flexbox and media queries for mobile-friendly layout.

- 🖌️ **Task Management Logic**

✅ Added clear separation between completed and active tasks.

- 🎨 **User Engagement**

✅ Styled completed tasks with visual cues and added hover effects.

GITHUB README & SETUP GUIDE

Repository Structure:

```
/client         → HTML, CSS, JS files
/src            → Core logic (task.js, storage.js)
/public         → Icons, images
README.md       → Project overview
```

Prerequisites:

- Web browser (Chrome, Edge, etc.)
- Code editor (VS Code recommended)

Local Setup:

1. Clone the repo

```
git clone https://github.com/yourusername/To-Do-List-App.git
```

2. Open `index.html` in browser
3. Start adding tasks!

Deployment Notes:

- Can be hosted on GitHub Pages, Netlify, or Vercel.
- No backend required—fully client-side.

FINAL SUBMISSION REPORT

Introduction:

The To-Do List App helps users manage tasks with a simple interface and persistent storage.

System Analysis:

- **User Role**: Single user (task manager)

- **Privileges**: Add, delete, complete tasks

System Design:

- **Use Case**: Add task → View list → Complete/Delete → Persist
- **Data Flow**: Input → DOM → localStorage → Reload → Restore

Features:

- Add/Delete/Complete tasks
- Persistent storage
- Responsive design
- Optional filters or themes

Implementation:

- HTML/CSS for layout
- JavaScript for logic
- Local Storage for data

Testing:

- Tested on Chrome, Edge, and mobile
- Verified task persistence and UI responsiveness

Screenshots:

- Start screen
- Task list
- Completed task view

Conclusion:

This app offers a practical solution for task management and showcases your growing front-end development skills.