# EXPERIMENT 2

## Code and explanation

### 1. Adding Tasks Manually

```
const addTask = () => {
    if (input.trim() !== "") {
      dispatch({ type: "ADD_TASK", payload: input });
      setInput("");
    }
  };
```

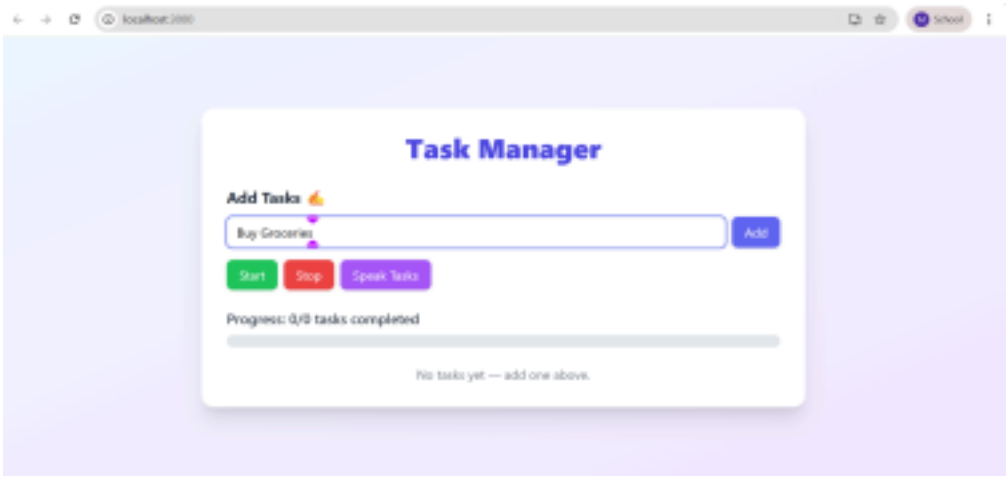



**Fig 1.1 - Adding tasks Manually**

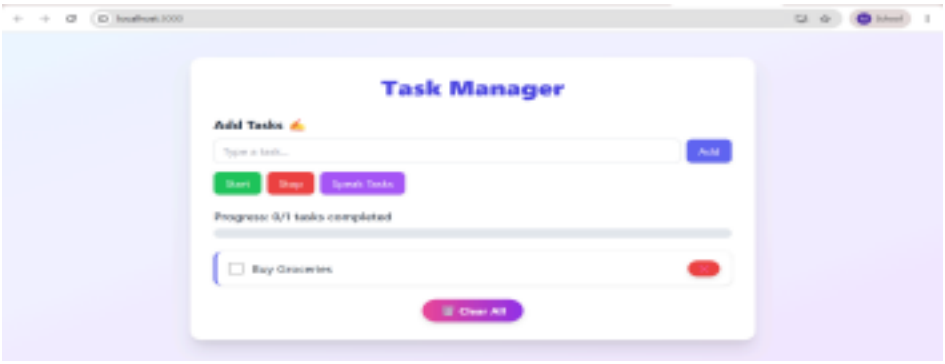



**Fig 1.2 - Tasks Added**

### 2. Adding Tasks through voice control (30% extra)

Adding tasks through voice control makes use of the Web Speech API's SpeechRecognition interface. Instead of typing into an input field, the user speaks a

command such as:

- *“Finish assignment”*
- *“Buy groceries”*

The application listens for these commands, transcribes the speech into text, and extracts the task description. This task is then added to the task list in the same way as manual input.

This improves accessibility (hands-free interaction for differently-abled users) and enhances user experience by making the application faster and more engaging.

```
import React, { useState } from "react";
import { useTasks } from "../context/TaskContext";

const VoiceControls = () => {
  const { tasks, dispatch } = useTasks();
  const [listening, setListening] = useState(false);
  const [input, setInput] = useState("");

  const startListening = () => {
    const recognition = new (window.SpeechRecognition ||
window.webkitSpeechRecognition)();
    recognition.lang = "en-US";
    recognition.start();

    recognition.onresult = (event) => {
      const transcript = event.results[0][0].transcript;
    dispatch({ type: "ADD_TASK", payload: transcript });
    };

    recognition.onend = () => setListening(false);
    setListening(true);
  };

  const stopListening = () => {
    setListening(false);
  };
```

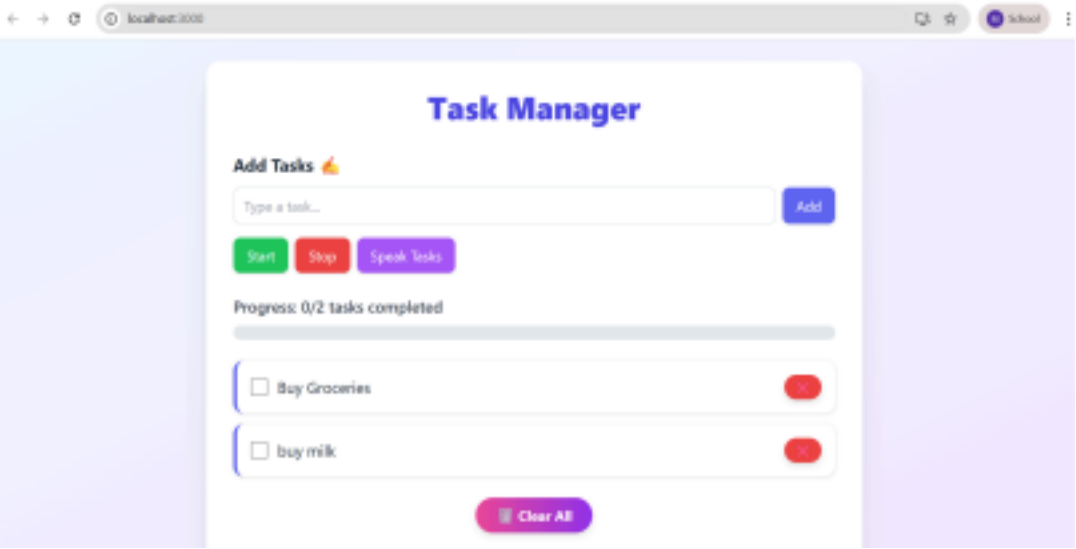



**Fig 2.1- Adding Tasks through voice input**

### 3. Voice Feedback (30% extra)

Voice feedback is implemented using the Web Speech API's Speech Synthesis interface.

- It enables the application to "speak back" to the user with natural-sounding speech.
- This provides auditory confirmation in addition to visual updates.
- Improves accessibility (useful for visually impaired users).
- Enhances user motivation and engagement (e.g., congratulatory messages when tasks are completed).

```
const readTasks = () => {
 if ("speechSynthesis" in window) {
   const pending = tasks.filter(t => !t.completed).map(t =>
t.text);
   const text =
     pending.length === 0
         ? "You have no pending tasks. Great job!"
          : `You have ${pending.length} pending tasks:
${pending.join(", ")}`;
     const utterance = new SpeechSynthesisUtterance(text);
      speechSynthesis.speak(utterance);
```

```
    }
  };
```

### 4. Progress Tracker

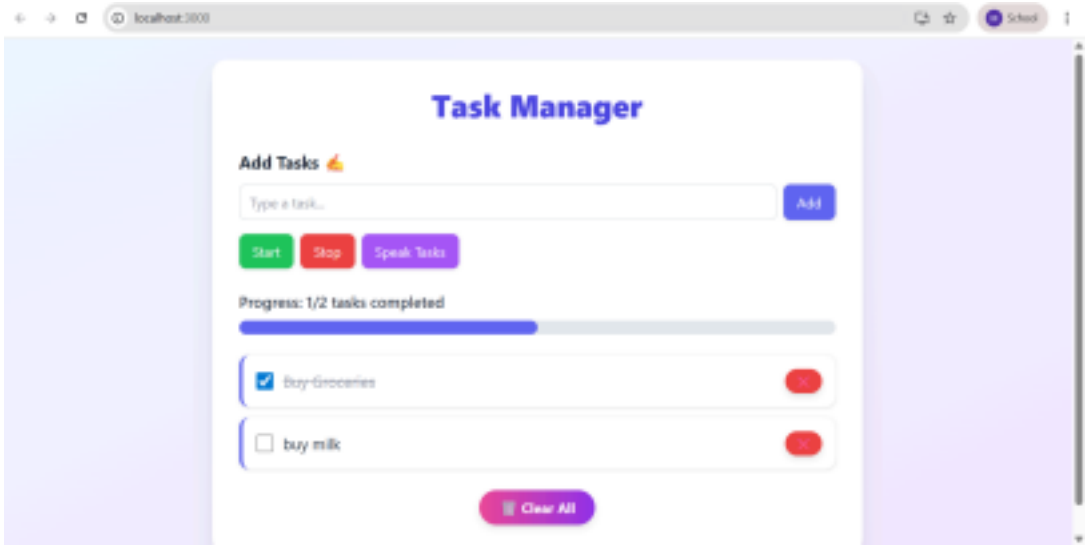
A progress tracker bar visually represents how many tasks the user has completed compared to the total tasks. It enhances motivation and provides instant feedback.

- Numerical feedback → e.g., *"Progress: 2/5 tasks completed"*.
- Visual feedback → a progress bar that fills as tasks are marked completed.

```
import React from "react";
import { useTasks } from "../context/TaskContext";

const ProgressTracker = () => {
  const { tasks } = useTasks();
  const completed = tasks.filter(t => t.completed).length;
  const total = tasks.length;
  const progress = total === 0 ? 0 : Math.round((completed /
total) * 100);

  return (
    <div className="my-6">
      <p className="text-lg font-semibold text-gray-700 mb-2">
        Progress: {completed}/{total} tasks completed
      </p>
      <div className="w-full bg-gray-200 rounded-full h-4">
        <div
          className="bg-indigo-500 h-4 rounded-full
transition-all duration-300"
          style={{ width: `${progress}%` }}
        ></div>
      </div>
    </div>
  );
};
export default ProgressTracker;
```


localhost:3000
Task Manager
Add Tasks
Type a task...
Add
Start
Stop
Speak Tasks
Progress: 1/2 tasks completed
Buy Groceries
buy milk
Clear All