**Aim:** Basic user interface design using react components, Stylesheet, Flexbox and XAML, manipulation of components using States and Props.

**Description:**

The goal of this project is to develop a basic mobile application with a user-friendly interface that demonstrates the usage of React components, Stylesheets, Flexbox, and XAML in a cross-platform development environment. The application will also show how states and props are used to manipulate and manage the UI components in both React (for web development) and .NET MAUI (for cross-platform mobile development).

**Objective:**

* React (Web/JSX): Build an interactive interface with React components using states and props for component data handling.
* .NET MAUI (XAML): Design an equivalent cross-platform mobile application with a UI that can run on Android, iOS, and Windows using XAML for UI layout and C# for logic.
* Learn how to integrate Flexbox for responsive layouts.
* Demonstrate the power of states and props to update and manage component behavior in the React app and use C# bindings in .NET MAUI to manage tasks.

**Tools Required:**

 Node.js and React Native CLI:

* Node.js for the backend JavaScript engine.
* React Native CLI for mobile development using JavaScript and JSX.

 Visual Studio 2022 with .NET MAUI:

* Visual Studio 2022 for building cross-platform mobile applications with .NET MAUI using XAML for the UI design and C# for logic implementation.
* .NET MAUI allows building applications that can run across multiple platforms like Android, iOS, and Windows.

 Code Editor:

* Visual Studio Code: For working on React components with JSX and Stylesheets.
* Visual Studio 2022: For working with XAML and C# in .NET MAUI.

 Android Studio (for Android Emulator):

* If testing on Android, Android Studio provides emulators or the use of a physical device for mobile testing.

**Implementation:**

1. React Web App Implementation:
   * Step 1: Set up a new React app using create-react-app.
   * Step 2: Create functional components for the app (e.g., Task Input, Task List).
   * Step 3: Implement states (for managing task data) and props (for passing data between components).
   * Step 4: Style the app with CSS (Flexbox layout) to make the UI responsive.
   * Step 5: Use React hooks (useState) to manage the state of the app.

1. Install Prerequisites

Before starting, make sure you have the following tools installed:

* Node.js: You can download it from [nodejs.org](https://nodejs.org).
* Visual Studio Code: Download and install from [here](https://code.visualstudio.com/).

**Create a new React app** using create-react-app

*npx create-react-app todo-app*

*cd todo-app*

**App.js**

import React, { useState } from "react";

import "./App.css";

const App = () => {

const [tasks, setTasks] = useState([]);

const [newTask, setNewTask] = useState("");

const addTask = () => {

if (newTask.trim()) {

setTasks([...tasks, { text: newTask, completed: false }]);

setNewTask("");

}

};

const toggleTaskCompletion = (index) => {

const updatedTasks = tasks.map((task, i) =>

i === index ? { ...task, completed: !task.completed } : task

);

setTasks(updatedTasks);

};

const removeCompletedTasks = () => {

setTasks(tasks.filter((task) => !task.completed));

};

return (

<div className="todo-app">

<h1>To-Do App</h1>

<div className="input-container">

<input

type="text"

value={newTask}

onChange={(e) => setNewTask(e.target.value)}

placeholder="Add a new task"

/>

<button onClick={addTask}>Add</button>

</div>

<ul className="task-list">

{tasks.map((task, index) => (

<li

key={index}

className={`task-item ${task.completed ? "completed" : ""}`}

onClick={() => toggleTaskCompletion(index)}

>

{task.text}

</li>

))}

</ul>

<button onClick={removeCompletedTasks} className="remove-btn">

Remove Completed Tasks

</button>

</div>

);

};

export default App;

**App.css**

.todo-app {

font-family: Arial, sans-serif;

margin: 20px auto;

max-width: 400px;

text-align: center;

}

.input-container {

display: flex;

justify-content: space-between;

margin-bottom: 10px;

}

.input-container input {

flex: 1;

padding: 8px;

margin-right: 5px;

}

.input-container button {

padding: 8px;

}

.task-list {

list-style: none;

padding: 0;

}

.task-item {

padding: 8px;

margin: 5px 0;

background: #f0f0f0;

cursor: pointer;

border-radius: 4px;

}

.task-item.completed {

text-decoration: line-through;

color: gray;

}

.remove-btn {

margin-top: 10px;

padding: 10px;

background: red;

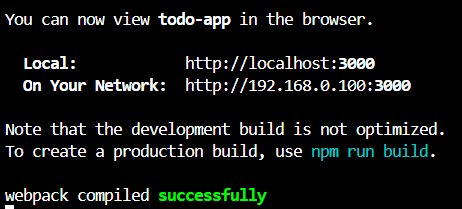
color: white;

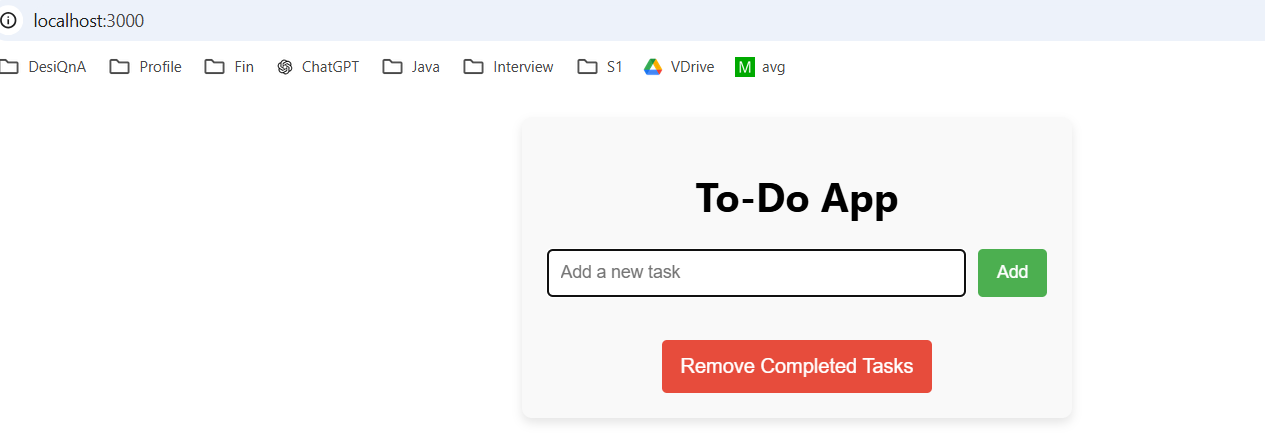
border: none;

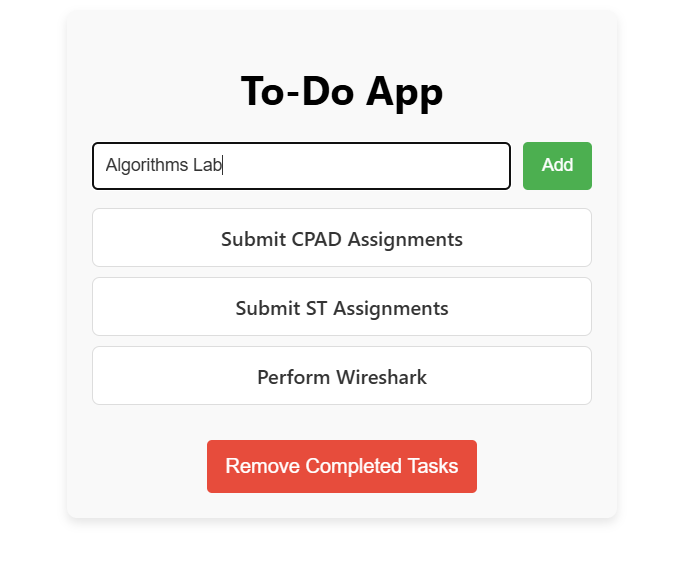
cursor: pointer;}

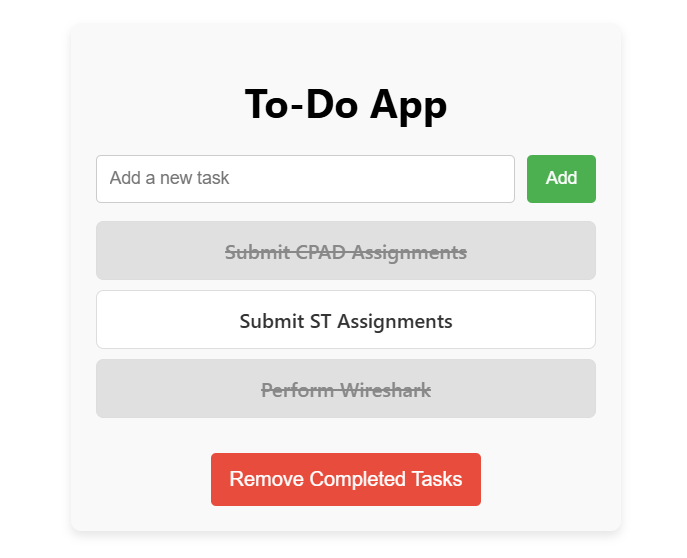
**4. Run and Test the React App**

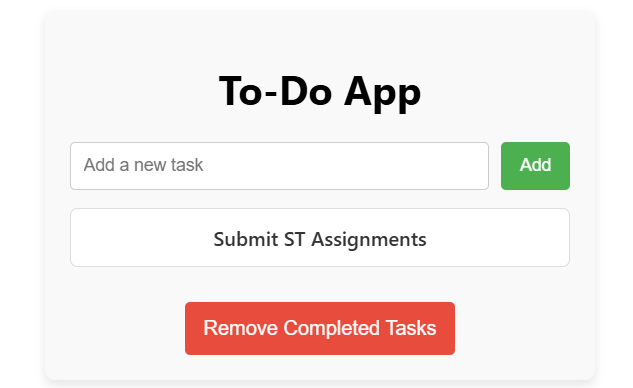
1. **Start the app** by running npm start in the terminal.
2. Open http://localhost:3000 in your browser to see the To-Do app.











**Conclusion:**

In this project, you’ve built a basic **To-Do App** using **React** for the web and **.NET MAUI** for cross-platform mobile development. By using **states and props** in React and **data binding** in .NET MAUI, you've learned how to manipulate and display components dynamically. You've also used **Flexbox** to create responsive layouts and **XAML** for designing a clean UI.

This project showcases how you can leverage different technologies to build apps for both web and mobile platforms, while managing states and props to manipulate the app components effectively.