

## Lesson 05 Demo 01

### Building a To-Do List Application Using GitHub Copilot

**Objective:** To demonstrate the code generation and enhancement of a To-Do List application using GitHub Copilot

**Tools required:** Visual Studio Code, Chat GPT and GitHub Copilot

**Prerequisites:** NA

Steps to be followed:

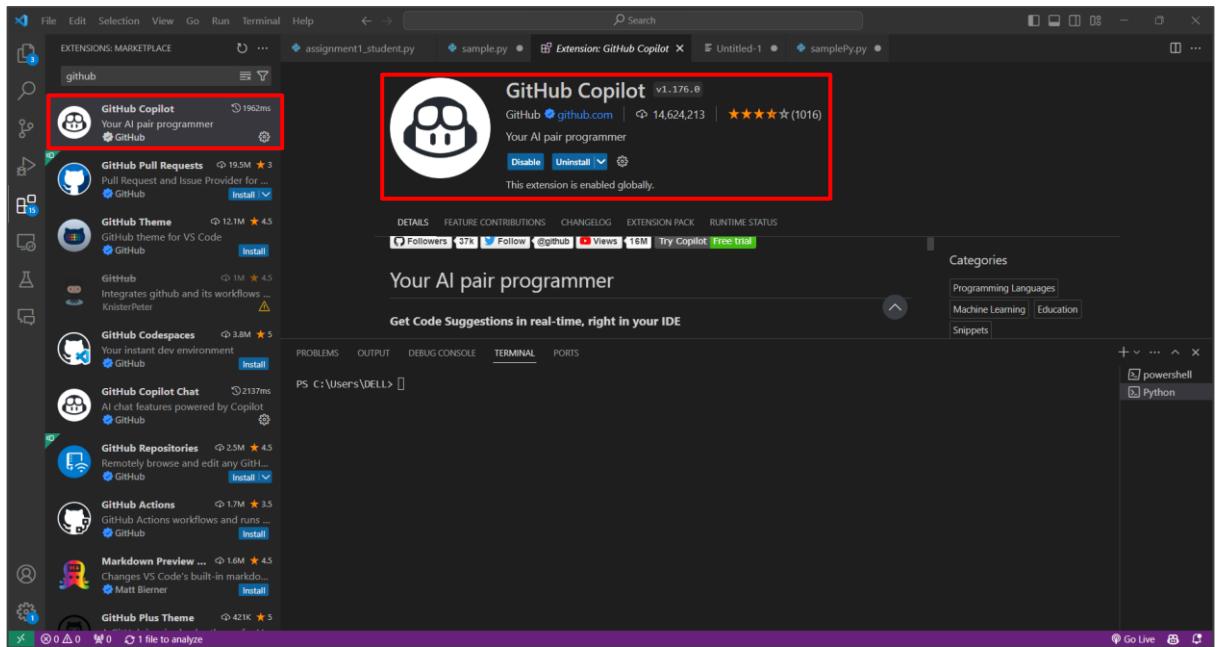
1. Integrate GitHub Copilot with Visual Studio
2. Create React JS Project with help of Chat GPT
3. Create User Defined component with help of GitHub Copilot which help To-Do List application.

**Note:** Create an account on <https://github.com/features/copilot> to access it in Visual Studio.

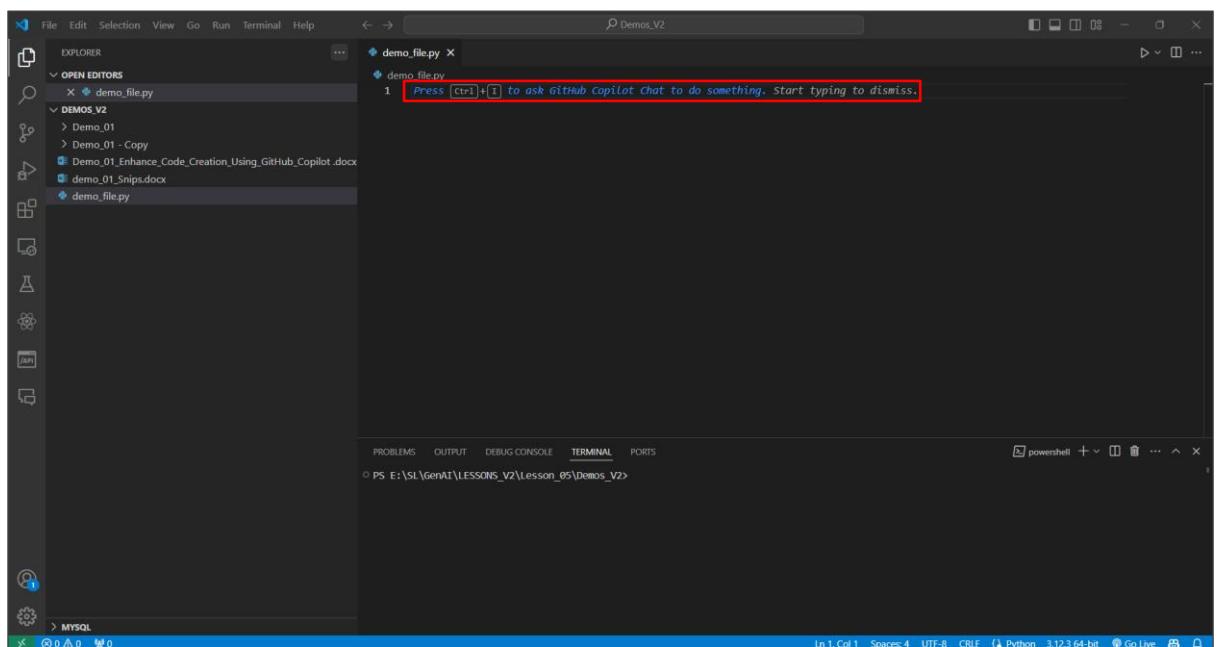
Generative AI tool used in this exercise can produce varied outputs even when presented with similar prompts. Thus, you may get different outputs for the same prompt.

#### Step 1: Integrate GitHub Copilot with Visual Studio

- 1.1 Install the **GitHub Copilot** extension in Visual Studio to use generative AI for code generation



Once the installation and sign-in are completed, a message will appear as shown below:



## Step 2: Create React JS Project with help of Chat GPT

### 2.1 Open the Chat GPT and write the prompt message as

"Step to create the React JS Project with help of new version"

Step to create the React JS Project with the help of the new version



Search



Reason



Create image

Summarize text

Brainstorm

Surprise me

Analyze data

More

The prompt message generated as

### Prerequisites

Ensure the following is installed on your system:

1. **Node.js** (Latest LTS version recommended) – [Download Node.js](#)  
If node JS is not installed please installed with below URL as  
<https://nodejs.org/en>
2. **npm** (Comes with Node.js) or **pnpm/yarn** (optional package managers)

Check your Node.js and npm versions:

Check your Node.js and npm versions:

bash

Copy Edit

```
node -v
npm -v
```

## ✍ Step 1: Set Up a New React Project

Use the latest **create-react-app** alternative: **Vite** or **Create React with Next.js**.

### Option 1: Using **create-react-app** (Still Supported)

#### Option 1: Using **create-react-app** (Still Supported)

bash

Copy Edit

```
npx create-react-app my-app
cd my-app
npm start
```

Note: `npx` ensures the latest package is used without global installation.

### Option 2: Using **Vite** (Faster and More Modern)

```
bash
```

 Copy 

```
npm create vite@latest my-app --template react
cd my-app
npm install
npm run dev
```

---

### Option 3: Using **Next.js** (React Framework – Recommended for Fullstack)

```
bash
```

 Copy 

```
npx create-next-app@latest my-app
cd my-app
npm run dev
```

## 🛠 Step 2: Project Structure

Your project will be structured like this:

```
pgsql
```

 Copy 

```
my-app/
├── node_modules/
├── public/
└── src/
    ├── App.jsx (Main Component)
    ├── index.jsx (Entry Point)
    └── styles.css (Global Styles)
├── .gitignore
└── package.json
└── vite.config.js (for Vite projects)
```

Here in this demo, we use 2<sup>nd</sup> option.

**npm create vite@latest my-to-do-app --template react**

```
using Generative AI>npm create vite@latest my-to-do-app --template react
? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
> React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Angular
  Others
```

Here select, the option as React

Here select as JavaScript

```
using Generative AI>npm create vite@latest my-to-do-app --template react
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
  TypeScript + SWC
> JavaScript
  JavaScript + SWC
  React Router v7 ✓
```

Now move inside a project folder using cd project-name

```
Done. Now run:
cd my-to-do-app
npm install
npm run dev
```

Then installed required dependencies. Run the command as **npm install**

```
| Generative AI\my-to-do-app>npm install  
258 packages, and audited 259 packages in 14s
```

To run the project we need to use the below command as **npm run dev**

```
C:\Users\akash\Desktop\Project with Gen AI\my-to-do-app>npm run dev
```

2.3 Now open this project in VS code.

The project structure is

```
MY-TO-DO-APP
  > node_modules
  > public
    > src
      > assets
      # App.css
      # App.jsx
      # index.css
      # main.jsx
      .gitignore
      eslint.config.js
      index.html
      package-lock.json
      package.json
      README.md
      vite.config.js
```

Here we create the project with configuration as vite. **Vite** (pronounced "veet", meaning "fast" in French) is a **next-generation frontend build tool** that provides an extremely fast development experience for **React, Vue, Svelte, and other modern frameworks**. It was created by **Evan You** (the creator of Vue.js).

So in React with vite configuration extension of file is .jsx. But if you need to .js also not problem. So file can be .jsx or .js both are same.

2.4 Now run the project using the **npm run dev** command.

```
using Generative AI\my-to-do-app>npm run dev
```

```
VITE v6.1.1 ready in 192 ms  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

Open the project in below URL.

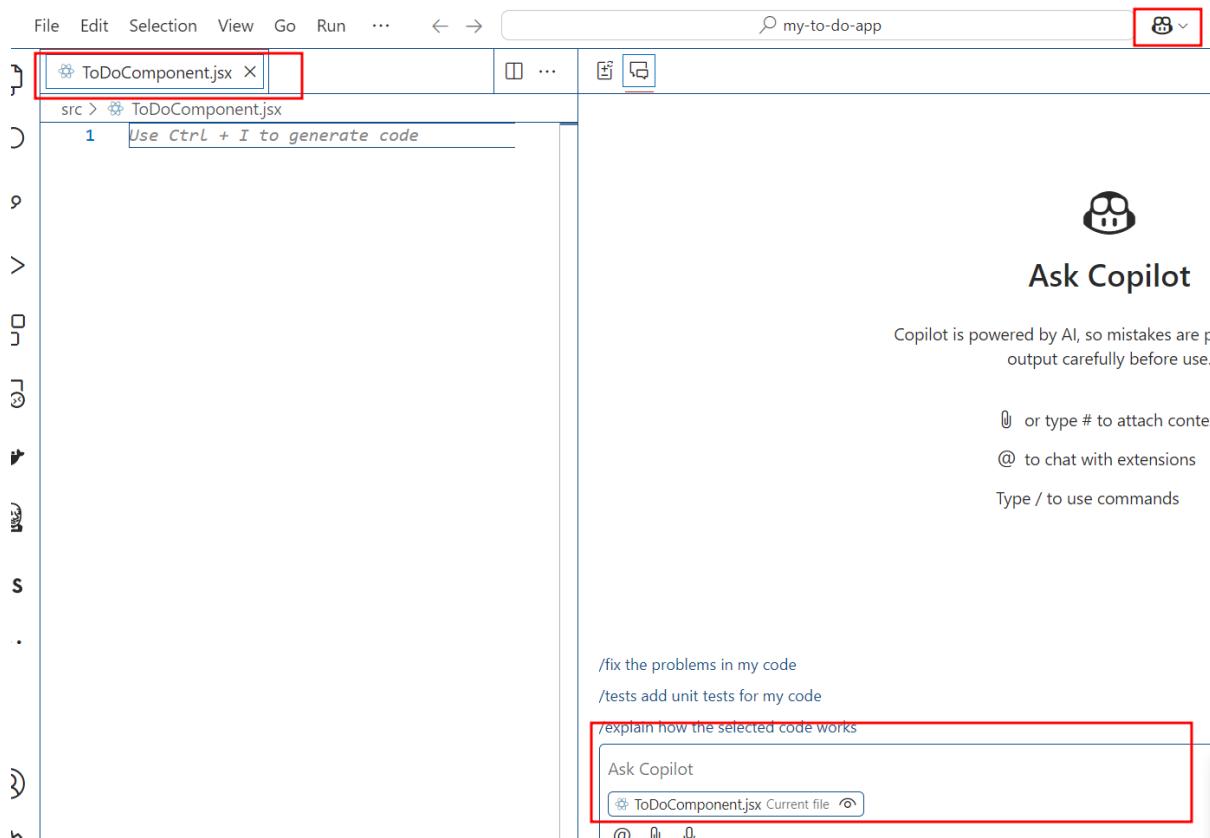
<http://localhost:5173/>



Step 3 : Create User Defined component with help of GitHub Copilot

3.1 Now we will create the new component with help of GitHub Copilot

Create a **ToDoComponent.jsx** file inside a src folder.



Open the GitHub copilot chat option and write the prompt message to generate the React JS component code.

Then Open the Github copilot terminal and write the prompt message as

"Create React JS Component code to generate the To do logic using"

The screenshot shows the GitHub Copilot terminal interface. A red box highlights the input field where the message "Create React JS Component code to generate the To do logic using" is typed. Below the input field, it says "ToDoComponent.jsx Current file". At the bottom, there are command completion suggestions: "@", "U", and "Q".

/fix the problems in my code  
/tests add unit tests for my code  
/explain how the selected code works

Create React JS Component code to generate the To do logic using

ToDoComponent.jsx Current file

@ U Q

After prompt it generate the code.

Create React JS Component code to generate the To do logic using

[ToDoComponent.jsx:1](#)

 GitHub Copilot

> Used 1 reference

Here's a basic implementation of a ToDo component in React. This component allows users to add, delete, and mark tasks as completed.

```
import React, { useState } from 'react';

const ToDoComponent = () => {
  const [tasks, setTasks] = useState([]);
  const [task, setTask] = useState('');

  const addTask = () => {
    if (task.trim()) {
      setTasks([...tasks, { text: task, completed: false }]);
      setTask('');
    }
  };
}
```

How to style this component?



Even GitHub copilot provide the explanation about the generated code.

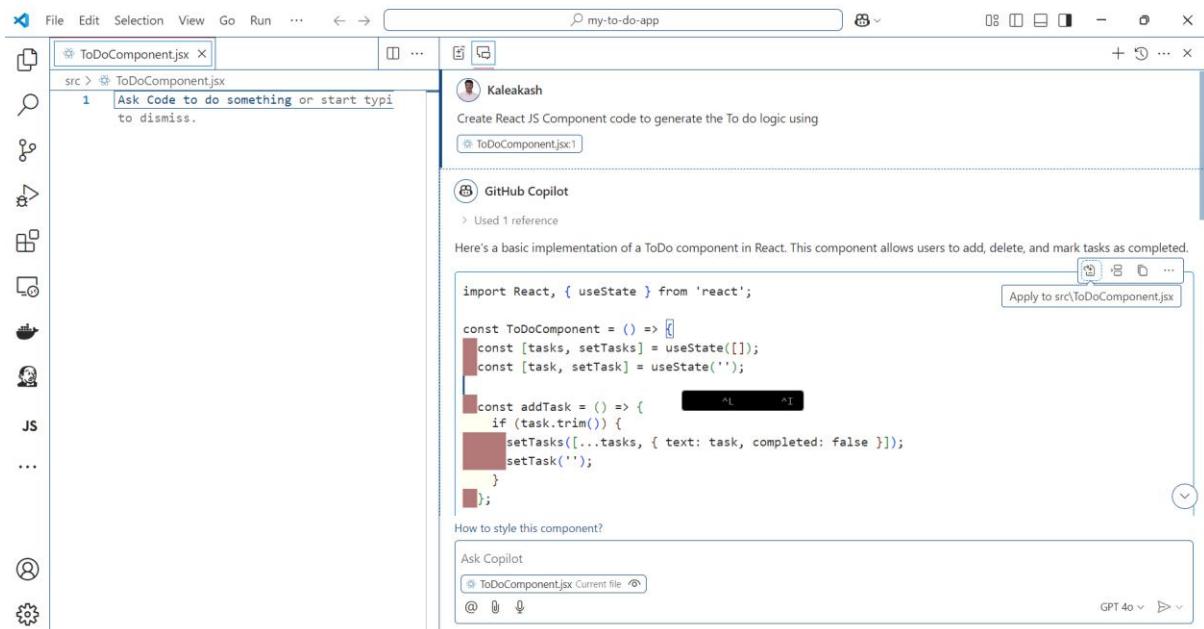
This code defines a `ToDoComponent` that maintains a list of tasks using React's `useState` hook. Users can add new tasks, mark tasks as completed, and delete tasks. The component renders an input field for adding tasks, a button to add the task, and a list of tasks with buttons to complete or delete each task.

Similar code found with 1 license type - [View matches](#)



How to style this component?

If you like this code you can click on apply this code the component

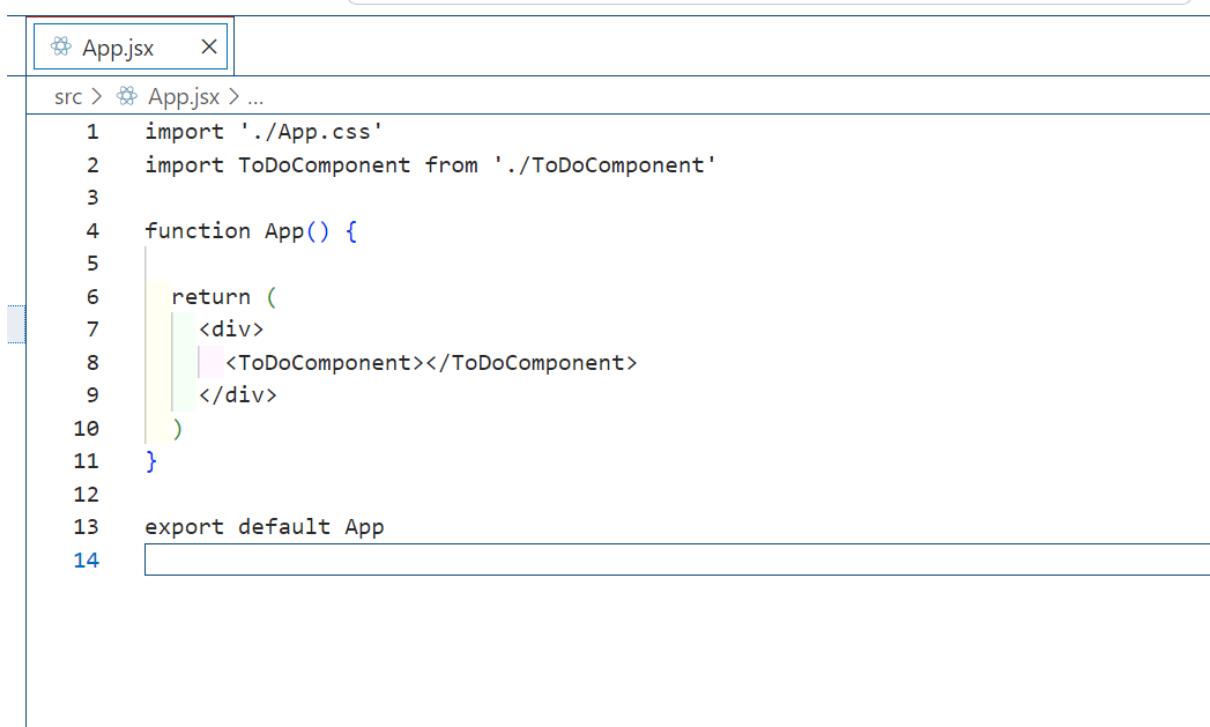


Once you click on Apply to src/ToDoComponent.jsx. This code copy and paste automatically in component tsx file.

The screenshot shows the same code editor window with the "ToDoComponent.jsx" tab selected. The code is identical to the one shown in the previous screenshot, with the "Add Task" button and list item styling removed. The code is now part of the file, ready for use.

```
src > ToDoComponent.jsx ...
1 import React, { useState } from 'react';
2
3 const ToDoComponent = () => {
4   const [tasks, setTasks] = useState([]);
5   const [task, setTask] = useState('');
6
7   const addTask = () => {
8     if (task.trim()) {
9       setTasks([...tasks, { text: task, completed: false }]);
10      setTask('');
11    }
12  };
13
14  const deleteTask = (index) => {
15    const newTasks = tasks.filter((_, i) => i !== index);
16    setTasks(newTasks);
17  };
18
19  const toggleTaskCompletion = (index) => {
20    const newTasks = tasks.map((task, i) =>
21      i === index ? { ...task, completed: !task.completed } : task
22    );
23    setTasks(newTasks);
24  };
25
26  return (
27    <div>
      <h1>To-Do List</h1>
      <input type="text" value={task} onChange={(e) => setTask(e.target.value)} />
      <button onClick={addTask}>Add Task</button>
      <ul style={{ listStyleType: 'none' }}>
        {tasks.map((task, index) => (
          <li key={index}>
            {task.text}
            <span><span style={{ cursor: 'pointer' }}>X</span></span>
            <span style={{ float: 'right' }>
              <span style={{ cursor: 'pointer' }}>Edit</span>
              <span style={{ margin: '0 10px' }}>|</span>
              <span style={{ cursor: 'pointer' }}>Delete</span>
            </span>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

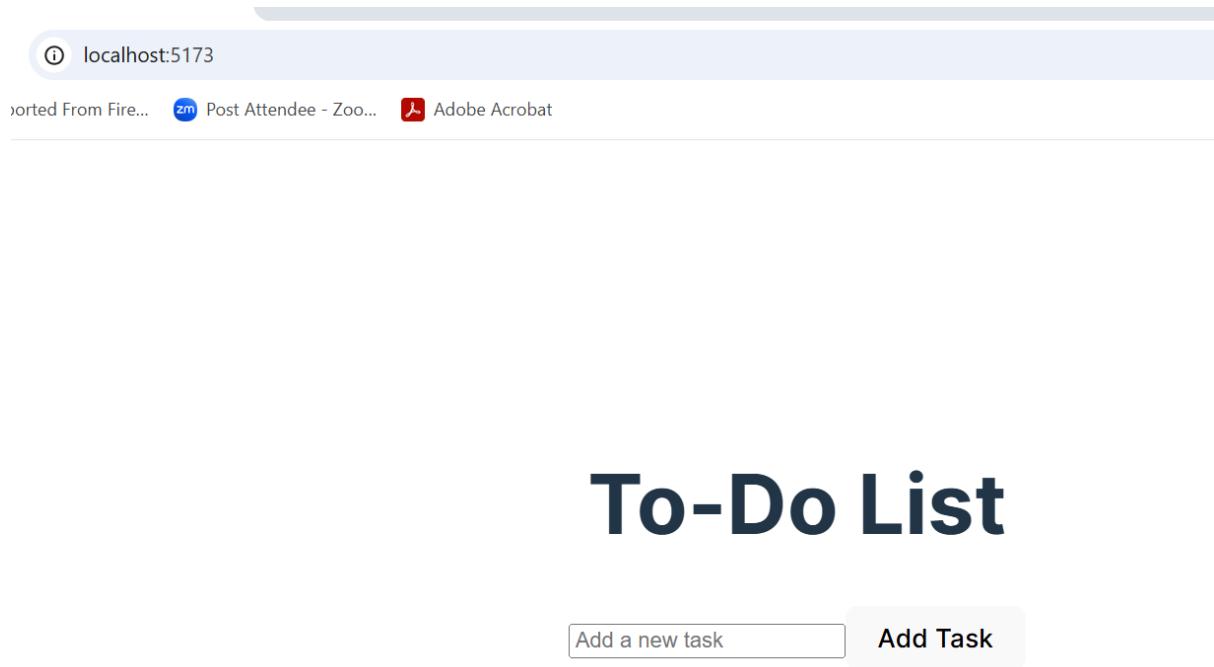
Now Open the App.jsx file and include this component.



The screenshot shows a code editor window with the file 'App.jsx' selected. The code defines a functional component 'App' that imports 'App.css' and 'ToDoComponent'. It returns a single 'ToDoComponent' wrapped in a 'div'. The code ends with an 'export default App' statement.

```
App.jsx
src > App.jsx > ...
1 import './App.css'
2 import ToDoComponent from './ToDoComponent'
3
4 function App() {
5
6     return (
7         <div>
8             <ToDoComponent></ToDoComponent>
9         </div>
10    )
11 }
12
13 export default App
14
```

Now check the updated output on browser



The browser window shows the application running at 'localhost:5173'. The title bar says 'localhost:5173'. The page displays a large 'To-Do List' heading. Below it is a form with a text input field containing 'Add a new task' and a button labeled 'Add Task'.

Add few more task you want to do.

# To-Do List

Add a new task

Add Task

- Send the mail    Complete    Delete
- Attend the meeting    Complete    Delete
- Debug the code    Complete    Delete

Make few task complete.

# To-Do List

Add a new task

Add Task

- ~~Send the mail~~    Undo    Delete
- Attend the meeting    Complete    Delete
- ~~Debug the code~~    Undo    Delete

Remove few task.

# To-Do List

Add a new task

Add Task

- ~~Send the mail~~ Undo Delete
- ~~Debug the code~~ Undo Delete