


Build a React To-Do List App (With Add & Remove Functionality)

 **Concepts:** Functional Components, `useState`, Props, Event Handling, Lists

This activity is a solid **introductory React project**. It teaches the foundational skills needed for dynamic UIs using:

- State management
 - Conditional rendering
 - Event handling
 - JSX structure
-

What You Will Build

A small app where they can:

- Add a new to-do item
 - View a dynamic list of tasks
 - Delete items by clicking a button
-

Step-by-Step Instructions

Step 1: Set up the React Project

1. Open **Visual Studio Code**.
2. Open a terminal inside VS Code.
3. Type the following to create a new React app:

```
npx create-react-app todo-app
```

Wait for it to finish installing. Then navigate into the project folder:

```
cd todo-app
```

Start the development server:

```
npm start
```

It should open a browser window at `http://localhost:3000` showing the default React page.

✅ **Checkpoint:** Make sure the React welcome page loads!

Step 2: Clean Up the Starter Code

1. In the `src/` folder, delete everything inside `App.css`, `App.test.js`, `logo.svg`, `reportWebVitals.js`, and `setupTests.js`.

Step 3: Add State for the To-Do App

1. At the top of the `App.js` file, import the `useState` hook from React.
2. Inside the `App` function:
 - Create a **state variable for the input text** (this is where the user will type the task).
 - Create another **state variable for the list of tasks**, which starts as an empty array.
3. Below the heading, create a form:
 - Add a **text input field**. Connect it to the input state so whatever the user types appears there.
 - Add a button labeled **"Add Task"**.
4. Write a function that runs when the "Add Task" button is clicked:
 - If the input field is not empty, **add the input text to the task list**.
 - Then, **clear the input field**.

👉 **Tip:** Don't forget to use the `set` functions to update your state variables!

Step 4: Display the List of Tasks

1. Below the input and button, add a list using ``.
2. Use a method (like `map`) to loop through your task list.
3. For each task:
 - Create a list item (``) that shows the task text.
 - Add a button next to it labeled **"Delete"**.
4. Write a function that deletes a task when the delete button is clicked:
 - This function should **receive the index of the task to remove**.
 - Then, **filter that task out of the list** and update your state with the new list.

Step 5: Style It (Optional but Encouraged!)



1. Open the `App.css` file.
2. Add styles to make your to-do list look nice:
 - Center the content.
 - Add space around the input field and buttons.
 - Remove bullet points from the list.
 - Maybe use different colors for the buttons.

Final Step: Submit Your Work

1. Take a screenshot of your to-do list running in the browser.
2. Save your work:
 - App.js
 - App.css
3. Upload the screenshot to the class Canvas.

Want a challenge?

After you're done, try adding one of these:

-  Mark tasks as completed
-  Edit a task