## Day 7: Mini Project – Simple Counter App (using props, state, events)

### 🎯 Goal of Day 7

Today, we will build a **Simple Counter App** in React. This project will bring together everything you’ve learned so far: **props, state, and event handling**. By the end, you will: - Understand how to structure a small React app. - Use useState for managing dynamic data. - Pass data via props. - Handle user interactions (button clicks).

### 📝 Problem Statement

We want to create a **Counter App** where: 1. A number is displayed on the screen (the counter). 2. There are buttons to **increase**, **decrease**, and **reset** the counter. 3. The counter value should dynamically update on screen whenever a button is clicked.

### 🧩 Concepts Refresher

#### 1. **State (useState)**

* State stores dynamic values.
* Example:

const [count, setCount] = useState(0);

Here count is the value, setCount is the function to update it.

#### 2. **Props**

* Props are used to pass data from parent to child.
* Example:

<CounterButton label="Increase" onClick={increaseHandler} />

#### 3. **Event Handling**

* Functions triggered by user actions.
* Example:

<button onClick={handleClick}>Click Me</button>

### 🏗️ Step-by-Step Exercise (Build the App)

#### Step 1: Setup State in Parent Component

* Create a parent component CounterApp.
* Use useState to store counter value.

const [count, setCount] = useState(0);

#### Step 2: Create Functions for Events

* increaseCounter → increases count.
* decreaseCounter → decreases count.
* resetCounter → sets count back to 0.

#### Step 3: Pass Functions as Props

* Create child components like CounterButton.
* Pass onClick handlers and labels via props.

#### Step 4: Display Current Count

* Use JSX to show the count value.

### 💻 Example Code

import React, { useState } from "react";  
  
function CounterButton({ label, onClick }) {  
 return <button onClick={onClick}>{label}</button>;  
}  
  
function CounterApp() {  
 const [count, setCount] = useState(0);  
  
 const increaseCounter = () => setCount(count + 1);  
 const decreaseCounter = () => setCount(count - 1);  
 const resetCounter = () => setCount(0);  
  
 return (  
 <div>  
 <h1>Counter: {count}</h1>  
 <CounterButton label="Increase" onClick={increaseCounter} />  
 <CounterButton label="Decrease" onClick={decreaseCounter} />  
 <CounterButton label="Reset" onClick={resetCounter} />  
 </div>  
 );  
}  
  
export default CounterApp;

### ⏱️ 15–20 Minute Exercise

✅ Build the **Counter App** following these steps: 1. Create CounterApp component. 2. Add state for the counter. 3. Add buttons to increase, decrease, and reset. 4. Extract a CounterButton child component that takes label and onClick as props. 5. Display the counter value.

### ⭐ Bonus Challenges (Optional)

* Add a prop to set the initial value of the counter.
* Prevent the counter from going below 0.
* Add a button that increases by 5 instead of 1.

✅ By finishing this, you’ll be ready to move into slightly larger projects and dive deeper into React’s ecosystem.