**ASSIGNMENT – 3**

**VYSHNAVI NAGALLA – 700759215**

**Step 1: Create a React App**

npx create-react-app counter-app

cd counter-app

npm start

**Step 2: Clean Up Default Files**

function App() {

return (

<div>

<h1>Hello React</h1>

</div>

);

}

export default App;

**Step 3: Create a Component**

Create a new file called Counter.jsx in the src folder.

**Jsx:**

function Counter() {

return (

<div>

<h2>Counter</h2>

<button>Increment</button>

</div>

);

}

export default Counter;

**Then use it in App.js:**

import Counter from './Counter';

function App() {

return (

<div>

<h1>Hello React</h1>

<Counter />

</div>

);

}

**Step 4: Add State with useState**

Update your Counter.jsx like this:

import { useState } from "react";

function Counter() {

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

const handleIncrement = () => {

setCount(count + 1);

};

return (

<div>

<h2>Counter</h2>

<span>{count}</span>

<button onClick={handleIncrement}>Increment</button>

</div>

);

}

export default Counter;

**Step 5: Add Decrement & Reset Buttons**

**Update Counter.jsx:**

import { useState } from "react";

function Counter() {

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

const handleIncrement = () => setCount(count + 1);

const handleDecrement = () => setCount(count - 1);

const handleReset = () => setCount(0);

return (

<div>

<h2>Counter</h2>

<span className="badge bg-primary fs-4 me-3">{count}</span>

<button onClick={handleIncrement} className="btn btn-success me-2">+</button>

<button onClick={handleDecrement} className="btn btn-danger me-2">-</button>

<button onClick={handleReset} className="btn btn-warning">Reset</button>

</div>

);

}

export default Counter;

**Step 6: Add Bootstrap for Styling**

In public/index.html, add this inside the <head> tag:

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"

rel="stylesheet"

/>

**Step 7: Render Multiple Counters**

We'll create a parent component called Counters.jsx that manages a list of counters.

**Create Counters.jsx:**

import Counter from "./Counter";

function Counters() {

return (

<div>

<Counter />

<Counter />

<Counter />

</div>

);

}

export default Counters;

**Update App.js to use it:**

import Counters from "./Counters";

function App() {

return (

<div className="container mt-5">

<h1>Hello React</h1>

<Counters />

</div>

);

}

export default App;

**Step 8: Move State to Parent (Counters.jsx)**

Update Counters.jsx to manage an array of counters:

import { useState } from "react";

import Counter from "./Counter";

function Counters() {

const [counters, setCounters] = useState([

{ id: 1, value: 0 },

{ id: 2, value: 0 },

{ id: 3, value: 0 },

]);

const handleIncrement = (counterId) => {

const updated = counters.map((c) =>

c.id === counterId ? { ...c, value: c.value + 1 } : c

);

setCounters(updated);

};

return (

<div>

{counters.map((counter) => (

<Counter

key={counter.id}

value={counter.value}

onIncrement={() => handleIncrement(counter.id)}

/>

))}

</div>

);

}

export default Counters;

**Update Counter.jsx to Receive Props**

function Counter({ value, onIncrement }) {

return (

<div className="mb-3">

<span className="badge bg-primary fs-5 me-3">{value}</span>

<button onClick={onIncrement} className="btn btn-success">+</button>

</div>

);

}

export default Counter;

**Step 9: Add "Reset All" Button**

Update Counters.jsx to include a reset handler and button:

**jsx**

const handleReset = () => {

const resetCounters = counters.map(c => ({ ...c, value: 0 }));

setCounters(resetCounters);

};

**Add the button above the map:**

**jsx**

<button onClick={handleReset} className="btn btn-warning mb-3">

Reset All

</button>

**Step 10: Add Delete Button**

**Add a handleDelete function:**

**jsx**

const handleDelete = (counterId) => {

const updated = counters.filter(c => c.id !== counterId);

setCounters(updated);

};

**Pass it as a prop to each Counter:**

**jsx**

<Counter

key={counter.id}

value={counter.value}

onIncrement={() => handleIncrement(counter.id)}

onDelete={() => handleDelete(counter.id)}

/>

**Update Counter.jsx to Use onDelete**

**jsx**

function Counter({ value, onIncrement, onDelete }) {

return (

<div className="mb-3">

<span className="badge bg-primary fs-5 me-3">{value}</span>

<button onClick={onIncrement} className="btn btn-success me-2">+</button>

<button onClick={onDelete} className="btn btn-danger">Delete</button>

</div>

);

}

**Step 11: Add a Navbar Component**

**Create a new file called Navbar.jsx:**

**jsx**

function Navbar({ totalCounters }) {

return (

<nav className="navbar navbar-light bg-light mb-4">

<div className="container-fluid">

<span className="navbar-brand mb-0 h1">

Counters{" "}

<span className="badge bg-secondary">

{totalCounters}

</span>

</span>

</div>

</nav>

);

}

Update App.js to Use Navbar

jsx

import Navbar from "./Navbar";

import Counters from "./Counters";

function App() {

return (

<div className="container">

<Navbar totalCounters={3} /> {/\* We'll update this in a second \*/}

<Counters />

</div>

);

}

export default App;

**Step 12: Refactor State Management**

**We’ll move the state for the counters back to App.js and pass it down to Counters and Navbar.**

**Update App.js:**

**jsx**

import { useState } from "react";

import Navbar from "./Navbar";

import Counters from "./Counters";

function App() {

const [counters, setCounters] = useState([

{ id: 1, value: 0 },

{ id: 2, value: 0 },

{ id: 3, value: 0 },

]);

const handleIncrement = (counterId) => {

const updated = counters.map((c) =>

c.id === counterId ? { ...c, value: c.value + 1 } : c

);

setCounters(updated);

};

const handleReset = () => {

const resetCounters = counters.map((c) => ({ ...c, value: 0 }));

setCounters(resetCounters);

};

const handleDelete = (counterId) => {

const updated = counters.filter((c) => c.id !== counterId);

setCounters(updated);

};

const totalCounters = counters.filter((c) => c.value > 0).length;

return (

<div className="container">

<Navbar totalCounters={totalCounters} />

<Counters

counters={counters}

onIncrement={handleIncrement}

onDelete={handleDelete}

onReset={handleReset}

/>

</div>

);

}

export default App;

**Update Counters.jsx to Use Props:**

**jsx**

function Counters({ counters, onIncrement, onDelete, onReset }) {

return (

<div>

<button onClick={onReset} className="btn btn-warning mb-3">

Reset All

</button>

{counters.map((counter) => (

<Counter

key={counter.id}

value={counter.value}

onIncrement={() => onIncrement(counter.id)}

onDelete={() => onDelete(counter.id)}

/>

))}

</div>

);

}

export default Counters;

**Step 13: Finalize and Test**

Now, the **Navbar** dynamically updates to show the number of active counters, and the **state is managed centrally** in App.js. You can test it by:

1. Incrementing and deleting counters.
2. Clicking **Reset All** to reset them to zero.