**Exercise 18: Demo about State as a Snapshot**

Objectives and Outcomes

In this example, we'll create a simple counter component that allows you to take a snapshot of the current count value and restore it later.

Exercises

Create a new React component called SnapshotDemo

import React, {useState} from 'react';

const SnapshotDemo = () => {

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

const [snapshot, setSnapshot] = useState(null);

const handleIncrement = () => {

setCount(count + 1);

};

const handleSnapshot = () => {

setSnapshot(count);

};

const handleRestore = () => {

if (snapshot !== null) {

setCount(snapshot);

}

};

return (

<div>

<h1>State as a Snapshot Demo</h1>

<p>Count: {count}</p>

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

<button onClick={handleSnapshot}>Take Snapshot</button>

<button onClick={handleRestore}>Restore Snapshot</button>

</div>

);

};

export default SnapshotDemo;

In this component, we use the useState hook to create two state variables: count and snapshot. The count state represents the current count value, while the snapshot state holds the snapshot of the count value.

We have three event handlers:

* handleIncrement: Increases the count value by 1 when the "Increment" button is clicked.
* handleSnapshot: Takes a snapshot of the current count value and stores it in the snapshot state when the "Take Snapshot" button is clicked.
* handleRestore: Restores the count value from the snapshot state when the "Restore Snapshot" button is clicked, but only if a snapshot is available.

The rendered JSX displays the count value, along with three buttons to increment the count, take a snapshot, and restore the snapshot.



To use the SnapshotDemo component, render it in the root of your application:

import './App.css';

import React from 'react';

import './Component/SnapshotDemo';

function App() {

  return (

    <React.StrictMode>

      <SnapshotDemo />

    </React.StrictMode>

  );

}

Conclusion

Using state as a snapshot allows you to save and restore specific states of your component. In this demo, we save the count value as a snapshot and restore it if a snapshot is available. This concept can be useful when you want to implement features like undo/redo functionality or save and restore user input in forms.