Experiment 2

Explore JSX syntax and its similarities with HTML, render components within the main application, and implement state in a react component using *useState.*

```
import React from 'react';

function Greeting() {
  return <h1>Hello, React!</h1>;
}

function App() {
  return (
    <div>
      <Greeting />
      <p>Welcome to the React world.</p>
    </div>
  );
}

export default App;
```

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

function Counter() {
  // Declare a state variable named "count" with an initial value of 0
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me !!!
      </button>
    </div>
  );
}
```

```
function App() {
  return (
    <div>
      <h1>Counter Example</h1>
      <Counter />
    </div>
  );
}

export default App;
```

Experiment 3

Pass data between components using props, display dynamic data using state and props.

```
import React, { useState } from 'react';
// Child Component
function UserProfile(props) {
  return (
            <div>
               <h2>User Profile</h2>
              <p>Name: {props.name}</p>
               <p>Age: {props.age}</p>
              </div>
         );
}
// Parent Component
function App() {
         // State to manage dynamic data
  const [user, setUser] = useState ({name: 'Alice', age: 25});

          // Function to update user data
          const updateUser = () => {
           setUser({ name: 'Bob', age: 30 });
         };
  return (
              <div>
```

```
                <UserProfile name={user.name} age={user.age} />
<button onClick={updateUser}> Update User
</button>
              </div>
  );
}
export default App;
```

```
import React from 'react';
// Child Component
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
// Parent Component
function App () {
        return (
            <div>
             <Greeting name="Alice" />
                 <Greeting name="Bob" />
           </div>
         );
}
export default App;
```

Experiment 4

Implement the useState and useEffect hooks, understand the difference between class component lifecycle methods and hooks, and manage component state and side effects using hooks.

```
import React, { useState, useEffect } from 'react';
import App from '../../../Experiment2/src/App';

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

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  }, [count]); // Only re-run the effect if count changes

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
export default App;
```

```
import React, { useState } from 'react';
import App from '../../../Experiment2/src/App';

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

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
export default App;
```

Experiment 5

Create a form with controlled inputs, implement form validation using state, handle form submission and update components state accordingly, setup react router for client-side routing.

```
import { useState } from "react";
function SimpleForm() {
  const [formData, setFormData] = useState({ name: "", email: "" });
  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Submitted Data:", formData);
  };
  return (
    <form onSubmit={handleSubmit}>
      <label> Name:
       <input type="text" name="name" value={formData.name}
onChange={handleChange} />
      </label>
      <label> Email:
     <input type="email" name="email" value={formData.email}
onChange={handleChange} />
      </label>
      <button type="submit">Submit</button>
    </form>
  );
}
export default SimpleForm;
```

```
import { useRef } from "react";

function UncontrolledForm() {
  const nameRef = useRef();
  const emailRef = useRef();

  const handleSubmit = (e) => {
    e.preventDefault();
```

```
    console.log("Name:", nameRef.current.value);
    console.log("Email:", emailRef.current.value);
  };

  return (
    <form onSubmit={handleSubmit}>
      <label> Name <input type="text" ref={nameRef} /> </label>
      <label> Email: <input type="email" ref={emailRef} /> </label>
      <button type="submit">Submit</button>
    </form>
  );
}
export default UncontrolledForm;
```

```
import './App.css';
import React from 'react';
function App(){
  return (
    <form className='App'>
      <label> First name: </label><br/>
      <input type="text" value="vilas" /> <br/>
    </form>
  );
}
export default App;
```

Experiment 6

Implement nested routes and route parameters, create navigation links for seamless user experience.

```
import React from "react";
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
import Home from './Pages/Home.jsx';
import Project from './Pages/Project.jsx';
import Contact from './Pages/Contact.jsx';
```

```
const App = () => (
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/Project" element={<Project />} />
      <Route path="/Contact" element={<Contact />} />
    </Routes>
  </BrowserRouter>
);

export default App;
```

Experiment 7

Demonstration of conditional rendering in react and react list is used. Implement inline styles in react components, explore the usage of CSS modules for scoped styling, and compare & contrast the pros & cons of inline styles & CSS modules.

```
function App() {
  const items = ['Apple', 'Banana', 'Cherry'];

  return (
    <div>
      <h1>My Fruit List</h1>
      <ul>
        {items.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

Experiment 8

Implementation of Prop Drilling

```
import './App.css';
import React from 'react';
import ChildA from './ChildA';

function App() {
  const name = "Vilas";
  return (
    <>
      <ChildA name={name} />
    </>
  );
}

export default App;
```

```
import React from 'react';
import ChildB from './ChildB';

function ChildA({ name }) {
  return (
    <ChildB name={name} />
  );
}

export default ChildA;
```

```
import React from 'react';
import ChildC from './ChildC';

function ChildB({ name }) {
  return (
    <ChildC name={name} />
  );
}

export default ChildB;
import React from 'react';
```

```
function ChildC({ name }) {
  return (
    <h1>Hello, {name}!</h1>
  );
}

export default ChildC;
```

Experiment 9

Implement the context API for global state management to share data between components using the context explore use cases and best practices for context API.

```
import './App.css';
import React from 'react';
import ChildA from './ChildA';
import { createContext } from 'react';

const data = createContext();
const data1 = createContext();

function App() {
  const name = "Yoshita";
  const gender = "Female";

  return (
    <>
      <data.Provider value={name}>
        <data1.Provider value={gender}>
          <ChildA />
        </data1.Provider>
      </data.Provider>
    </>
  );
}
```

```
export default App;
export { data, data1 };
```

```
import React from 'react';
import ChildC from './ChildC';

function ChildA() {
  return (
    <ChildC />
  );
}

export default ChildA;
```

```
import React from 'react';
import { data, data1 } from './App';

function ChildC() {
  return (
    <data.Consumer>
      {(name) => {
        return (
          <data1.Consumer>
            {(gender) => {
              return (
                <h1>My name is {name} and my gender is {gender}</h1>
              );
            }}
          </data1.Consumer>
        );
      }}
    </data.Consumer>
  );
}

export default ChildC;
```

Experiment 10

Create a higher order component for code reuse, implement render props patterns in react applications, compare a contract and HOCs, & Render props.

```
Higher.jsx
import React, { useState } from 'react'

function UpdatedComponent(OriginalComponent){
    function NewComponent(){
        const [Money,setMoney]=useState(10)
        const handleincrese =() => {
        setMoney(Money*2);
        };
        return <OriginalComponent handleincrese={handleincrese}
Money={Money} />
    }
    return NewComponent;
}

export default UpdatedComponent;
```

```
Person1.jsx
import UpdatedComponent from './higher';

function Person1({Money,handleincrese}) {
  // // problem is Money and handleincrese undefined here, so it solved
using props
  return (
    <div>
      <h2>Jimmy is offering ${Money}</h2>
      <button onClick={handleincrese}>Increase Money</button>
    </div>
  );
}

export default UpdatedComponent(Person1);
```

```
Person2.jsx
```

```
import UpdatedComponent from './higher';

function Person2({Money,handleincrese}) {
   // problem is Money and handleincrese undefined here, so it solved
using props
  return (
    <div>
      <h2>John is offering ${Money}</h2>
      <button onClick={handleincrese}>Increase Money</button>
    </div>
  )
}
export default UpdatedComponent(Person2);
```