**Name - Gaurang A Raorane Roll No - 49**

**Class - D15A Batch - C**

**EXPERIMENT NO - 8**

**Aim:-** Tostudy the Advanced React.

**Theory:-**

* **State Management:**

Advanced React often involves managing state with more complex scenarios. This can be achieved through various methods, including Context API, Redux, or Mobx.

Understanding the pros and cons of different state management solutions is crucial.

* **React Router:**

React Router is a popular library for implementing client-side routing in React applications.

Study the concepts of routes, nested routes, route parameters, and route guards for authentication and authorization.

* **Forms and Form Libraries:**

Handling complex forms with controlled components and form validation.

Exploring form libraries like Formik and Yup to simplify form management.

* **Hooks in Depth:**

Dive deeper into React hooks like useMemo, useCallback, and useRef to optimize performance and understand their use cases.

* **Context API:**

Utilizing the Context API for managing global application state and reducing prop drilling.

* **Redux (Optional):**

If not covered in state management, explore Redux for more complex state management scenarios and middleware.

* **Higher-Order Components (HOCs):**

Understand how HOCs can be used to enhance component behavior and reuse code.

* **Error Handling:**

Implement error boundaries to gracefully handle errors in your application.

* **Performance Optimization:**

Techniques like memoization, code splitting, lazy loading, and using React's built-in tools for optimizing performance.

* **Server-Side Rendering (SSR) and Next.js (Optional)**:

Learn about SSR with React for improved SEO and performance. Explore frameworks like Next.js that simplify SSR in React.

* **Testing:**

Explore advanced testing techniques using tools like Jest and React Testing Library to ensure application reliability.

**Input:-**

import React, { useState, useRef } from "react";

import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

function App() {

// State to manage the shopping cart

const [cart, setCart] = useState([]);

// Ref for input field

const quantityInputRef = useRef();

// Function to add an item to the cart

const addItemToCart = (item) => {

setCart([...cart, item]);

quantityInputRef.current.value = ""; // Clear the quantity input

};

return (

<Router>

<div className="App">

<h1>Online Shopping</h1>

<nav>

<ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

<Link to="/products">Products</Link>

</li>

<li>

<Link to="/cart">Cart ({cart.length} items)</Link>

</li>

</ul>

</nav>

<Routes>

<Route path="/" element={<Home />} />

<Route

path="/products"

element={

<ProductList

addItemToCart={addItemToCart}

quantityInputRef={quantityInputRef}

/>

}

/>

<Route path="/cart" element={<Cart cart={cart} />} />

</Routes>

</div>

</Router>

);

}

// Home component

function Home() {

return <div>Welcome to our online store!</div>;

}

// ProductList component

function ProductList({ addItemToCart, quantityInputRef }) {

const products = [

{ id: 1, name: "Product 1", price: 10 },

{ id: 2, name: "Product 2", price: 20 },

{ id: 3, name: "Product 3", price: 30 },

];

return (

<div>

<h2>Products</h2>

<ul>

{products.map((product) => (

<li key={product.id}>

{product.name} - ${product.price}

<input

type="number"

ref={quantityInputRef}

defaultValue="1"

min="1"

max="10"

/>

<button onClick={() => addItemToCart(product)}>Add to Cart</button>

</li>

))}

</ul>

</div>

);

}

// Cart component

function Cart({ cart }) {

return (

<div>

<h2>Cart</h2>

<ul>

{cart.map((item, index) => (

<li key={index}>{item.name}</li>

))}

</ul>

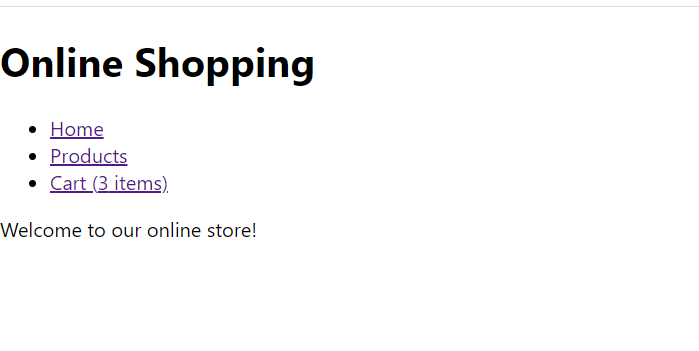
</div>

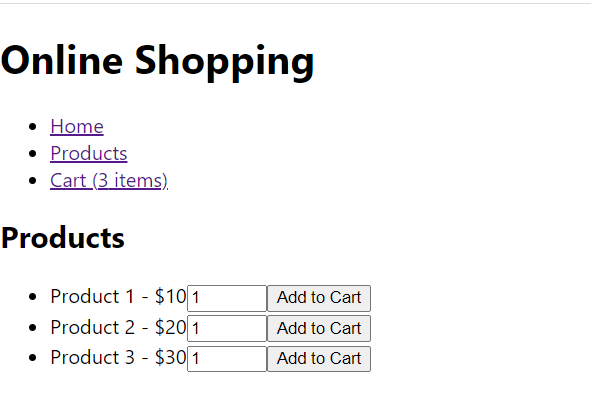
);

}

export default App;

**Output:-**

****

****