Here are the **key React & JS concepts** you learned from this project:

**✅ React State Management**

* Using useState to manage **cart items**, **quantities**, and **flags**.
* **Lazy initializer** in useState to read from sessionStorage only on **first render**.

**✅ Side Effects with useEffect**

* Persisting state to **sessionStorage** whenever cartItems changes.
* Hydrating state from **sessionStorage** on component mount.

**✅ Session Storage vs Local Storage**

* **SessionStorage** → persists until tab is closed.
* **LocalStorage** → persists until manually cleared.
* Used sessionStorage to **retain cart state** even on refresh.

**✅ Routing & Navigation**

* Used **React Router useNavigate** to go between pages.
* Passed data via navigate('/cart', { state }) and accessed with useLocation.

**✅ Conditional Rendering**

* Show “Cart is empty” message when no items exist.
* Show **Go back to Home** button only when cart is cleared.

**✅ Immutable State Updates**

* Used **spread operator** (...prev) to update arrays instead of mutating state.
* Managed quantityState immutably.

**✅ Array Methods**

* map() → Render products and cart items.
* filter() → Remove specific item from cart.
* some() → Check if item already exists in cart.

**✅ Performance Optimization**

* Avoid unnecessary storage reads by using **lazy initialization** in useState.
* Kept **products array logic in memory** but learned why resetting flags after navigation is tricky → solved via session storage.

**✅ React Render Flow**

* Learned that:
  + **Render → Commit → Paint → useEffect runs after commit**.
  + State updates are **batched**.

**React + JS Interview Cheat Sheet**

*(Key concepts + example snippets + likely interview questions)*

**1. State Management**

**Key Concept:**

* useState manages component state.
* Lazy initialization with a function is useful for expensive operations or reading from storage.

**Example:**

javascript

CopyEdit

const [cartItems, setCartItems] = useState(() => {

const storedCart = JSON.parse(sessionStorage.getItem('cart')) || [];

return storedCart;

});

**Questions:**

* When should you use a function in useState instead of a value?
* Why should state updates be immutable in React?

**2. Side Effects & useEffect**

**Key Concept:**

* Runs after render and commit phase.
* Used for syncing state with storage or fetching data.

**Example:**

useEffect(() => {

sessionStorage.setItem('cart', JSON.stringify(cartItems));

}, [cartItems]);

**Questions:**

* When does useEffect run?
* Difference between **render**, **commit**, and **paint** phases in React?

**3. Session Storage vs Local Storage**

**Key Concept:**

* sessionStorage: persists until tab is closed.
* localStorage: persists until manually cleared.

**Example:**

sessionStorage.setItem('cart', JSON.stringify(cartItems));

**Questions:**

* What is the difference between session and local storage?
* Why did you choose session storage for this project?

**4. Routing with React Router**

**Key Concept:**

* Navigate and pass state with useNavigate.
* Retrieve state with useLocation.

**Example:**

navigate('/cart', { state: { products } });

const location = useLocation();

const { products } = location.state;

**Questions:**

* What happens to location.state when you refresh the page?
* How do you persist data across page refreshes?

**5. Array Operations**

**Key Concept:**

* map() → render list.
* filter() → remove items.
* some() → check for duplicates.

**Example:**

if (cartItems.some(item => item.name === products[idx].name)) {

alert('Item already added');

}

**Questions:**

* Difference between map and forEach in React rendering?
* How do you check for duplicate objects in an array?