

Shopping Cart with Context and Reducers

You are developing an e-commerce application, and you want to manage the shopping cart state efficiently.

1. Create a CartContext using React Context to manage the shopping cart state.
2. Implement a CartProvider component to wrap your application and provide the cart context.
3. Define reducer functions for adding items, removing items, and updating quantities in the cart.
4. Use the cart context and reducer functions in your components to interact with the shopping cart.

Solution –

App.js

```
import React from 'react';
import Product from './Product';
import Cart from './Cart';
import './App.css';
```

```
function App() {
  return (
    <div className="App">
      <h1>Shopping Cart</h1>
      <Product />
      <Cart />
    </div>
  );
}
```

```
export default App;
```

```
JS CartContext.js U JS index.js U JS Product.js U JS Cart.js U JS App.js U X # App.css U
shopping-cart-app > src > JS App.js > App
1 import React from 'react';
2 import Product from './Product';
3 import Cart from './Cart';
4 import './App.css';
5
6 function App() {
7   return (
8     <div className="App">
9       <h1>Shopping Cart</h1>
10      <Product />
11      <Cart />
12    </div>
13  );
14 }
15
16 export default App;
```

CartContext.js

```
import React, { createContext, useReducer, useContext } from 'react';
```

```
// Create the Cart Context
```

```
const CartContext = createContext();
```

```
// Define the initial state
```

```
const initialState = {
  cartItems: [],
};
```

```
// Define the reducer function
```

```
const cartReducer = (state, action) => {
  switch (action.type) {
    case 'ADD_ITEM':
      const existingItem = state.cartItems.find(
        (item) => item.id === action.payload.id
      );
      if (existingItem) {
```

```

return {
  ...state,
  cartItems: state.cartItems.map((item) =>
    item.id === action.payload.id
      ? { ...item, quantity: item.quantity + 1 }
      : item
  ),
};
} else {
  return {
    ...state,
    cartItems: [...state.cartItems, { ...action.payload, quantity: 1 }],
  };
}

```

```

case 'REMOVE_ITEM':

```

```

  return {
    ...state,
    cartItems: state.cartItems.filter((item) => item.id !== action.payload),
  };

```

```

case 'UPDATE_QUANTITY':

```

```

  return {
    ...state,
    cartItems: state.cartItems.map((item) =>
      item.id === action.payload.id
        ? { ...item, quantity: action.payload.quantity }

```

```

        : item
      ),
    };

    default:
      return state;
    }
  };

// Create the Cart Provider component
export const CartProvider = ({ children }) => {
  const [state, dispatch] = useReducer(cartReducer, initialState);

  return (
    <CartContext.Provider value={{ state, dispatch }}>
      {children}
    </CartContext.Provider>
  );
};

// Custom hook to use the Cart Context
export const useCart = () => {
  return useContext(CartContext);
};

```

JS CartContext.js U X JS index.js U JS Product.js U JS Cart.js U JS App.js U # App.css U

```
shopping-cart-app > src > JS CartContext.js > [9] cartReducer
1  import React, { createContext, useReducer, useContext } from 'react';
2
3  // Create the Cart Context
4  const CartContext = createContext();
5
6  // Define the initial state
7  const initialState = {
8    cartItems: [],
9  };
10
11 // Define the reducer function
12 const cartReducer = (state, action) => {
13   switch (action.type) {
14     case 'ADD_ITEM':
15       const existingItem = state.cartItems.find(
16         (item) => item.id === action.payload.id
17       );
18       if (existingItem) {
19         return [
20           ...state,
21           cartItems: state.cartItems.map((item) =>
22             item.id === action.payload.id
23               ? { ...item, quantity: item.quantity + 1 }
24               : item
25           ),
26         ];
27       } else {
28         return {
29           ...state,
30           cartItems: [...state.cartItems, { ...action.payload, quantity: 1 }],
31         };
32       }
33
34     case 'REMOVE_ITEM':
35       return {
36         ...state,
37         cartItems: state.cartItems.filter((item) => item.id !== action.payload),
38       };
39
40     case 'UPDATE_QUANTITY':
41       return {
42         ...state,
43         cartItems: state.cartItems.map((item) =>
44           item.id === action.payload.id
45             ? { ...item, quantity: action.payload.quantity }
46             : item
47         ),
48       };
49
50     default:
51       return state;
52   }
53 };
54
55 // Create the Cart Provider component
56 export const CartProvider = ({ children }) => {
57   const [state, dispatch] = useReducer(cartReducer, initialState);
58
59   return (
60     <CartContext.Provider value={{ state, dispatch }}>
61       {children}
62     </CartContext.Provider>
63   );
64 };
65
66 // Custom hook to use the Cart Context
67 export const useCart = () => {
68   return useContext(CartContext);
69 };
```

Product.js

```
import React from 'react';
import { useCart } from './CartContext';

const products = [
  { id: 1, name: 'Product 1', price: 10 },
  { id: 2, name: 'Product 2', price: 20 },
  { id: 3, name: 'Product 3', price: 30 },
];

const Product = () => {
  const { dispatch } = useCart();

  const addToCart = (product) => {
    dispatch({ type: 'ADD_ITEM', payload: product });
  };

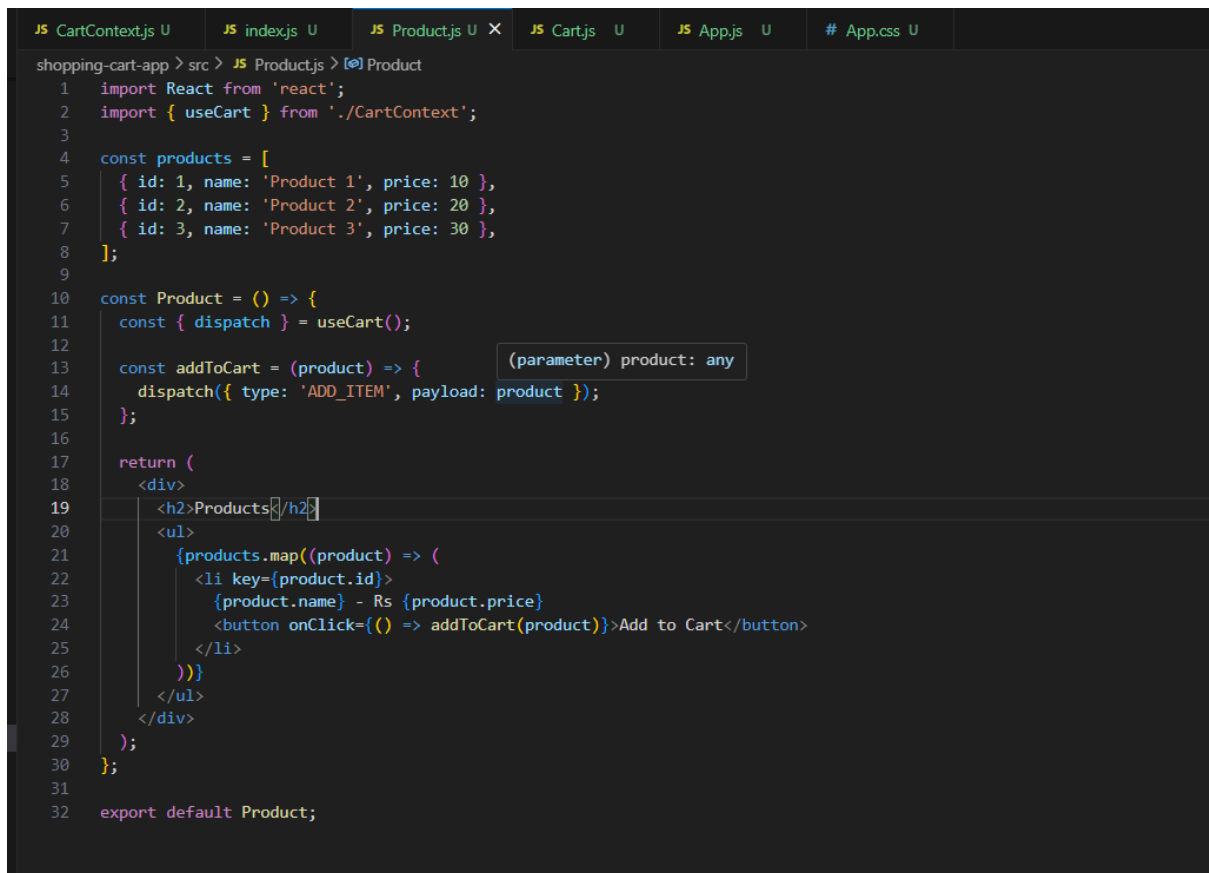
  return (
    <div>
      <h2>Products</h2>
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            {product.name} - Rs {product.price}
            <button onClick={() => addToCart(product)}>Add to Cart</button>
          </li>
        ))}
      </ul>
    </div>
  );
}
```


</div>

);

};

export default Product;



```
JS CartContext.js U JS index.js U JS Product.js U X JS Cart.js U JS App.js U # App.css U
shopping-cart-app > src > JS Product.js > [e] Product
1 import React from 'react';
2 import { useCart } from './CartContext';
3
4 const products = [
5   { id: 1, name: 'Product 1', price: 10 },
6   { id: 2, name: 'Product 2', price: 20 },
7   { id: 3, name: 'Product 3', price: 30 },
8 ];
9
10 const Product = () => {
11   const { dispatch } = useCart();
12
13   const addToCart = (product) => { (parameter) product: any
14     dispatch({ type: 'ADD_ITEM', payload: product });
15   };
16
17   return (
18     <div>
19       <h2>Products</h2>
20       <ul>
21         {products.map((product) => (
22           <li key={product.id}>
23             {product.name} - Rs {product.price}
24             <button onClick={() => addToCart(product)}>Add to Cart</button>
25           </li>
26         ))}
27       </ul>
28     </div>
29   );
30 };
31
32 export default Product;
```

Cart.js

import React, { useState } from 'react';

import { useCart } from './CartContext';

const Cart = () => {

const { state, dispatch } = useCart();

const { cartItems } = state;

```

const removeItem = (id) => {
  dispatch({ type: 'REMOVE_ITEM', payload: id });
};

const updateQuantity = (id, quantity) => {
  if (quantity <= 0) return;
  dispatch({ type: 'UPDATE_QUANTITY', payload: { id, quantity } });
};

return (
  <div>
    <h2>Cart</h2>
    {cartItems.length === 0 ? (
      <p>Your cart is empty.</p>
    ) : (
      <ul>
        {cartItems.map((item) => (
          <li key={item.id}>
            {item.name} – Rs {item.price} x {item.quantity}
            <button onClick={() => removeItem(item.id)}>Remove</button>
            <input
              type="number"
              value={item.quantity}
              onChange={(e) =>
                updateQuantity(item.id, parseInt(e.target.value))
              }
            />
          </li>
        ))}
      </ul>
    )}
  </div>
)

```



```

    </ul>

  })
</div>

);

};

export default Cart;

```

```

JS CartContext.js U  JS index.js U  JS Product.js U  JS Cart.js U  JS App.js U  # App.css U
shopping-cart-app > src > JS Cart.js > [e] Cart
1  import React, { useState } from 'react';
2  import { useCart } from './CartContext';
3
4  const Cart = () => {
5    const { state, dispatch } = useCart();
6    const { cartItems } = state;
7
8    const removeItem = (id) => {
9      dispatch({ type: 'REMOVE_ITEM', payload: id });
10   };
11   const updateQuantity = (id, quantity) => {
12     if (quantity <= 0) return;
13     dispatch({ type: 'UPDATE_QUANTITY', payload: { id, quantity } });
14   };
15   return (
16     <div>
17       <h2>Cart</h2>
18       {cartItems.length === 0 ? (
19         <p>Your cart is empty.</p>
20       ) : (
21         <ul>
22           {cartItems.map((item) => (
23             <li key={item.id}>
24               {item.name} - ${item.price} x {item.quantity}
25               <button onClick={() => removeItem(item.id)}>Remove</button>
26               <input
27                 type="number"
28                 value={item.quantity}
29                 onChange={(e) =>
30                   updateQuantity(item.id, parseInt(e.target.value))
31                 }
32               />
33             </li>
34           ))}
35         </ul>
36       )}
37     </div>
38   );
39 };
40

```

App.css

```

/* App.css */

body {
  font-family: Arial, sans-serif;
  background-color: #f8f9fa;
  margin: 0;
  padding: 0;
}

```

```
    text-align: center;
}
h1, h2 {
    color: #333;
}
.App {
    max-width: 800px;
    margin: 20px auto;
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
/* Product List */
ul {
    list-style-type: none;
    padding: 0;
}
li {
    background: #ffffff;
    margin: 10px 0;
    padding: 10px;
    border-radius: 5px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    border: 1px solid #ddd;
```

```
}  
button {  
  background-color: #28a745;  
  color: white;  
  border: none;  
  padding: 8px 12px;  
  border-radius: 5px;  
  cursor: pointer;  
  transition: background 0.3s;  
}  
button:hover {  
  background-color: #218838;  
}  
/* Cart */  
input[type="number"] {  
  width: 50px;  
  text-align: center;  
  margin: 0 10px;  
}  
.cart-empty {  
  font-size: 18px;  
  color: #777;  
}  
.cart-item {  
  display: flex;  
  align-items: center;  
  justify-content: space-between;
```

```
}
```

```
.cart-item button {
```

```
background-color: #dc3545;
```

```
}
```

```
.cart-item button:hover {
```

```
background-color: #c82333;
```

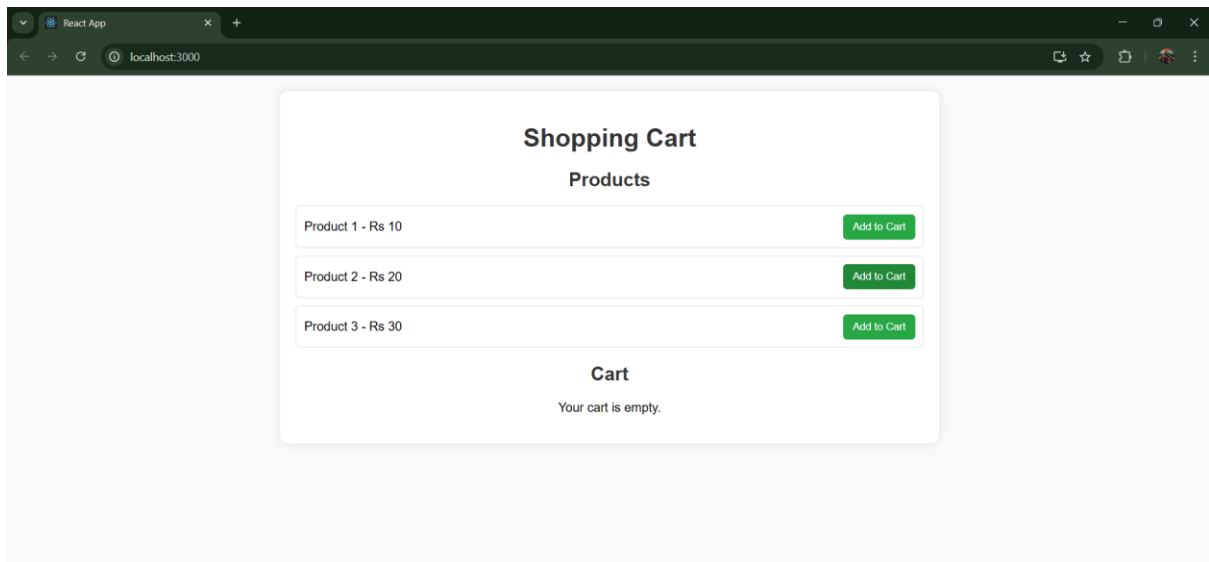
```
}
```

```
JS CartContext.js U X JS index.js U JS Product.js U JS Cart.js U JS App.js U # App.css U X
shopping-cart-app > src > # App.css > ul
1  /* App.css */
2
3  body {
4    font-family: Arial, sans-serif;
5    background-color: #f8f9fa;
6    margin: 0;
7    padding: 0;
8    text-align: center;
9  }
10
11  h1, h2 {
12    color: #333;
13  }
14
15  .App {
16    max-width: 800px;
17    margin: 20px auto;
18    background: white;
19    padding: 20px;
20    border-radius: 10px;
21    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
22  }
23
24  /* Product List */
25  ul {
26    list-style-type: none;
27    padding: 0;
28  }
29
30  li {
31    background: #ffffff;
32    margin: 10px 0;
33    padding: 10px;
34    border-radius: 5px;
35    display: flex;
36    justify-content: space-between;
37    align-items: center;
38    border: 1px solid #ddd;
39  }
```

```
JS CartContext.js U JS index.js U JS Product.js U JS Cart.js U JS App.js U # App.css U X
shopping-cart-app > src > # App.css > ul
41 button {
42   background-color: #28a745;
43   color: white;
44   border: none;
45   padding: 8px 12px;
46   border-radius: 5px;
47   cursor: pointer;
48   transition: background 0.3s;
49 }
50
51 button:hover {
52   background-color: #218838;
53 }
54
55 /* Cart */
56 input[type="number"] {
57   width: 50px;
58   text-align: center;
59   margin: 0 10px;
60 }
61
62 .cart-empty {
63   font-size: 18px;
64   color: #777;
65 }
66
67 .cart-item {
68   display: flex;
69   align-items: center;
70   justify-content: space-between;
71 }
72
73 .cart-item button {
74   background-color: #dc3545;
75 }
76
77 .cart-item button:hover {
78   background-color: #c82333;
79 }
80
```

OUTPUT –

Before Adding the Product to Cart



After Adding the Product to the Cart

