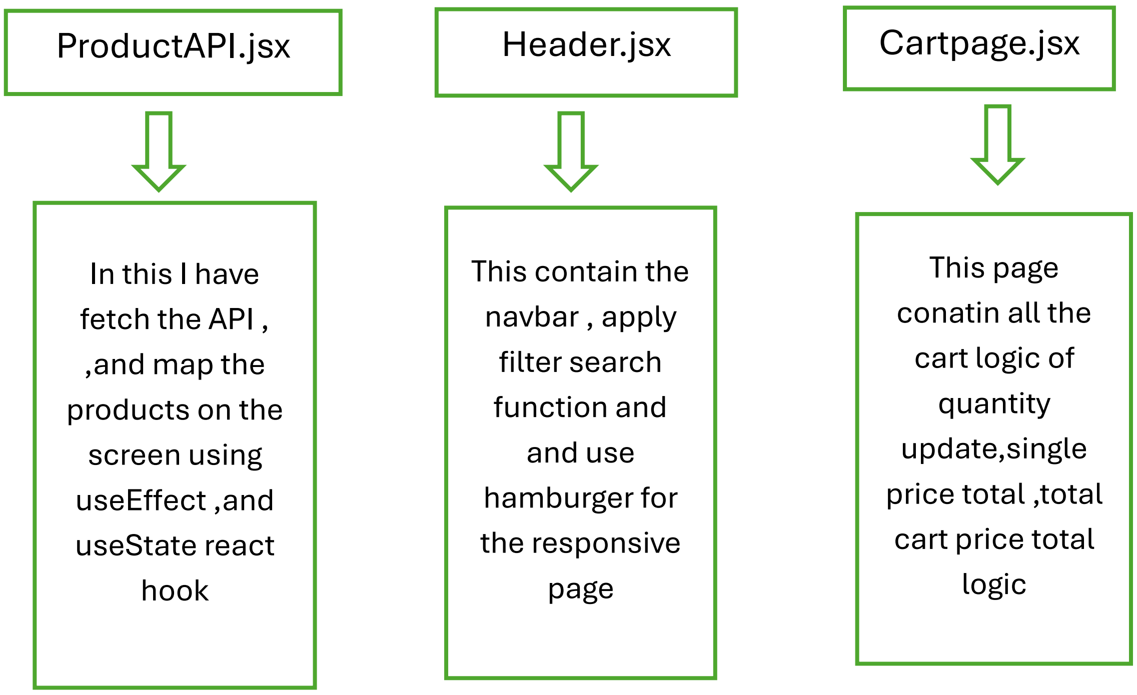
E-Commerce API Integration with cart logic

RF internship Task no 2

In this project I learn to fetch fake API for any e-commerce website

Main components I created as



ProductAPI.jsx

In this I have fetch the API , ,and map the products on the screen using useEffect ,and useState react hook

Header.jsx

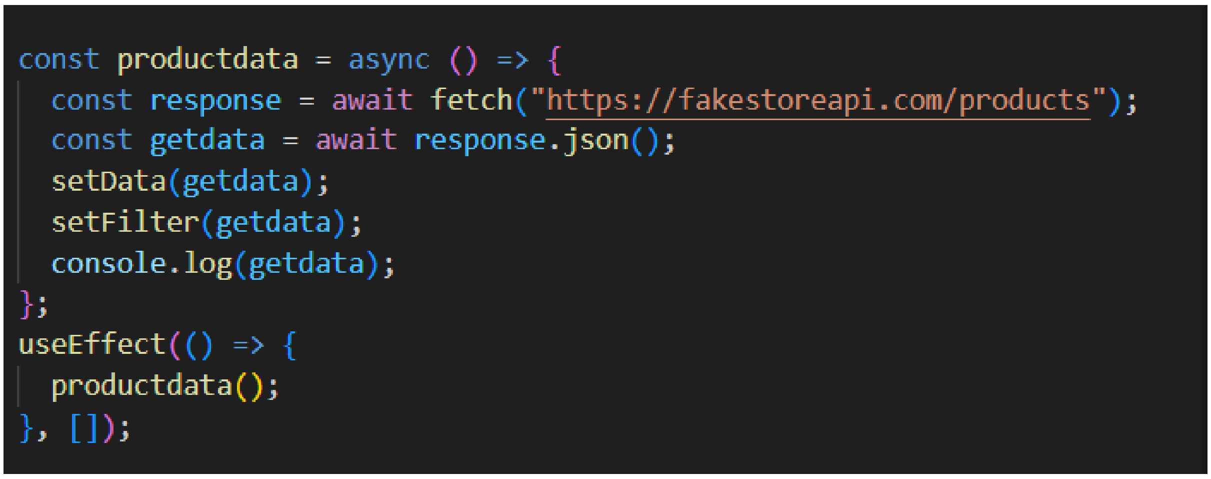
This contain the navbar , apply filter search function and and use hamburger for the responsive page

Cartpage.jsx

This page conatin all the cart logic of quantity update,single price total ,total cart price total logic

ProductAPI.jsx

Fetch function



```
const productdata = async () => {
  const response = await fetch("https://fakestoreapi.com/products");
  const getdata = await response.json();
  setData(getdata);
  setFilter(getdata);
  console.log(getdata);
};
useEffect(() => {
  productdata();
}, []);
```

async = creates a Promise

await = waits for a Promise

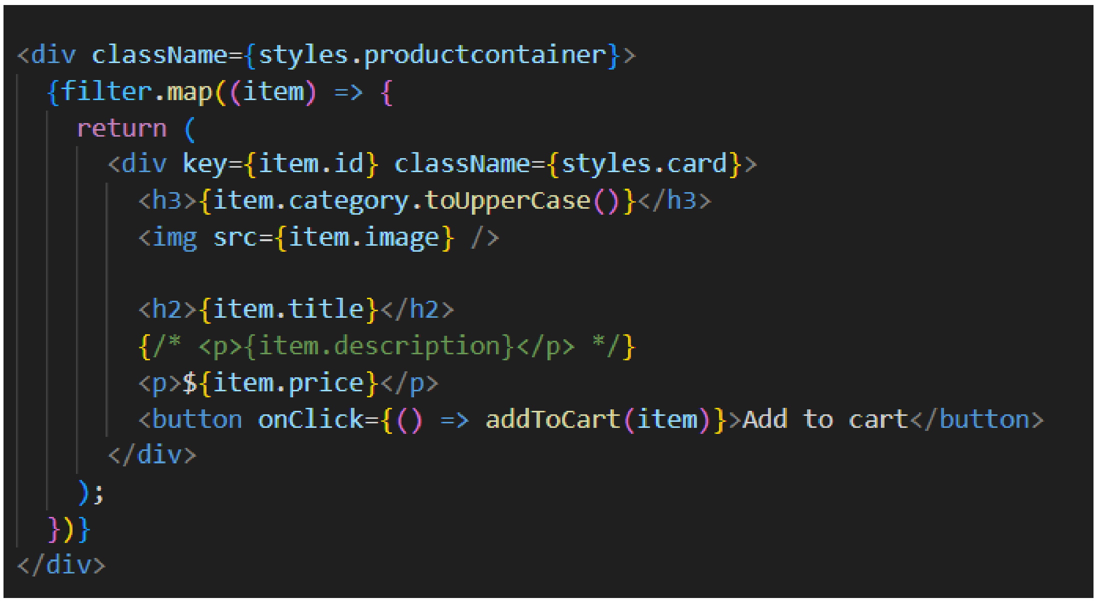
fetch() = already a Promise

This function will not block the rest of your app.It can run tasks that take time (like API calls). It automatically returns a promise.

Fetching data from internet doesn't happen instantly.
Without await, code will continue before data arrives → errors.

And useEffect() is used here to display the product or call it on first fetch of output only 1 time

Display the product and data from the api using map function



```jsx
<div className={styles.productcontainer}>
  {filter.map((item) => {
    return (
      <div key={item.id} className={styles.card}>
        <h3>{item.category.toUpperCase()}</h3>
        <img src={item.image} />

        <h2>{item.title}</h2>
        {/* <p>{item.description}</p> */}
        <p>${item.price}</p>
        <button onClick={() => addToCart(item)}>Add to cart</button>
      </div>
    );
  })}
</div>
```

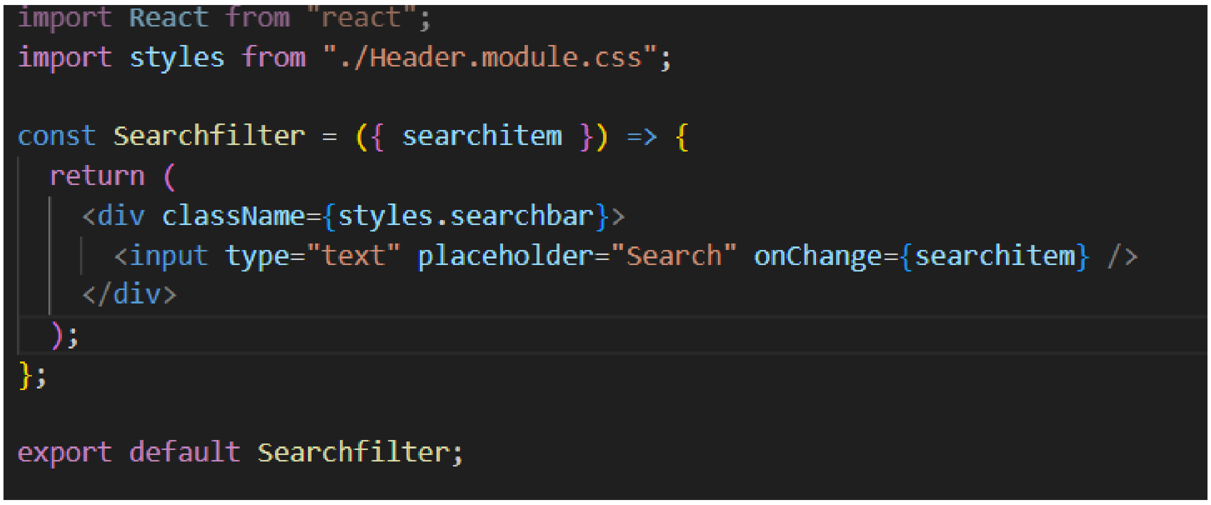
Header.jsx

This contain the navbar and searchfilter.jsx

New thing learn about filtering the data

1.I created separate Searchfilter.jsx page in that I write the input code for user when try to search for it then pass over there the prop of search from the header.jsx where I have make actual function of search filter

Searchfilter.jsx

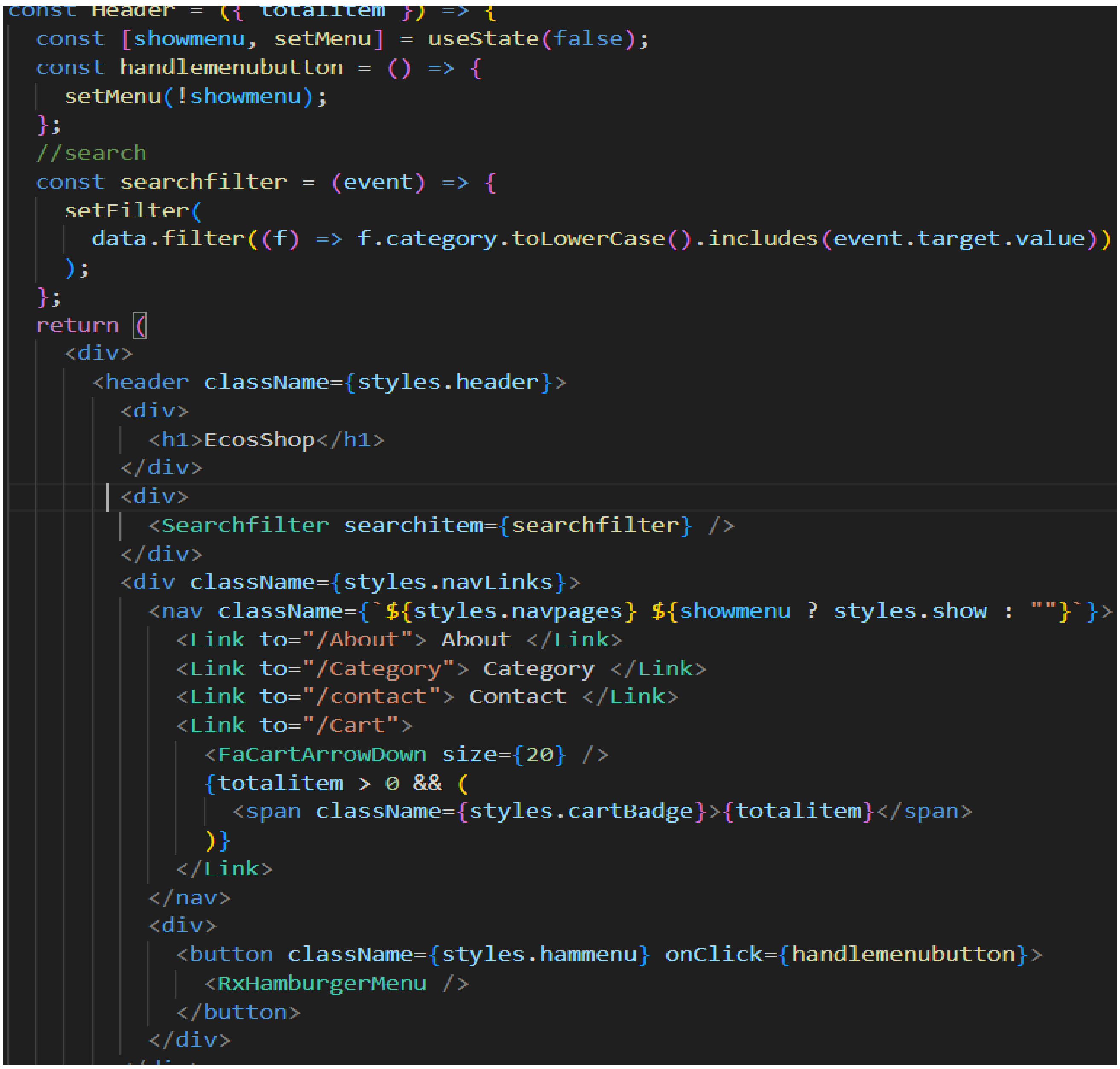


```jsx
import React from "react";
import styles from "./Header.module.css";

const Searchfilter = ({ searchitem }) => {
  return (
    <div className={styles.searchbar}>
      <input type="text" placeholder="Search" onChange={searchitem} />
    </div>
  );
};

export default Searchfilter;
```

Header.jsx with filter function



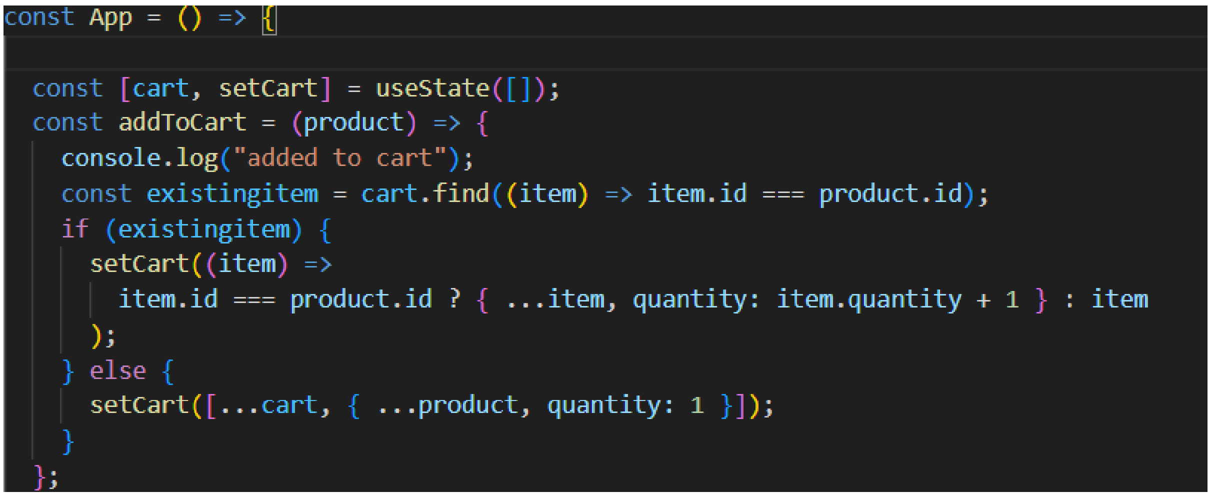
```
const Header = ({ totalitem }) => {
  const [showmenu, setMenu] = useState(false);
  const handlemenubutton = () => {
    setMenu(!showmenu);
  };
  //search
  const searchfilter = (event) => {
    setFilter(
      data.filter((f) => f.category.toLowerCase().includes(event.target.value))
    );
  };
  return (
    <div>
      <header className={styles.header}>
        <div>
          <h1>EcosShop</h1>
        </div>
        <div>
          <Searchfilter searchitem={searchfilter} />
        </div>
        <div className={styles.navLinks}>
          <nav className={`${styles.navpages} ${showmenu ? styles.show : ""}`}>
            <Link to="/About"> About </Link>
            <Link to="/Category"> Category </Link>
            <Link to="/contact"> Contact </Link>
            <Link to="/Cart">
              <FaCartArrowDown size={20} />
              {totalitem > 0 && (
                <span className={styles.cartBadge}>{totalitem}</span>
              )}
            </Link>
          </nav>
          <div>
            <button className={styles.hammenu} onClick={handlemenubutton}>
              <RxHamburgerMenu />
            </button>
          </div>
```

Cartpage.jsx

It is main hero of my website which I created using API it contain lot of function with logic behind the add to cart

For this I create all the logic for the cart inside the App.jsx and pass that each function into the routes of cart which pass in the same App.jsx

1.addToCart function



```
const App = () => {

  const [cart, setCart] = useState([]);
  const addToCart = (product) => {
    console.log("added to cart");
    const existingitem = cart.find((item) => item.id === product.id);
    if (existingitem) {
      setCart((item) =>
        item.id === product.id ? { ...item, quantity: item.quantity + 1 } : item
      );
    } else {
      setCart([...cart, { ...product, quantity: 1 }]);
    }
  };
```

const existingitem = cart.find((item) => item.id === product.id);

- “Look inside the cart...
- find if an item already has the same id
- If found → store it in existingitem
- If not found → existingitem will be undefined.”
- We don’t add the whole item again
- We only increase its quantity by 1

```
setCart((item) =>
        item.id === product.id
          ? { ...item, quantity: item.quantity + 1 }
          : item
      );
```

- “Go through every item in the cart”
- If the item has same ID → update quantity
- Otherwise → keep it same

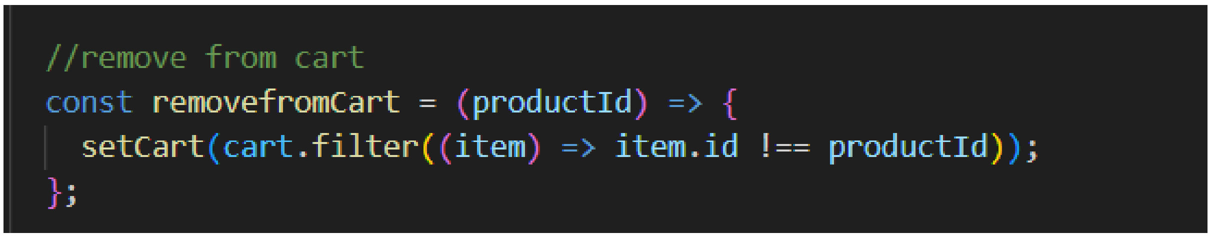
```
} else {
  setCart([...cart, { ...product, quantity: 1 }]);
}
```

“Add the new product to the cart and start quantity from 1.”

...cart = keep old cart
{...product, quantity: 1 } = new product + default quantity

[If the item exists → update quantity
If it doesn’t → add it.]

2.removefrom cart function



```
//remove from cart
const removefromCart = (productId) => {
  setCart(cart.filter((item) => item.id !== productId));
};
```

Step 1: click "Remove"

You pass the product ID you want to delete.

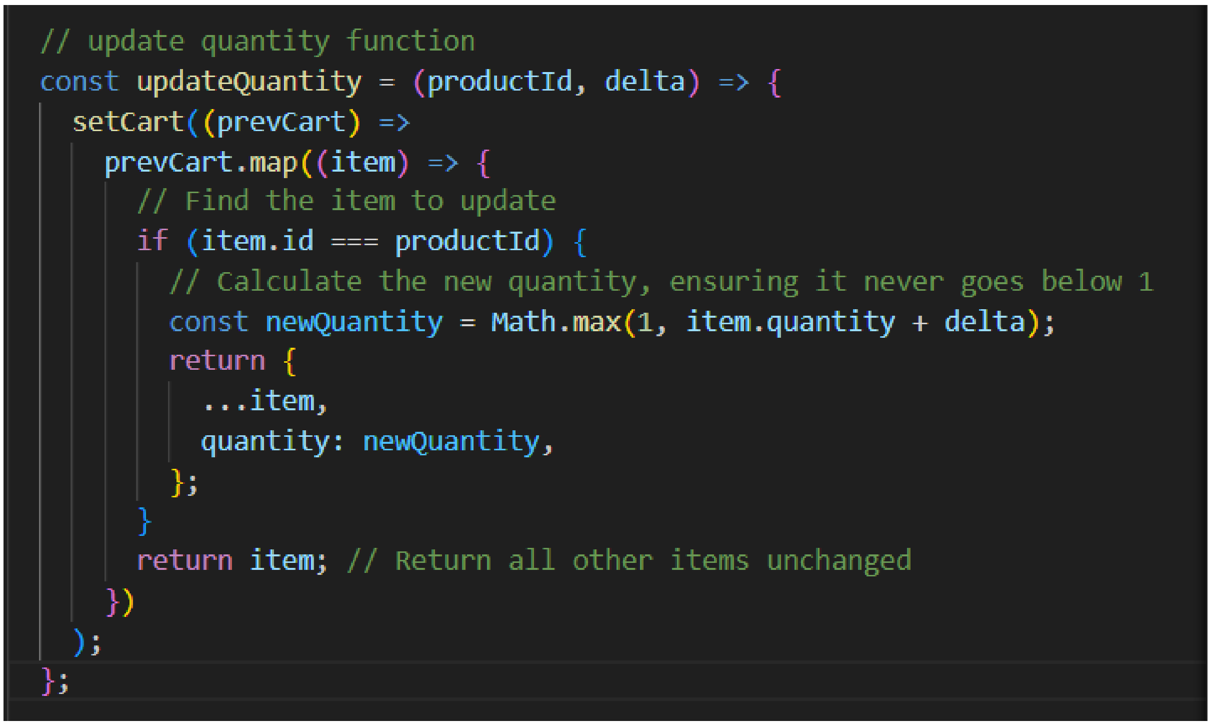
Step 2: filter creates a new cart

```
cart.filter((item) => item.id !== productId)
```

This means:

- Go through every item in the cart
- **KEEP only those items whose ID is NOT equal** to the productId
- This automatically **removes the matching item**

3.Update cart quantity function



```
// update quantity function
const updateQuantity = (productId, delta) => {
  setCart((prevCart) =>
    prevCart.map((item) => {
      // Find the item to update
      if (item.id === productId) {
        // Calculate the new quantity, ensuring it never goes below 1
        const newQuantity = Math.max(1, item.quantity + delta);
        return {
          ...item,
          quantity: newQuantity,
        };
      }
      return item; // Return all other items unchanged
    })
  );
};
```

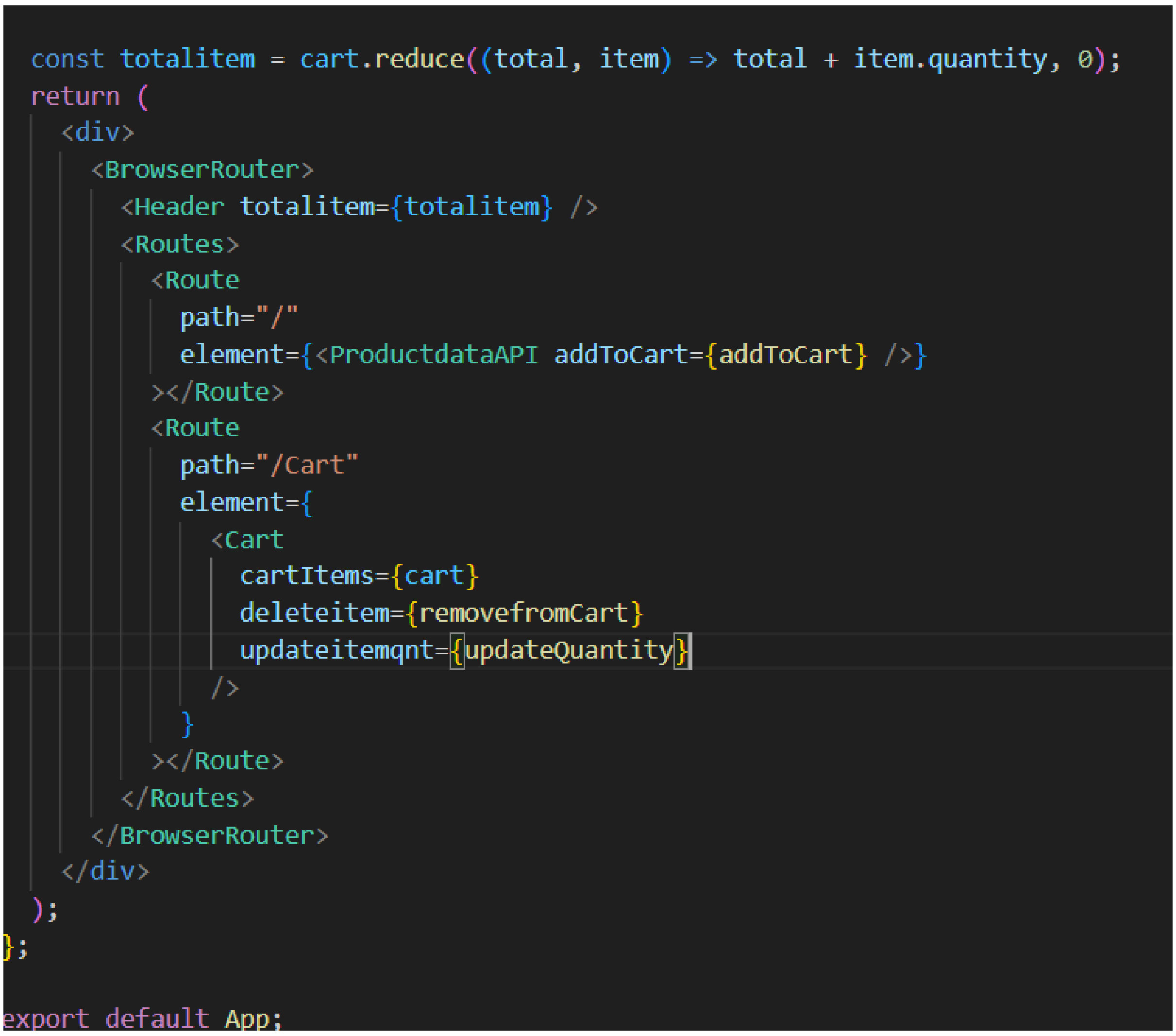
4.total quantity display above the cart symbol

const totalitem = cart.reduce((total, item) => total + item.quantity, 0);

- **It calculates the total number of items in the cart.**
- **cart.reduce()**
- **reduce() goes through the entire cart one item at a time and keeps adding values.**
- **(total, item) => total + item.quantity, 0**
- **→ 0 means: "start total from 0"**

reduce() starts from 0 and keeps adding each item's quantity.

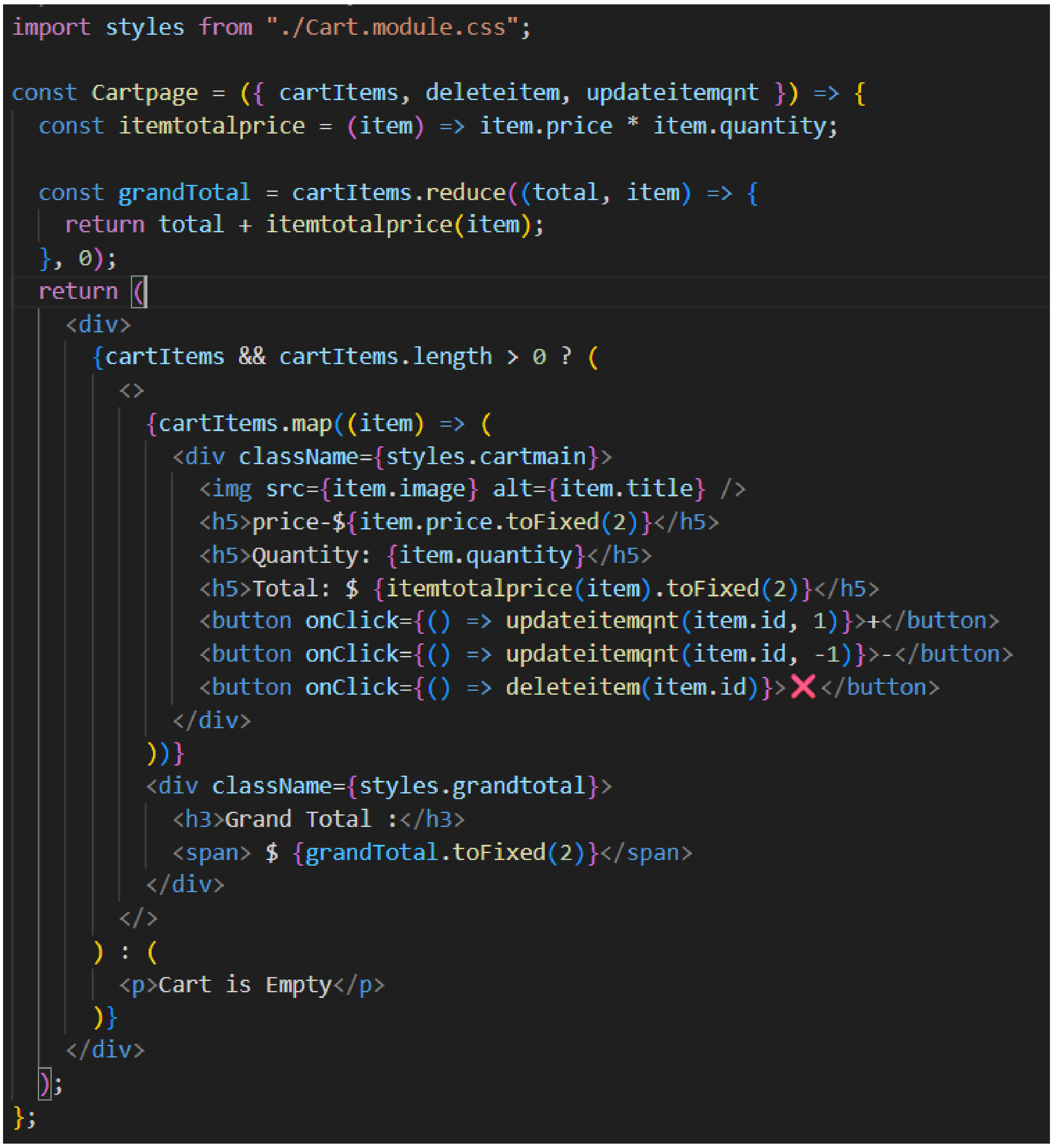
4.Now pass that each function to the routes of cart



```
const totalitem = cart.reduce((total, item) => total + item.quantity, 0);
return (
  <div>
    <BrowserRouter>
      <Header totalitem={totalitem} />
      <Routes>
        <Route
          path="/"
          element={<ProductdataAPI addToCart={addToCart} />}
        ></Route>
        <Route
          path="/Cart"
          element={
            <Cart
              cartItems={cart}
              deleteitem={removefromCart}
              updateitemqnt={updateQuantity}
            />
          }
        ></Route>
      </Routes>
    </BrowserRouter>
  </div>
);
};

export default App;
```

Then pass that cart parent prop in the main cartpage.jsx

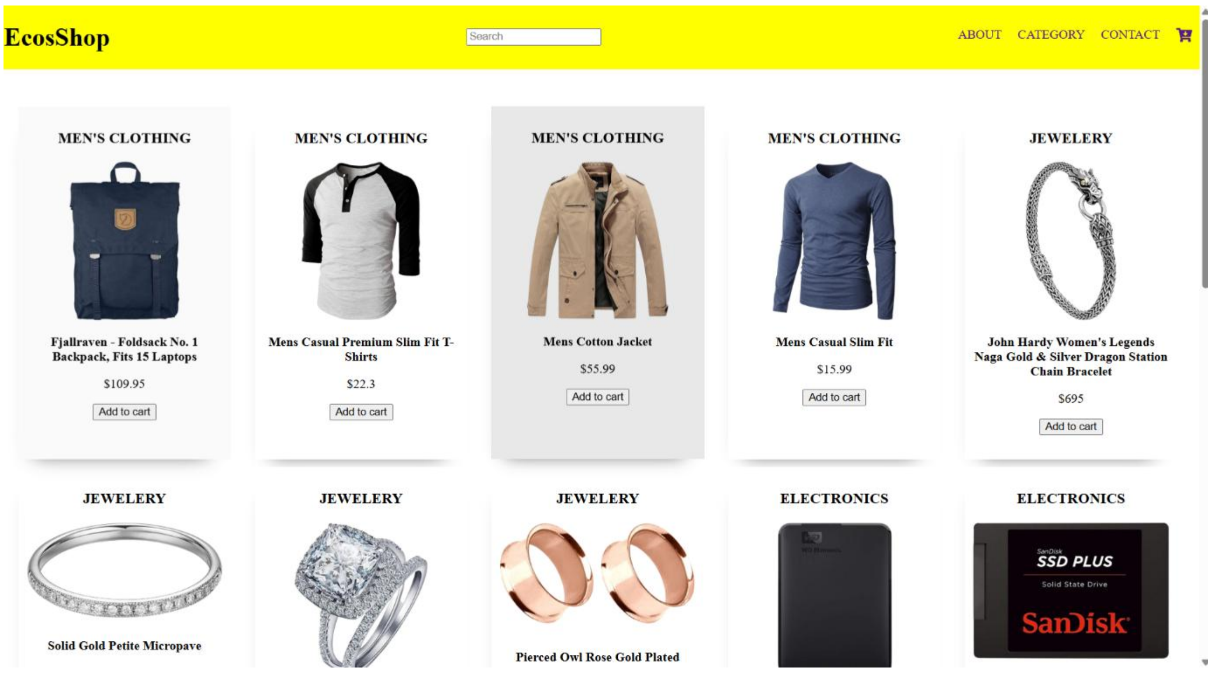


```jsx
import styles from "./Cart.module.css";

const Cartpage = ({ cartItems, deleteitem, updateitemqnt }) => {
  const itemtotalprice = (item) => item.price * item.quantity;

  const grandTotal = cartItems.reduce((total, item) => {
    return total + itemtotalprice(item);
  }, 0);
  return (
    <div>
      {cartItems && cartItems.length > 0 ? (
        <>
          {cartItems.map((item) => (
            <div className={styles.cartmain}>
              <img src={item.image} alt={item.title} />
              <h5>price-${item.price.toFixed(2)}</h5>
              <h5>Quantity: {item.quantity}</h5>
              <h5>Total: $ {itemtotalprice(item).toFixed(2)}</h5>
              <button onClick={() => updateitemqnt(item.id, 1)}>+</button>
              <button onClick={() => updateitemqnt(item.id, -1)}>-</button>
              <button onClick={() => deleteitem(item.id)}>❌</button>
            </div>
          ))}
          <div className={styles.grandtotal}>
            <h3>Grand Total :</h3>
            <span> $ {grandTotal.toFixed(2)}</span>
          </div>
        </>
      ) : (
        <p>Cart is Empty</p>
      )}
    </div>
  );
};
```

output



EcosShop

Search

ABOUT CATEGORY CONTACT

MEN'S CLOTHING
Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops
$109.95
Add to cart

MEN'S CLOTHING
Mens Casual Premium Slim Fit T-Shirts
$22.3
Add to cart

MEN'S CLOTHING
Mens Cotton Jacket
$55.99
Add to cart

MEN'S CLOTHING
Mens Casual Slim Fit
$15.99
Add to cart

JEWELERY
John Hardy Women's Legends Naga Gold & Silver Dragon Station Chain Bracelet
$695
Add to cart

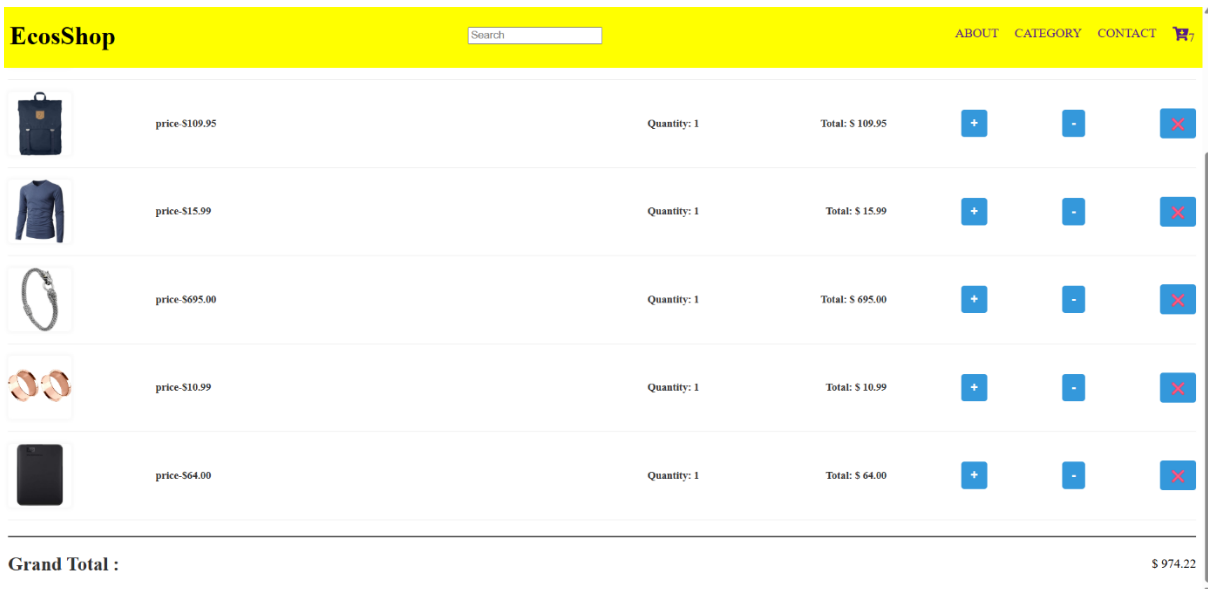
JEWELERY
Solid Gold Petite Micropave

JEWELERY

JEWELERY
Pierced Owl Rose Gold Plated

ELECTRONICS

ELECTRONICS



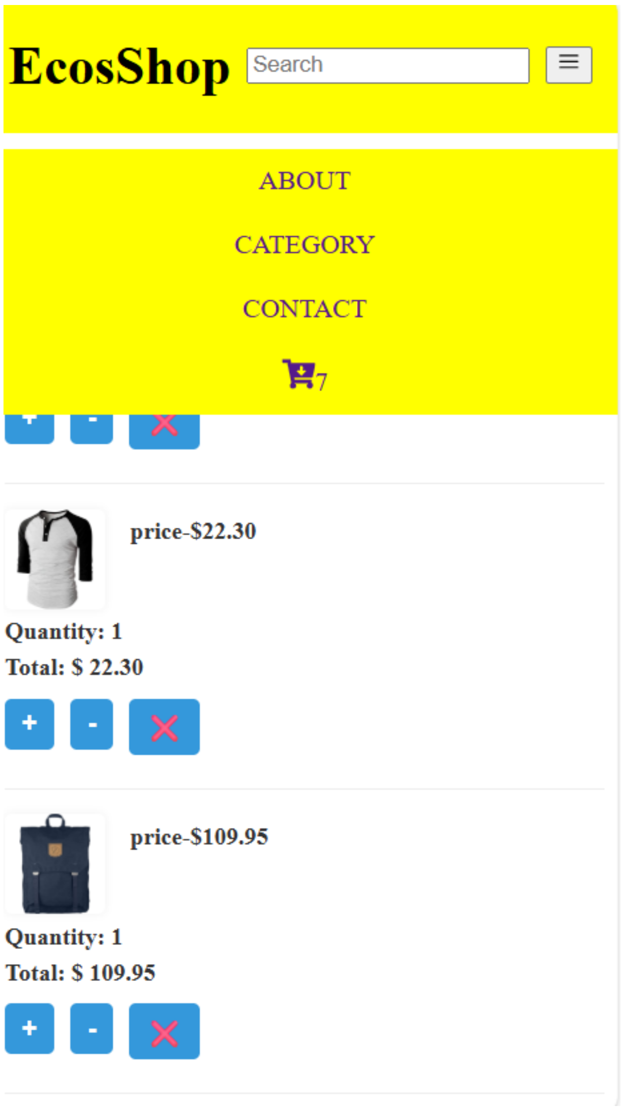
EcosShop

Search

ABOUT CATEGORY CONTACT 7

	price-$109.95	Quantity: 1	Total: $ 109.95	+	-	×
	price-$15.99	Quantity: 1	Total: $ 15.99	+	-	×
	price-$695.00	Quantity: 1	Total: $ 695.00	+	-	×
	price-$10.99	Quantity: 1	Total: $ 10.99	+	-	×
	price-$64.00	Quantity: 1	Total: $ 64.00	+	-	×

Grand Total : $ 974.22



price-$22.30

Quantity: 1

Total: $ 22.30

price-$109.95

Quantity: 1

Total: $ 109.95