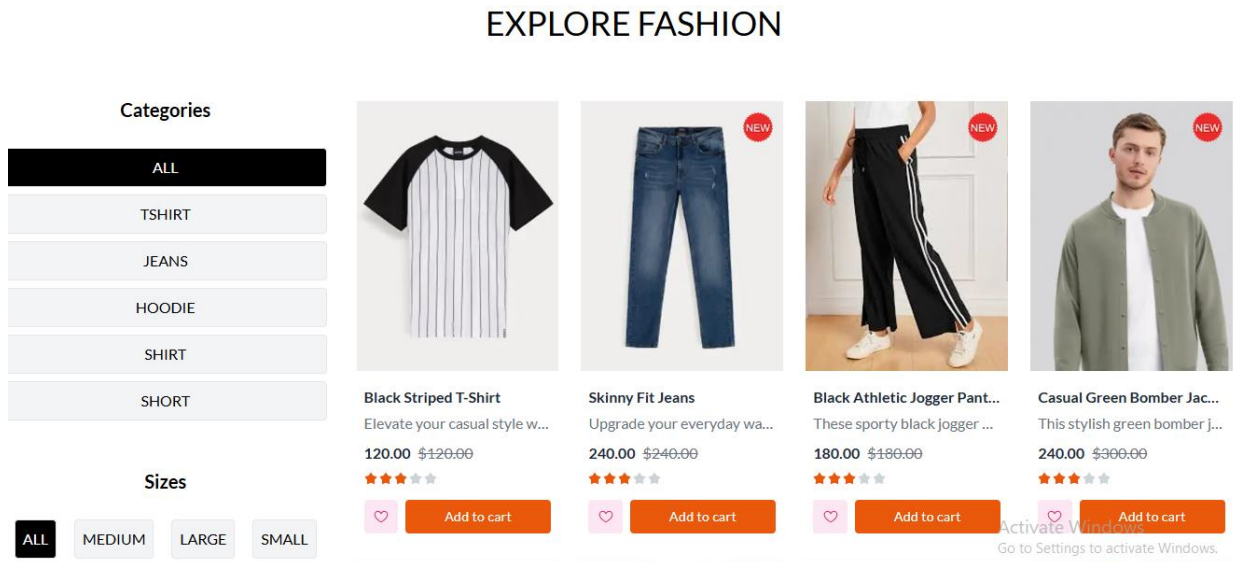# FUNCTIONAL DELIVERABLES
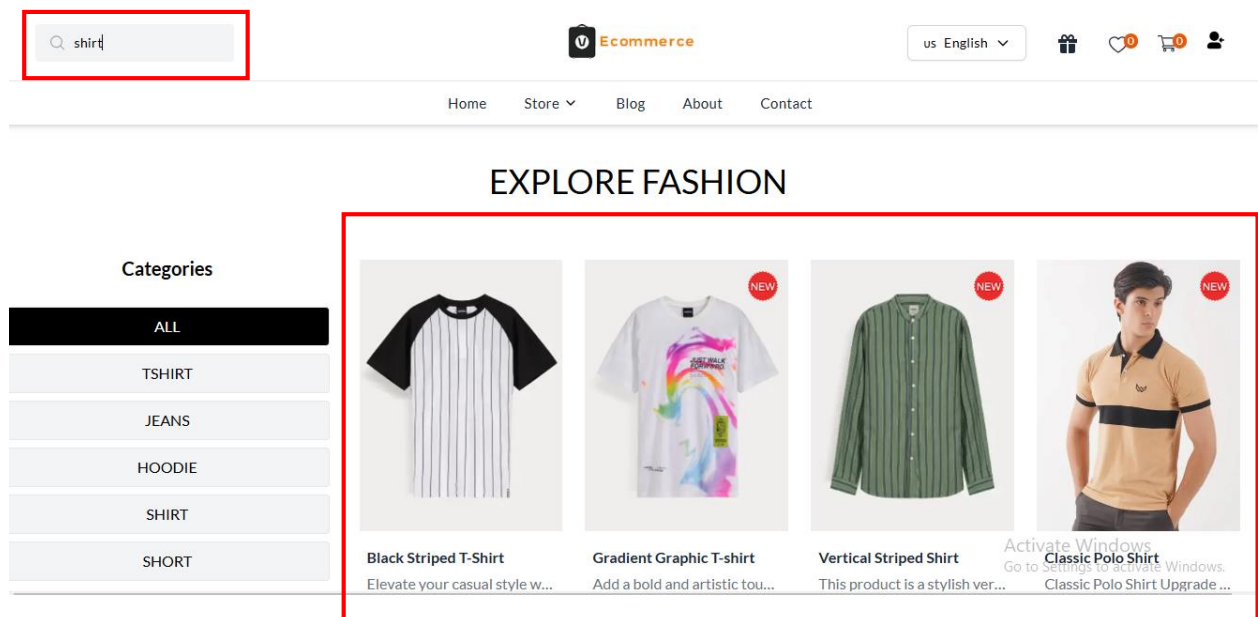
1. Functional Testing

    1.1.   Test Core Features

        1.1.1.   Product listing: Ensure products are displayed correctly.

        1.1.2.   Filters and search: Validate accurate results based on user inputs.

EXPLORE FASHION

Categories

ALL

TSHIRT

JEANS

HOODIE

SHIRT

SHORT

Sizes

ALL MEDIUM LARGE SMALL

Casual Green Bomber Jac...
This stylish green bomber j...
240.00 $300.00
★★★★★

Classic White Pullover Ho...
This white pullover hoodie ...
135.00 $150.00
★★★★★

Classic Black Pullover Hoo...
This versatile and stylish bl...
128.00 $128.00
★★★★★

♡ Add to cart   ♡ Add to cart   ♡ Add to cart

Sizes

ALL MEDIUM LARGE SMALL
EXTRA EXTRA LARGE EXTRA LARGE

Filter by Price

$141(Min)

$500(Max)

145.00 $145.00
★★★★★
♡ Add to cart

114.50 $229.00
★★★★★
♡ Add to cart

180.00 $180.00
★★★★★
♡ Add to cart

190.00 $190.00
★★★★★
♡ Add to cart

COURAGE GRAPHIC T-S...
This product is the Courag...
145.00 $145.00
★★★★★
♡ Add to cart

Checkered Shirt
This men's plaid shirt comb...
178.00 $178.00
★★★★★
♡ Add to cart

1.1.3. Cart operations: Add, update, and remove items from the cart.
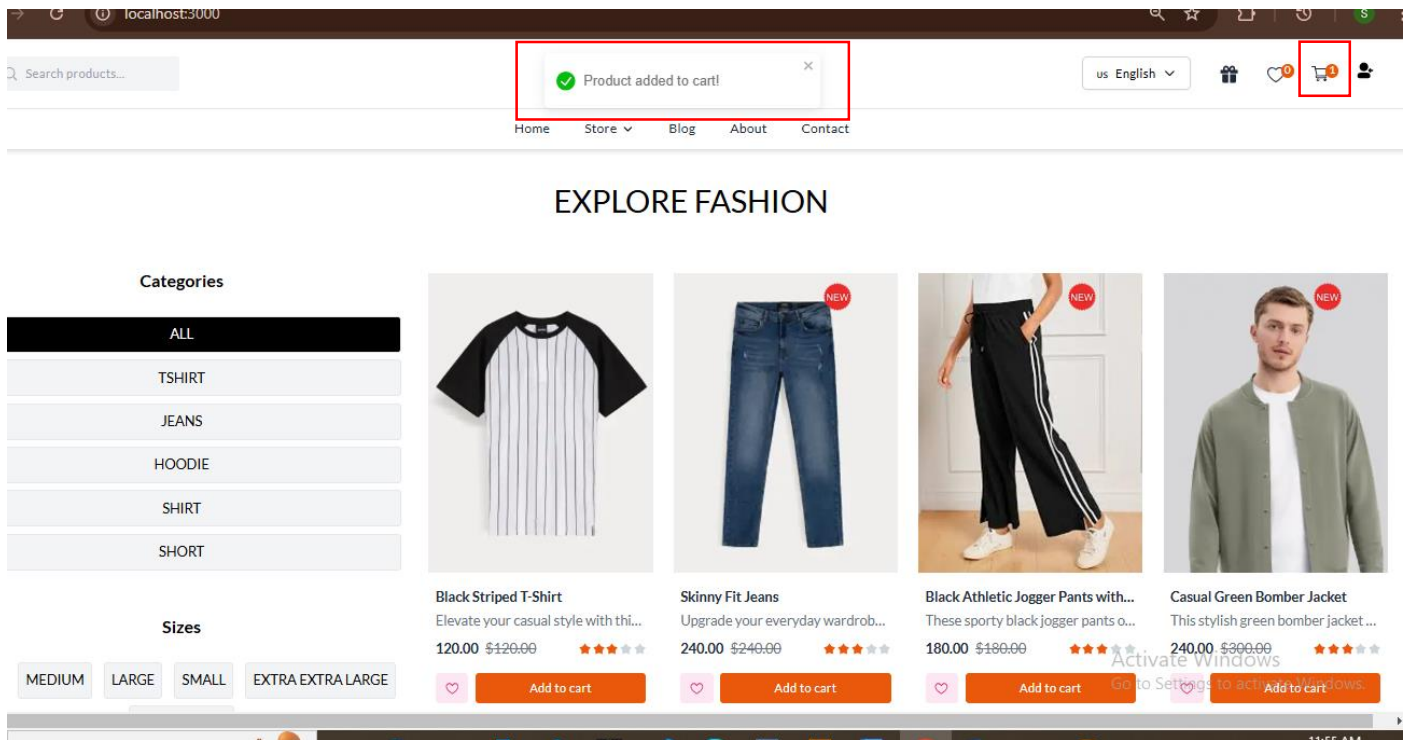
Add to Cart
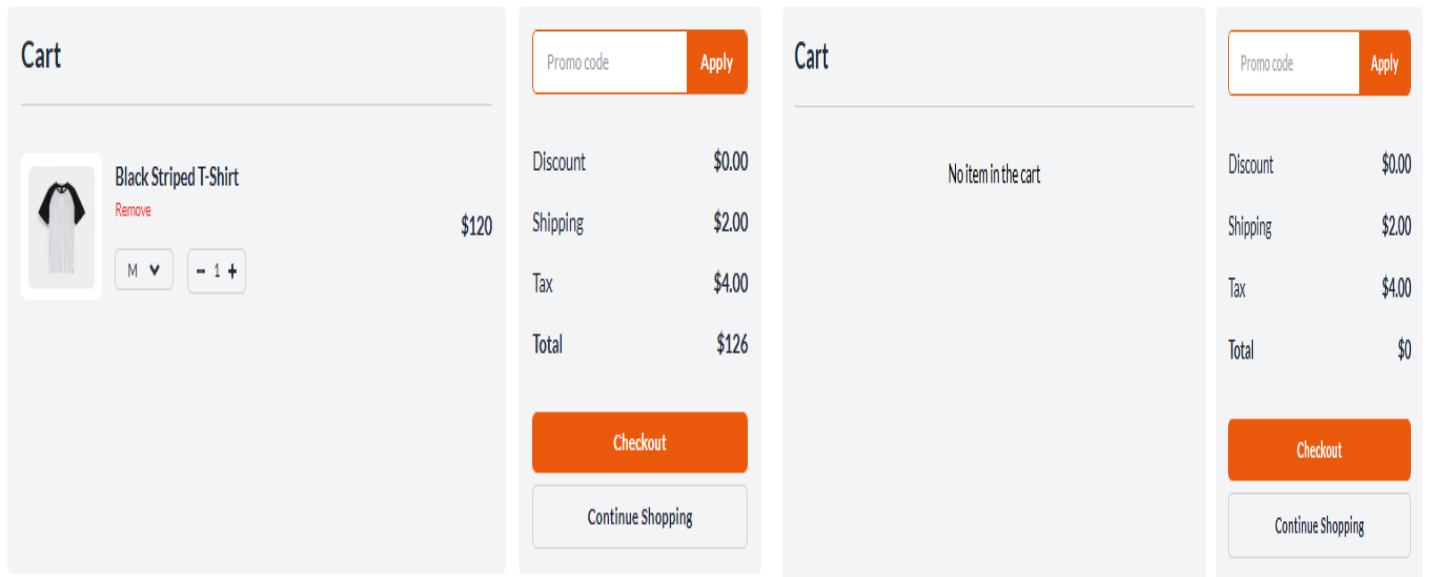
Remove From Cart

1.1.4. Dynamic routing: Verify individual product detail pages load correctly
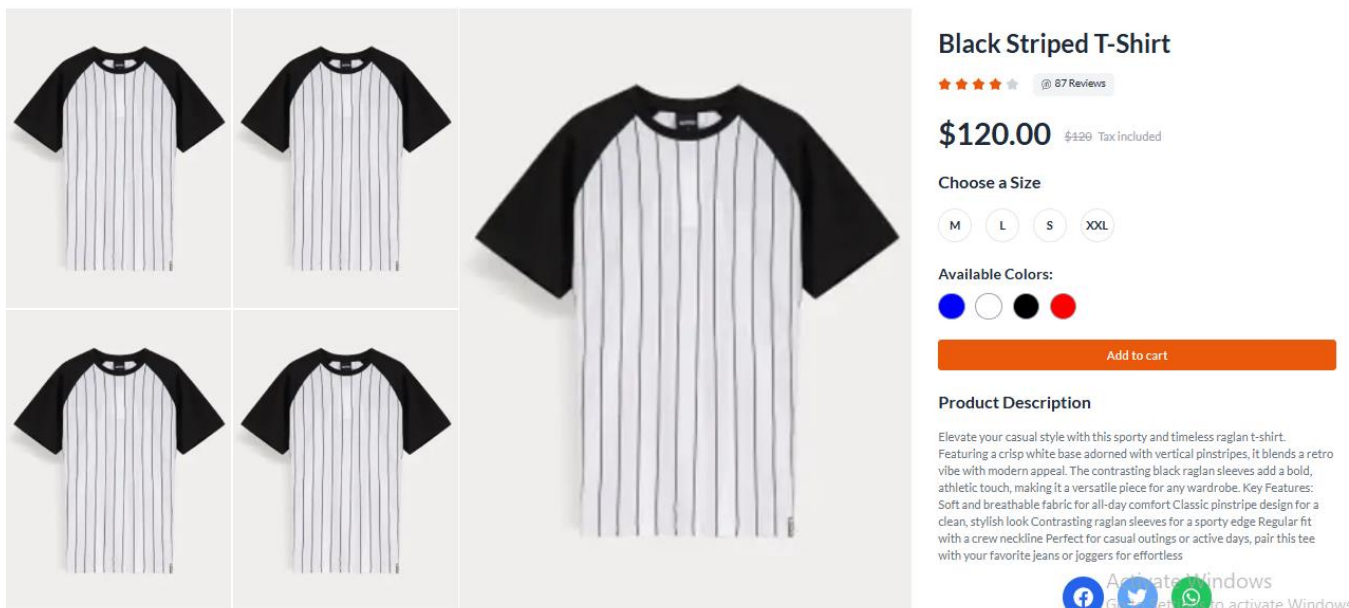
1.2. Testing Tools
1.2.1. Postman
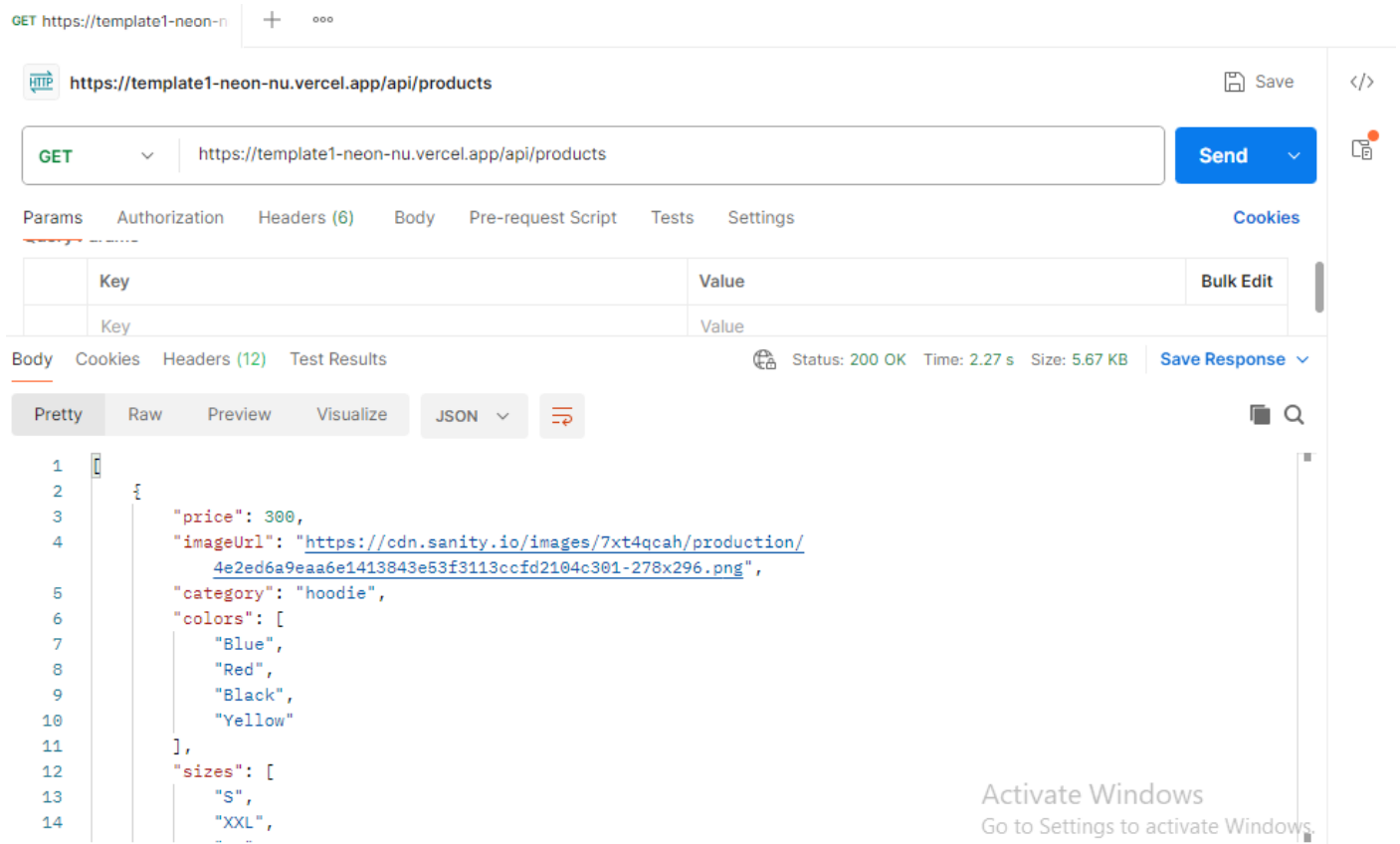
1.2.2. React Testing Library: For component behavior testing.

1.2.2.1. Testing Gift Crad Component in CodeSandbox with React Testing Libarary
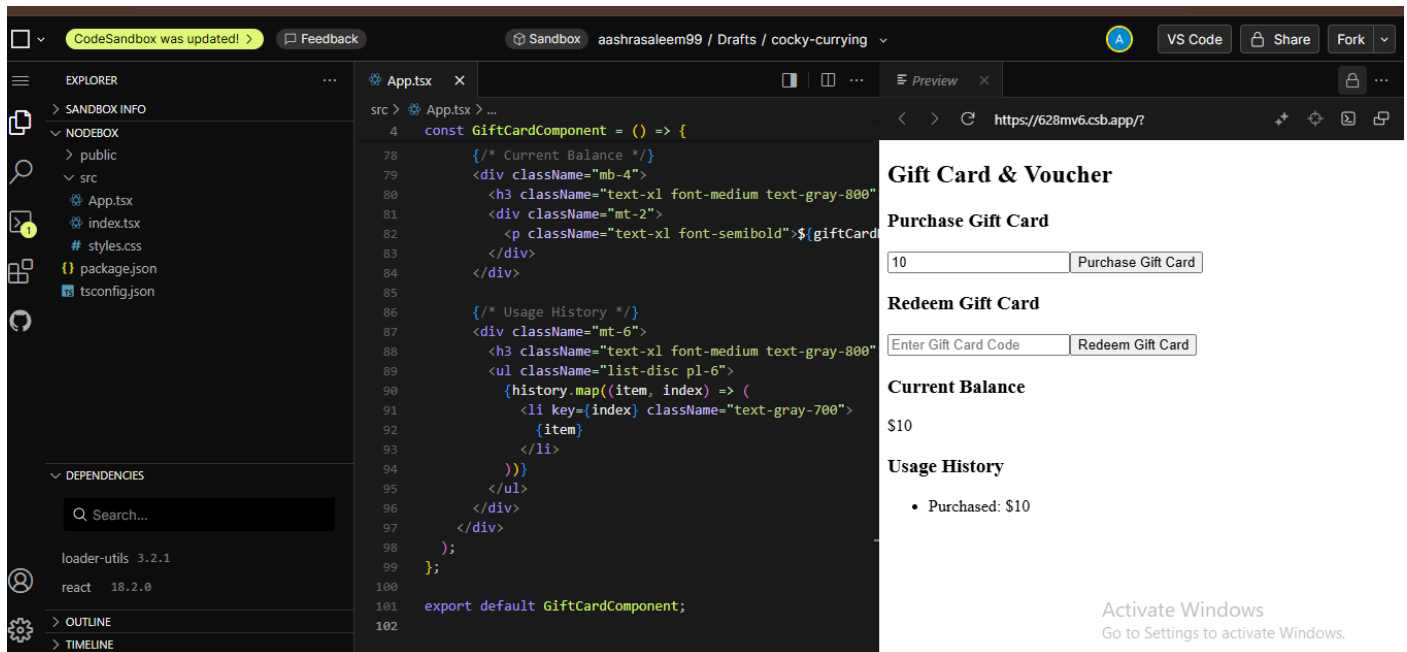
No issue Found

1.2.2.2. Testing Review Section Component in CodeSandbox with React Testing Libarary
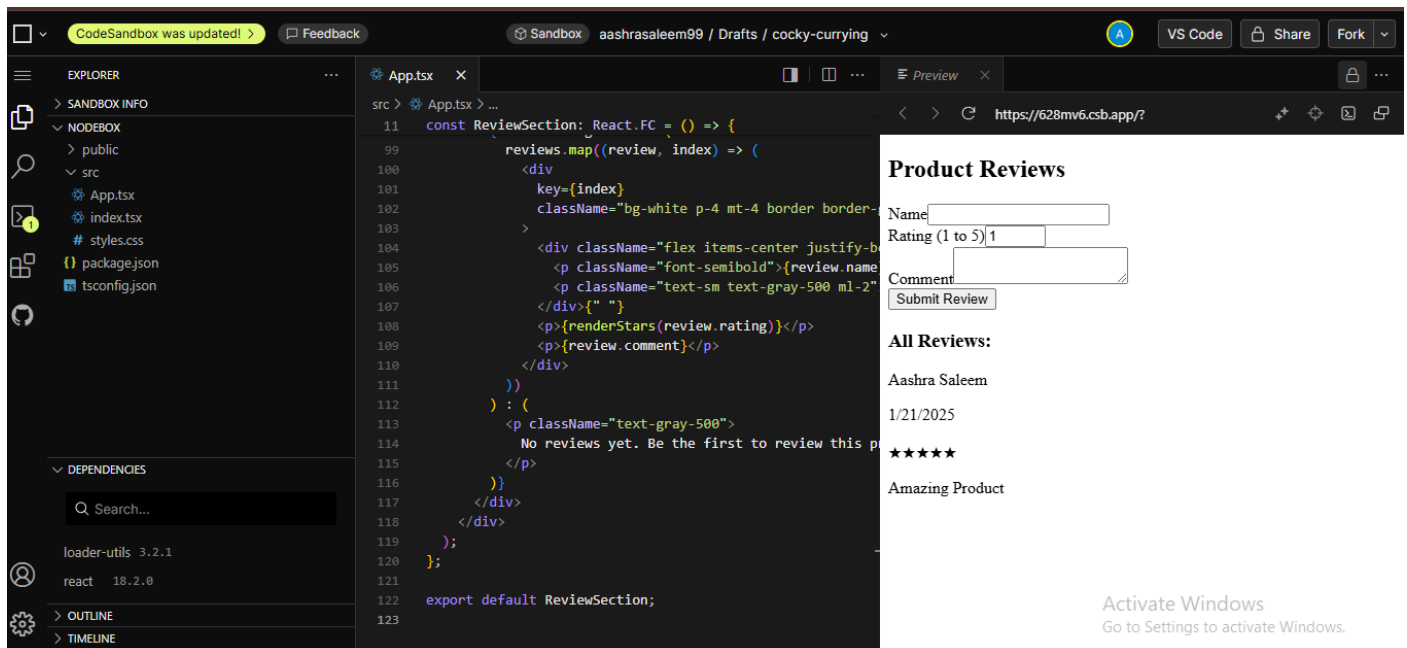
No issue Found

2. Error Handling

2.1. Add Error Messages:

2.1.1. Use try-catch blocks to handle API errors.

```
3.  useEffect(() => {
4.      // Fetch products data
5.      const fetchProducts = async () => {
6.        try {
7.          const data = await GetProductData();
8.          setProducts(data);
9.          setFilteredProducts(data); // Initially show all products
10.       } catch (error) {
11.         console.error("Error fetching products:", error);
12.       }
13.     };
14.
15.     fetchProducts();
16.  }, []);
```

2.1.2. Fallback UI

```
<div>
  {products.length === 0 ? (
    <div className="w-[100vw] h-[100vh] flex justify-center items-center">
      <div className="loader"></div>
    </div>
  ) : (
    <div className="flex md:flex-row flex-col gap-4 p-2">
```

3. Performance Optimization

3.1. Use lazy loading for large images or assets.

```
<Image
          src={urlFor(details.image).width(400).url()}
          width={450}
          height={450}
          quality={75}
          alt="Product"
          className="w-full aspect-[3/4] object-cover object-top shadow-md
hover:scale-[1.05] transition-all duration-300"
        />
```

3.2. Analyze Performance

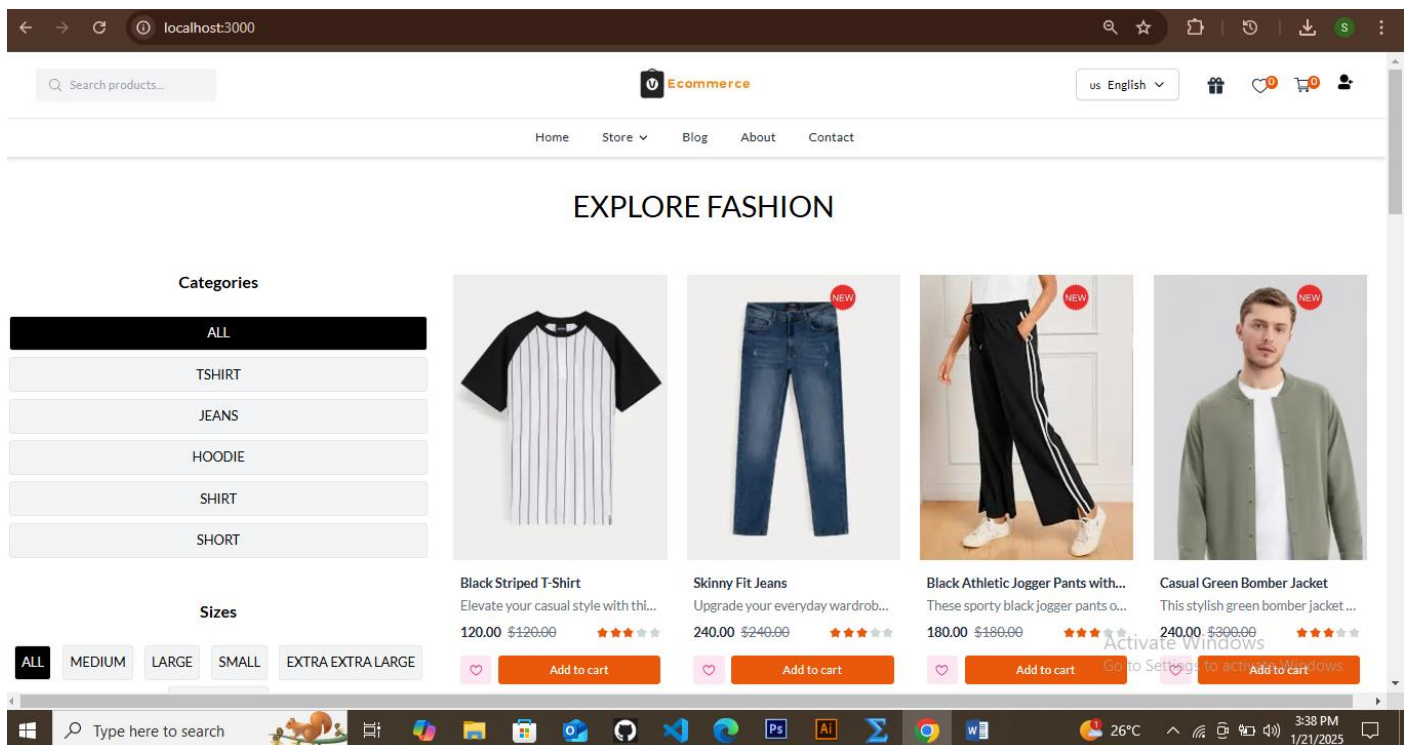3.2.1. Implement fixes such as reducing unused CSS, enabling browser caching, and optimizing JavaScript bundles.
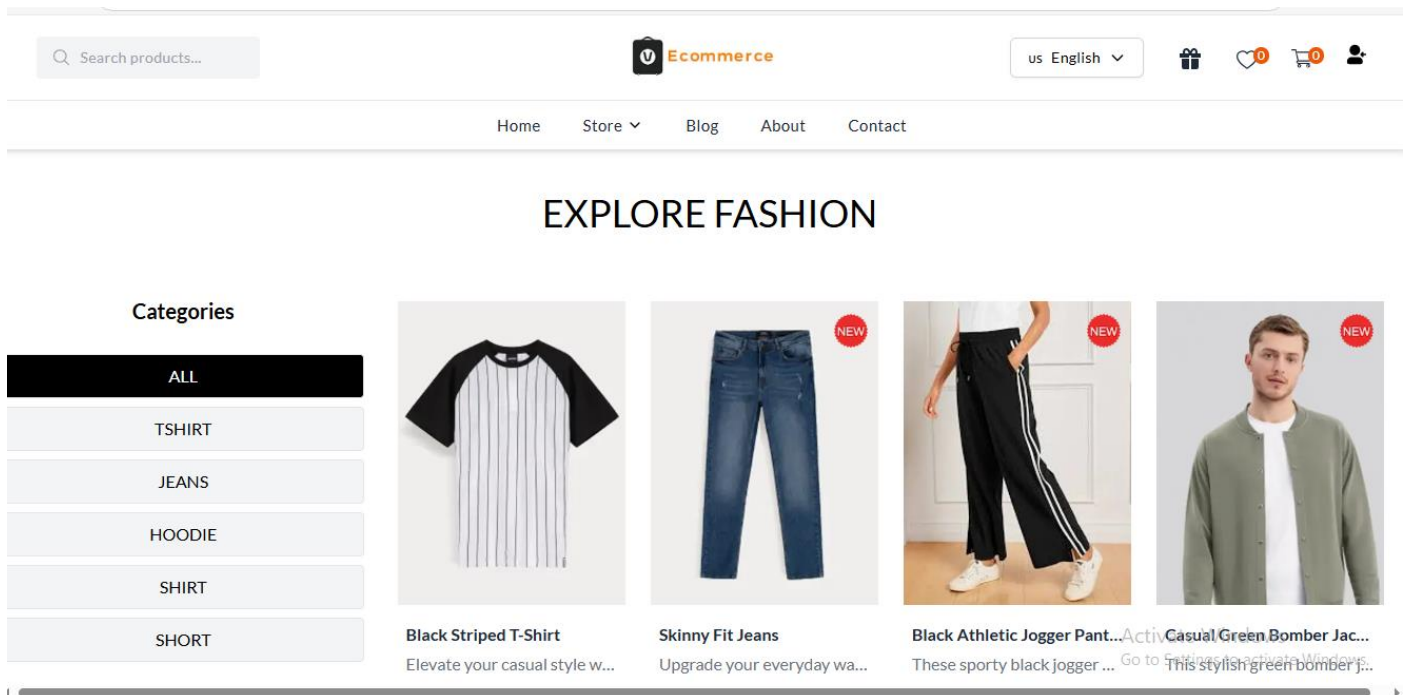
3.3. Test Load Times

2 seconds

4. Cross-Browser and Device Testing

4.1. Browser Testing:

4.1.1. Chrome

4.1.2. Microsoft Edge

4.2. Device Testing:

4.3. Secure API Communication:

4.3.1. Ensure API calls are made over HTTPS.

```
const response = await axios.get('https://template1-neon-nu.vercel.app/api/products')
```

5. User Acceptance Testing (UAT)