

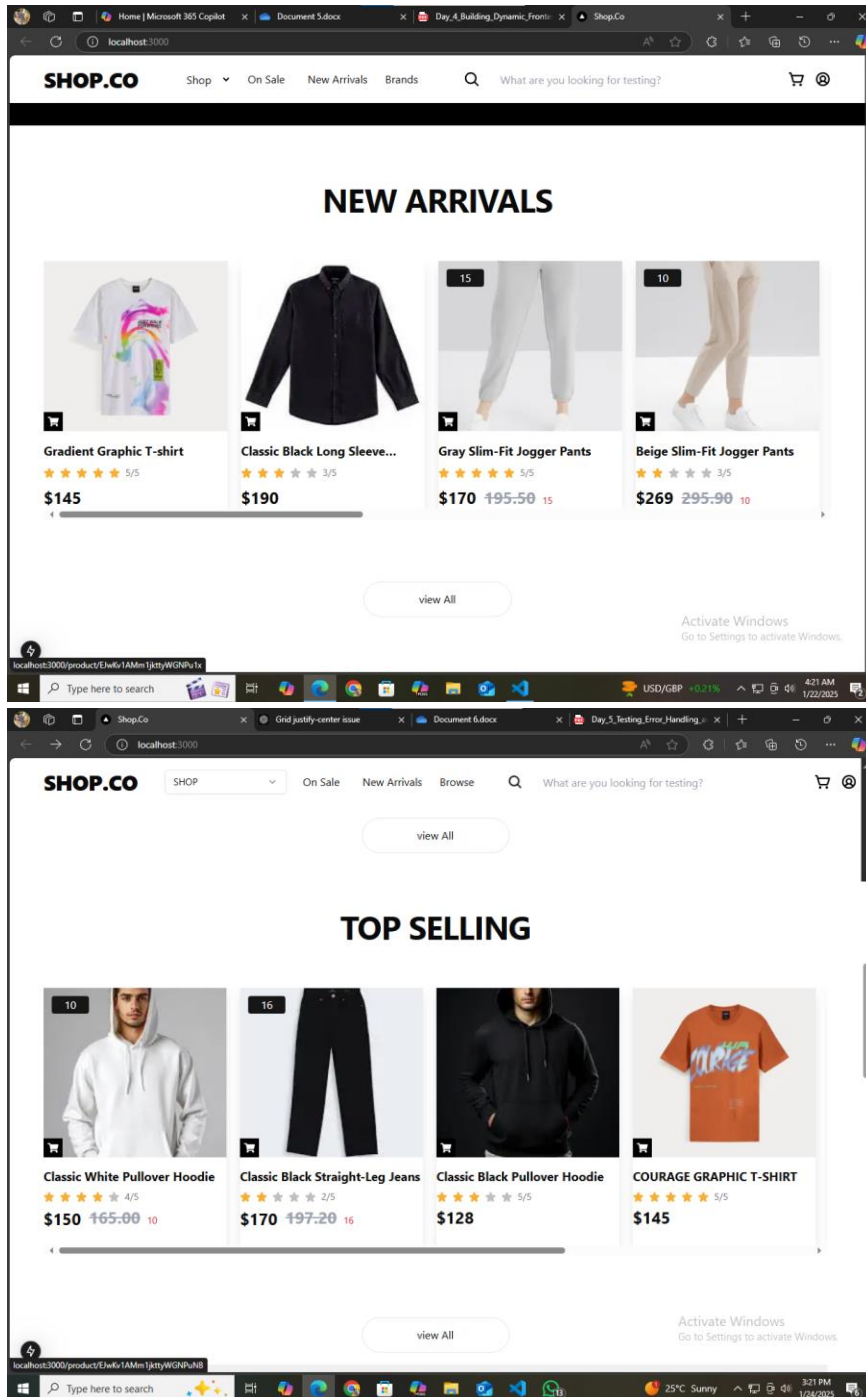
# DAY 5 - TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

## 1. Functional Testing

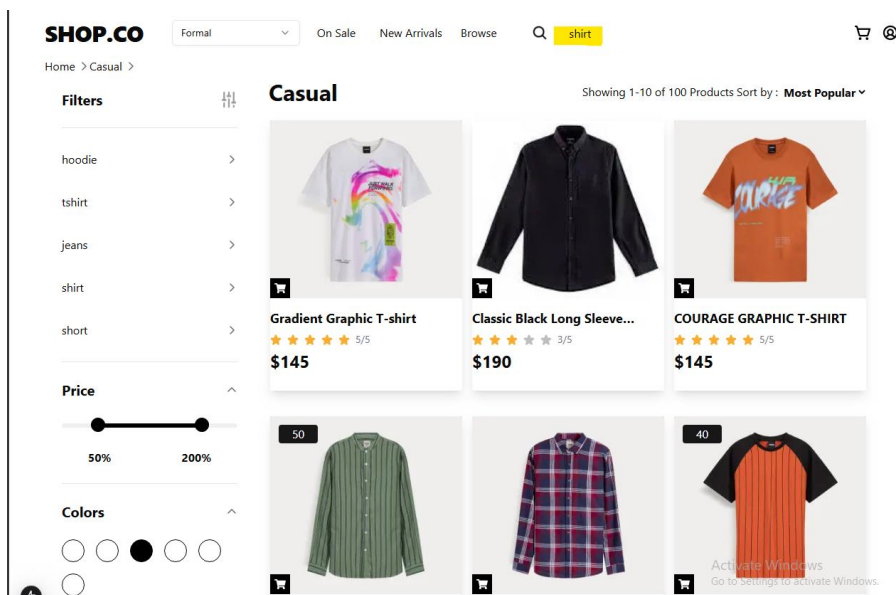
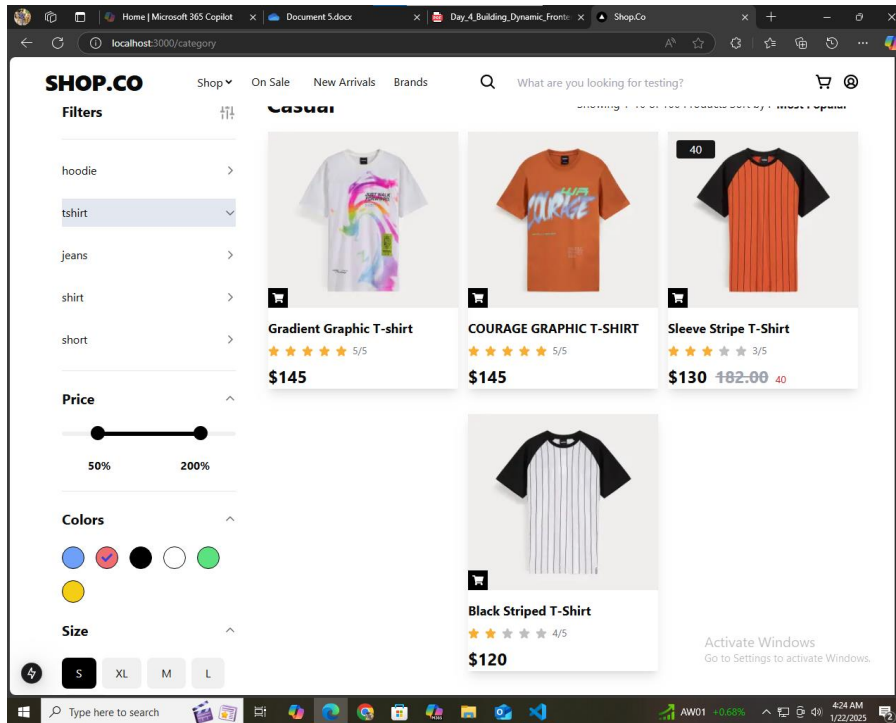
Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status
TC001	Verify product listing on the homepage	1. Visit the website's homepage.2. Scroll to the "New Arrival" section.3. Verify that products are displayed under "New Arrival."4. Scroll to the "Top Selling" section.5. Verify that products are displayed under "Top Products."	Products are displayed in both "New Arrival" and "Top Selling" sections.	Products were displayed correctly in both sections as expected.	<b>Passed</b>
TC002	Validate accurate results based on filters and search	1. Navigate to the search bar.2. Enter a keyword for a product and submit.3. Apply filters in category page (e.g., category, colors, size).4. Verify that results match the search term and applied filters.	The search results and filters produce accurate results based on user input.	Search results and filters worked as expected, showing correct results.	<b>Passed</b>
TC003	Validate cart operations: Add, update, and remove items	1. Navigate to any product .2. Add the product to the cart.3. Go to the cart and update the quantity of the	Products can be added, updated, and removed from the cart	Cart operations worked as expected, allowing add, update, and removal.	<b>Passed</b>

		product.4. Remove the product from the cart.	without errors.		
TC004	Verify individual product detail pages load correctly	1. Navigate to any product in a page.2. Click on a product to open its detail page.3. Verify that the product detail page displays accurate information (e.g., name, price, description, images).	Individual product detail pages load with accurate information.	Product detail pages loaded correctly with accurate details.	<b>Passed</b>

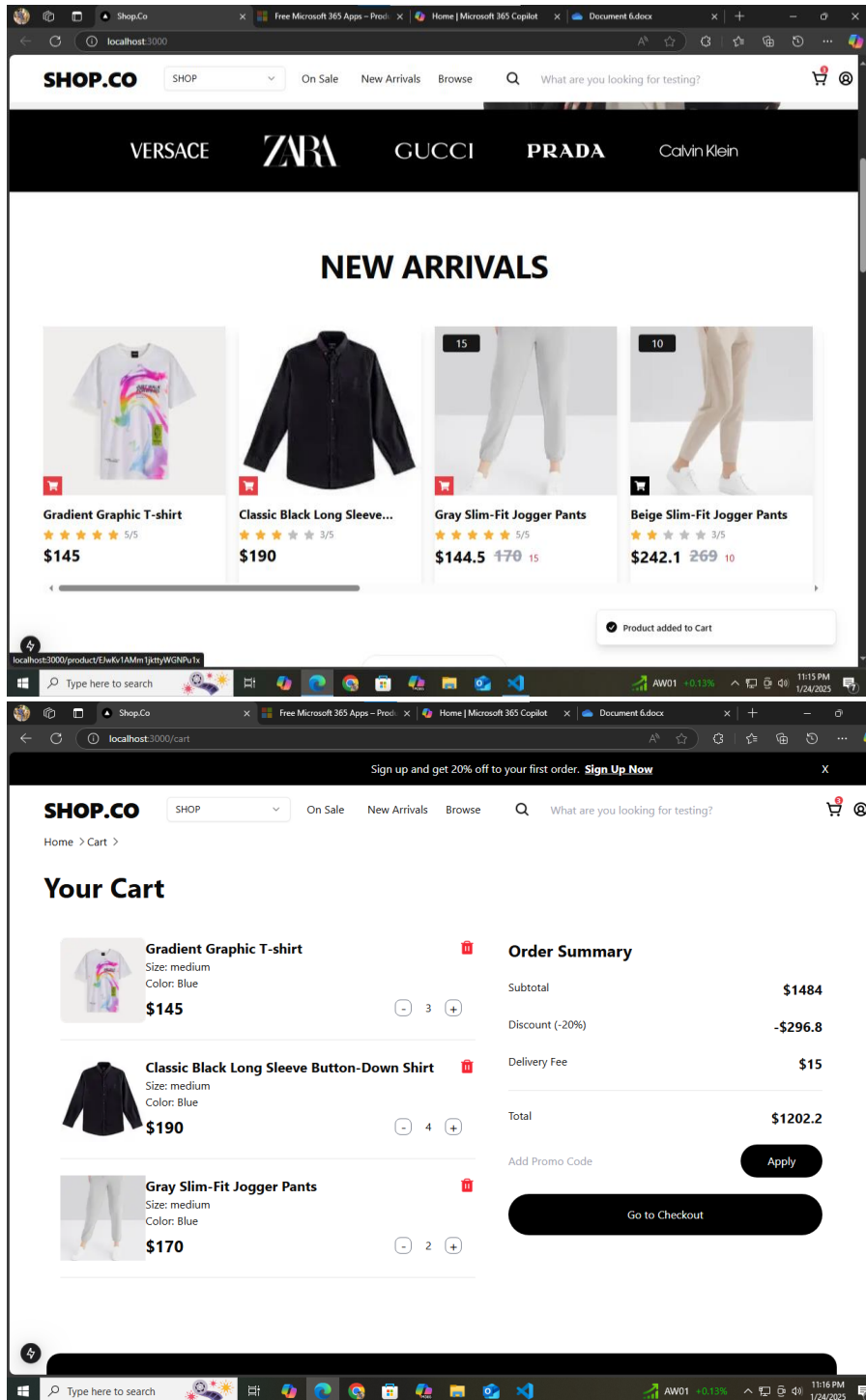
TC001:



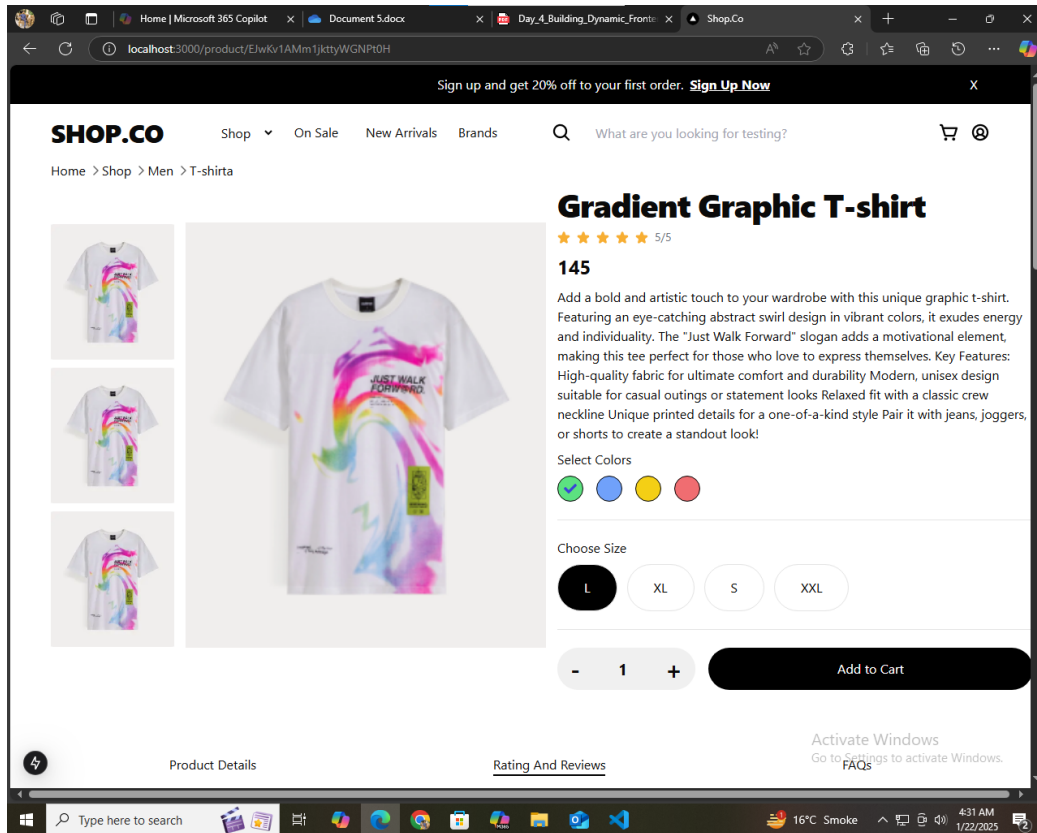
TC002:



TC003:



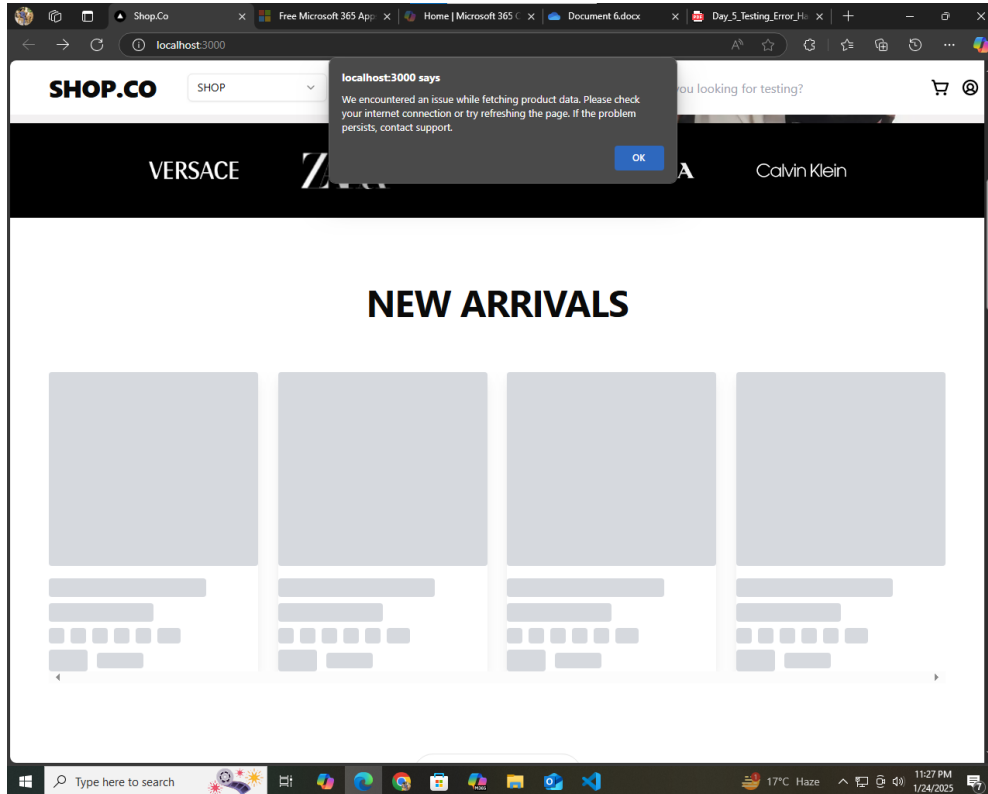
# TC004:



## 2. Error Handling

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status
TC005	Verify error handling using try-catch for console and alert	1. Simulate an error by triggering a function wrapped in a try-catch block.2. Verify that the error is logged in the console.3. Verify that an alert is displayed with the error message.	Errors are logged to the console, and an alert displays the error message.	Error was logged to the console and displayed in an alert successfully.	<b>Passed</b>
TC006	Validate conditional rendering when no products are found	1. Navigate to the categories page.2. Select multiple categories that do not have overlapping products.3. Verify that the message "No Product available" is displayed on the UI.	The message "No Product available" is displayed when no products match the filter.	Message displayed correctly when no products were found.	<b>Passed</b>

**TC005:**





```
14 export default function Home() {
15   useEffect(() => {
16     async function fetchProducts() {
17       const allProducts: Product[] = await client.fetch(`
18         discountPercent,
19         isNew,
20         stars,
21         rating,
22         colors,
23         sizes,
24         "image": image.asset->url
25       `);
26
27       // Check if the fetched data is valid
28       if (!allProducts || allProducts.length === 0) {
29         throw new Error("No products were fetched from Sanity.");
30       }
31
32       const products = {
33         allProducts,
34         newProducts: allProducts.filter((product) => product.isNew === true),
35       };
36
37       console.log("allProducts", allProducts && allProducts);
38
39       // Save to local storage
40       localStorage.setItem("products", JSON.stringify(products));
41
42       // Update state
43       setAllProducts(products.allProducts);
44       setNewProducts(products.newProducts);
45     } catch (err) {
46       alert(
47         "We encountered an issue while fetching product data. Please check your internet connection or try refreshing the page. If t
48       );
49       console.error("Error fetching data from Sanity:", err);
50     }
51   });
52
53   // Fetch products on the client side
54 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

GET /favicon.ico 200 in 80ms  
GET /Frame%20575.png 404 in 97ms

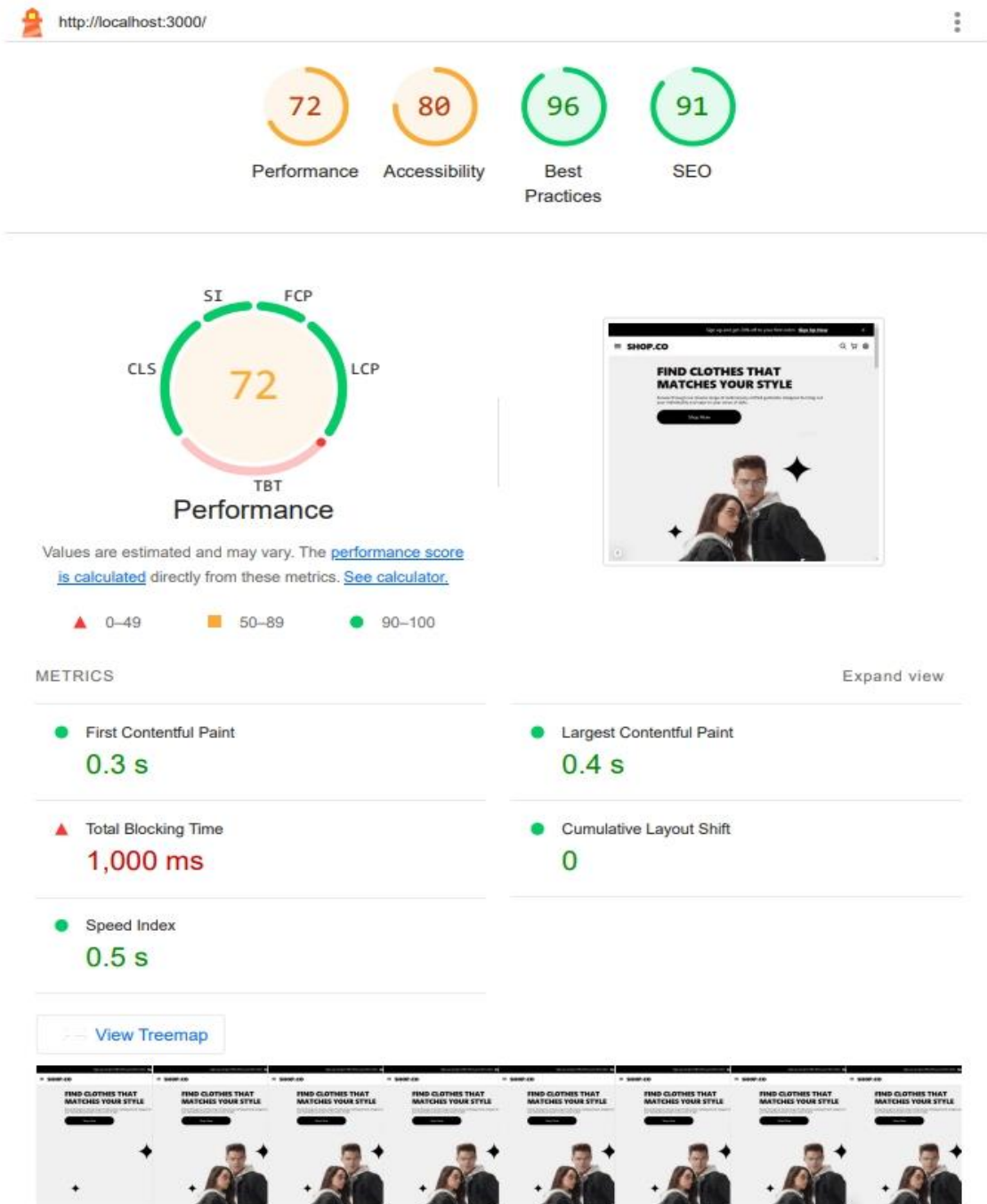
Ln 43, Col 80 Spaces: 2 UTF-8 LF TypeScript JSX Go Live Prettier

Type here to search 17°C Haze 11:28 PM 1/24/2025



# 3. Performance Testing

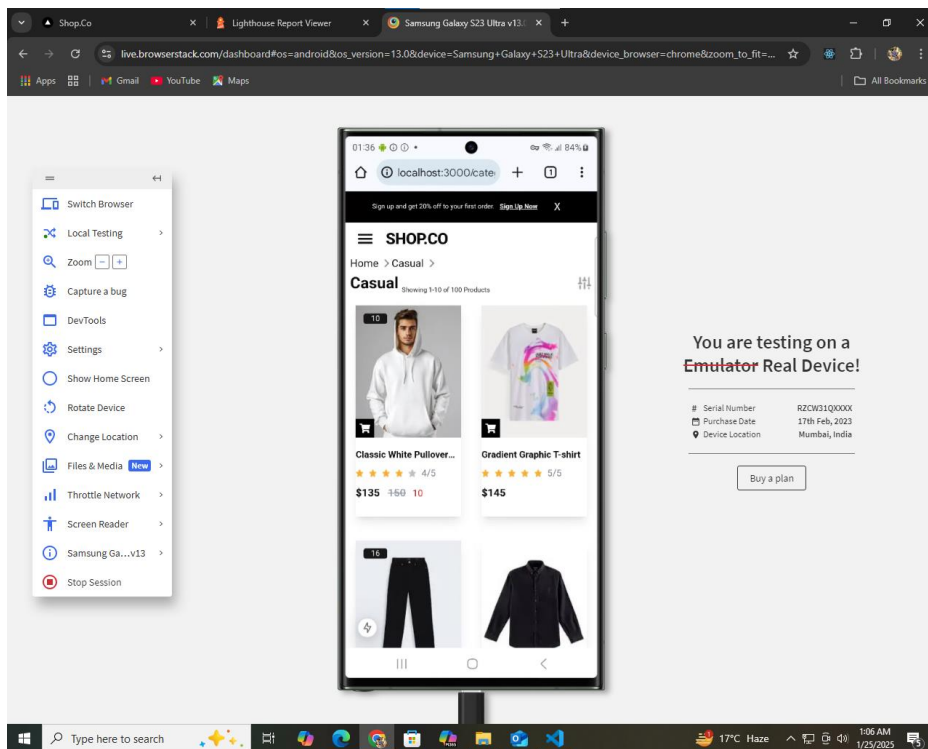
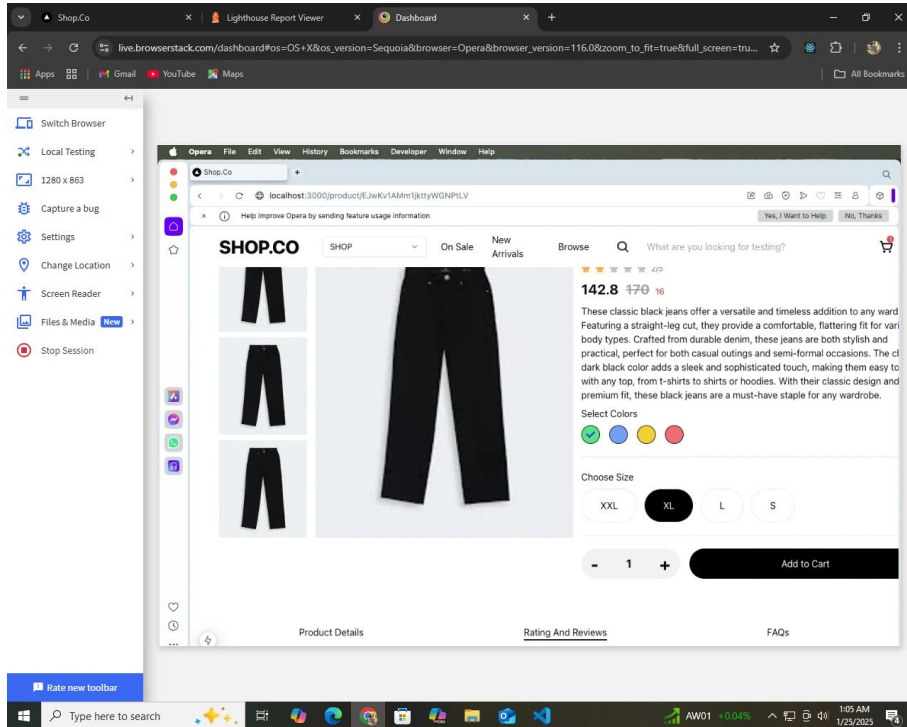
## Lighthouse Report:



## 4. Cross Browser and Device Testing

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status
TC007	Cross-browser and device compatibility testing	1. Use BrowserStack to test the website on different browsers: Safari, Chrome, Edge, and Opera (desktop versions).2. Use BrowserStack to test the website on mobile devices using the same browsers where applicable.3. Verify the website's functionality and layout on each.	The website functions and displays correctly on all tested browsers and devices.	The website functioned and displayed correctly on all tested browsers and devices.	<b>Passed</b>

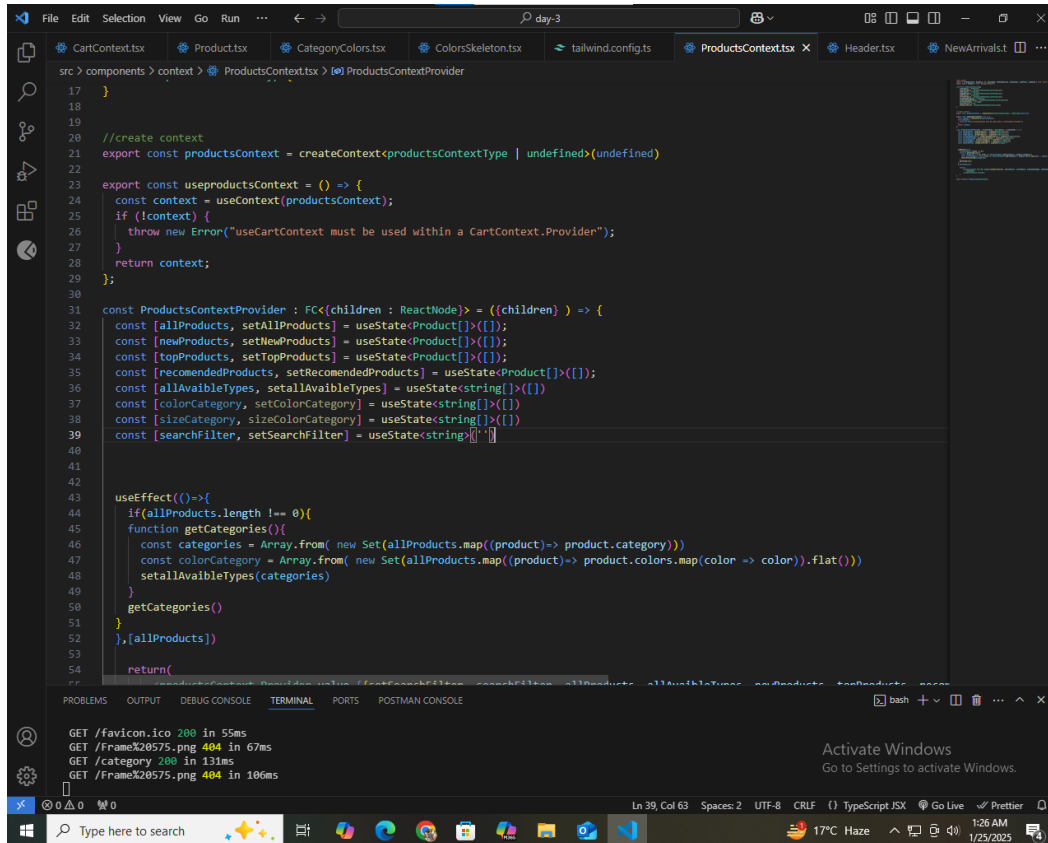
## TC007:



## 5. Security Testing

Type safety has been incorporated into all input fields as a security measure to prevent injection attacks. This ensures that only valid data is processed and submitted, mitigating the risks of SQL injections, cross-site

scripting (XSS), and other malicious input vulnerabilities.



```
17 }
18
19
20 //create context
21 export const productsContext = createContext<productsContextType | undefined>(undefined)
22
23 export const useproductsContext = () => {
24   const context = useContext(productsContext);
25   if (!context) {
26     throw new Error("useCartContext must be used within a CartContext.Provider");
27   }
28   return context;
29 };
30
31 const ProductsContextProvider : FC<{children : ReactNode}> = ({children} ) => {
32   const [allProducts, setAllProducts] = useState<Product[]>([]);
33   const [newProducts, setNewProducts] = useState<Product[]>([]);
34   const [topProducts, setTopProducts] = useState<Product[]>([]);
35   const [recommendedProducts, setRecommendedProducts] = useState<Product[]>([]);
36   const [allAvailableTypes, setallAvaiableTypes] = useState<string[]>([]);
37   const [colorCategory, setColorCategory] = useState<string[]>([]);
38   const [sizeCategory, setSizeCategory] = useState<string[]>([]);
39   const [searchFilter, setSearchFilter] = useState<string>('');
40
41
42
43   useEffect(()=>{
44     if(allProducts.length !== 0){
45       function getCategories(){
46         const categories = Array.from( new Set(allProducts.map((product)-> product.category)))
47         const colorCategory = Array.from( new Set(allProducts.map((product)-> product.colors.map(color -> color)).flat()))
48         setallAvaiableTypes(categories)
49       }
50       getCategories()
51     }
52     ,[allProducts])
53
54     return(
55       <ProductsContextProvider value={productsContext}>{children}</ProductsContextProvider>
56     )
57   })
58 }
```

## 6. User Acceptance Testing (UAT)

The User Acceptance Testing (UAT) for the marketplace was successfully completed by simulating real-world scenarios. Testers interacted with the platform, performing key workflows such as browsing products,

searching for items, and completing the checkout process. Throughout the testing, these workflows were found to be intuitive and seamless, with no errors encountered. This confirms that the user experience is smooth and aligns with expectations, ensuring that the platform is ready for end users without any significant issues.