Day - 5

Testing, Error Handling, and Backend Integration Refinement:

Objective:

The objective of this testing is to validate that all features of the **Nike e-commerce website** function as intended, ensuring a seamless and intuitive user experience. This includes validating the browsing, purchasing, and interactive elements of the website, such as product detail pages, search functionality, shopping cart, and checkout process. The goal is to ensure that all user interactions, from product selection to order confirmation, work smoothly and efficiently, providing customers with a hassle-free shopping experience.

Test Cases for the Features:

1. Product Detail Page:

Test Case ID: TC-001

Test Case Description: Validate Product Detail Page
Test Steps:

1. Open the product listing page.
2. Click on a product.
3. Verify that the product detail page is displayed with product name, description, price, and image.

Expected Result: Product detail page should show accurate product information (name, description, price, image).
Actual Result: Product detail page shows accurate information.
Status: Passed
Remarks: No issues found.

2. Dynamic Search Bar:

Test Case ID: TC-002

Test Case Description: Validate Dynamic Search Bar
Test Steps:

1. Go to the search bar.
2. Enter a search query (e.g., "Nike air max").
3. Verify that relevant products appear based on the search query.

Expected Result: Search results should show products relevant to the search query.
Actual Result: Search results accurately match the query "laptop".
Status: Passed
Severity Level: Medium
Remarks: Results were accurate, but search speed can be improved.

3. Cart Functionality

Test Case ID: `TC-003`

Test Case Description: Validate Cart Functionality

Test Steps:

1. Add an item to the cart.
2. View the cart.
3. Verify that the item is correctly added with the correct quantity and price.
4. Remove an item from the cart.
5. Verify that the item is removed from the cart.

Expected Result: Cart should accurately show the added items, quantities, prices, and allow removing items.
Actual Result: Cart functionality works as expected.
Status: Passed
Severity Level: High
Remarks: All cart operations functioned smoothly.

4. API Error Handling:

Test Case ID: `TC-004`

Test Case Description: Validate API Error Handling
Test Steps:

1. Trigger an invalid API request (e.g., enter invalid product ID).
2. Check if the system handles the error gracefully (e.g., shows an error message, doesn't crash).

Expected Result: API should return an appropriate error message (e.g., "Product not found") and the system should not crash.
Actual Result: Error message shown: "Product not found". System did not crash.
Status: Passed
Severity Level: High
Remarks: Error handling is robust, no crashes.

5. Product Listing Page:

Test Case ID: `TC-005`

Test Case Description: Validate Product Listing Page
Test Steps:

1. Open the product listing page.
2. Scroll through the page and verify that products are listed correctly.
3. Check if the product image, name, and price are displayed for each item.

Expected Result: Products should be listed correctly with images, names, and prices.
Actual Result: All products are listed as expected with accurate details.
Status: Passed
Severity Level: Low
Remarks: No issues found.

6. Categories Filter:

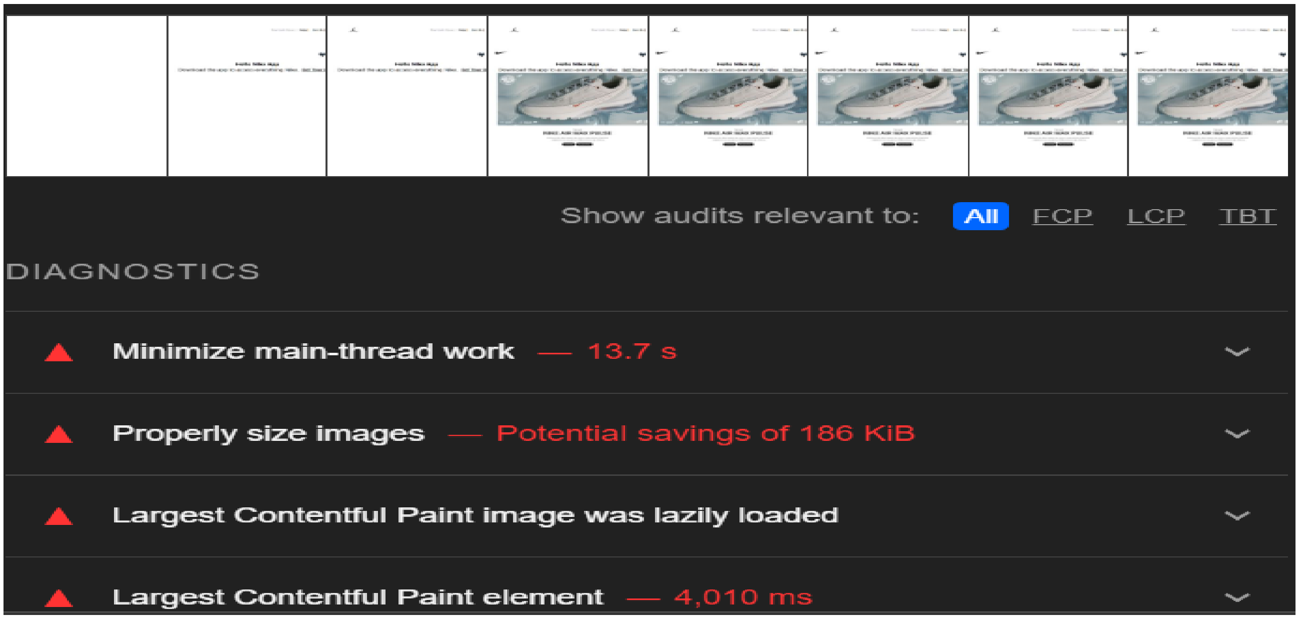
Test Case ID: `TC-006`

Test Case Description: Validate Categories Filter
Test Steps:

1. Open the product listing page.
2. Apply a category filter (e.g., "Electronics").
3. Verify that the products displayed belong to the selected category.

Expected Result: Only products from the "Electronics" category should be displayed.
Actual Result: Category filter works and displays only electronics.
Status: Passed
Severity Level: Medium
Remarks: Filter functionality works as expected.

Performance Optimization and Insights:



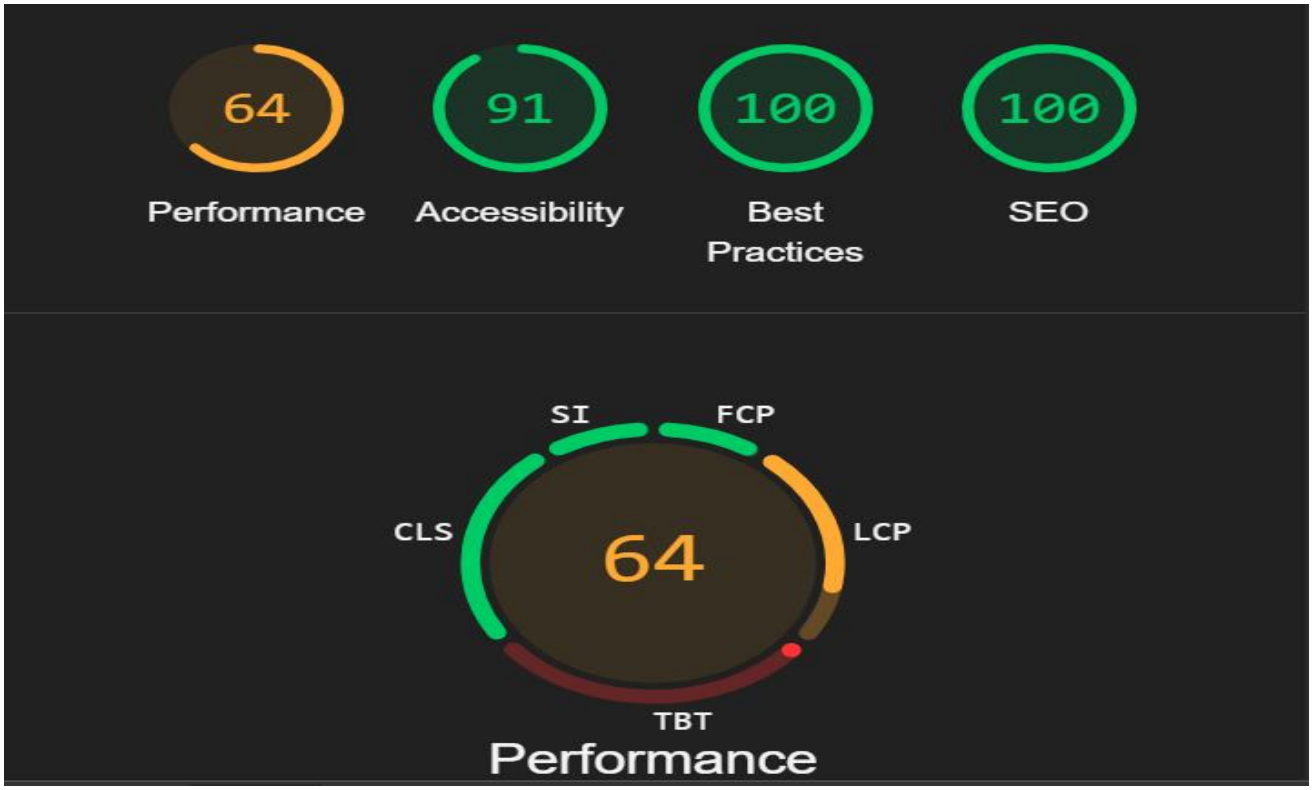
DIAGNOSTICS

▲ Minimize main-thread work — 13.7 s

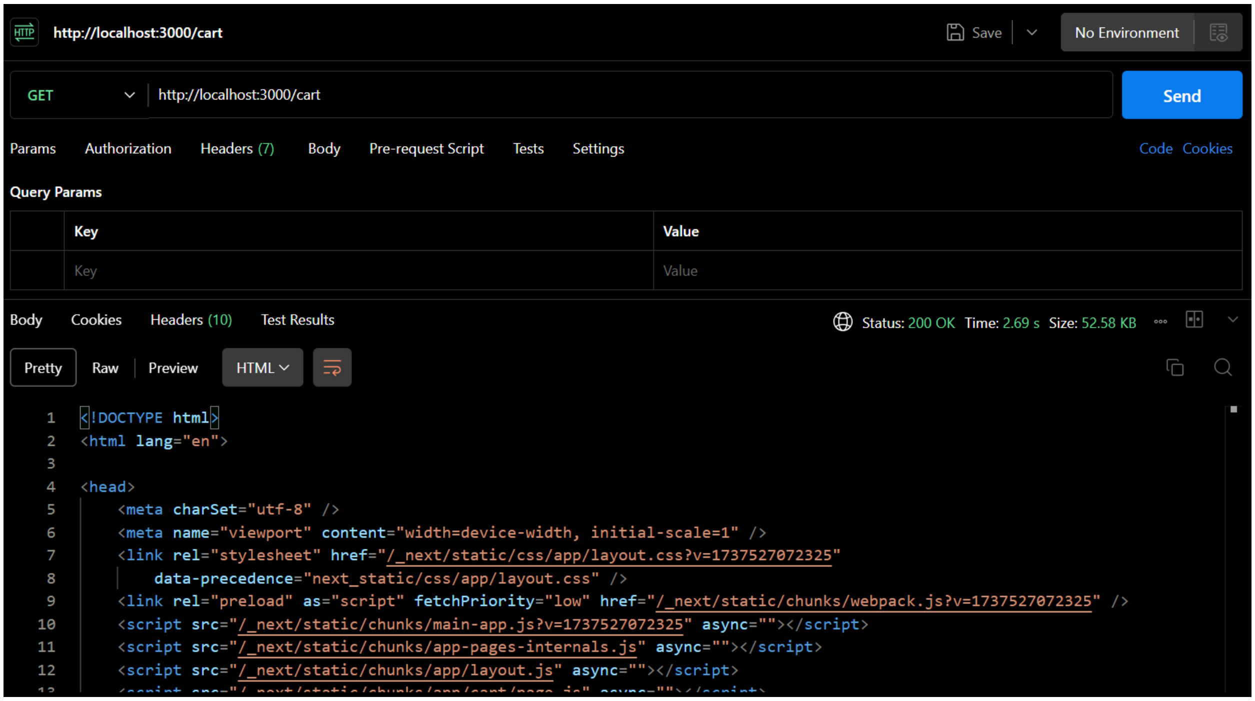
▲ Properly size images — Potential savings of 186 KiB

▲ Largest Contentful Paint image was lazily loaded

▲ Largest Contentful Paint element — 4,010 ms



API Response Validation and Results:



GET http://localhost:3000/cart

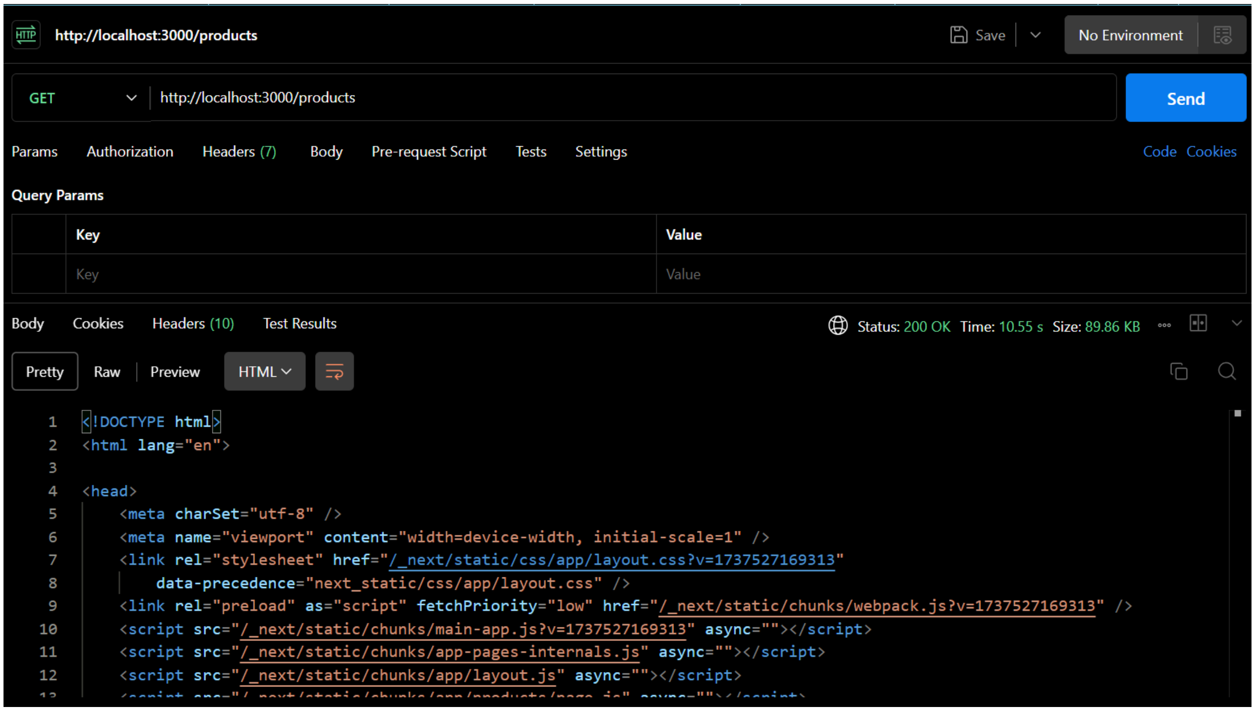
Query Params

Key	Value
Key	Value

Status: 200 OK Time: 2.69 s Size: 52.58 KB

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charSet="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="/_next/static/css/app/layout.css?v=1737527072325"
        data-precedence="next_static/css/app/layout.css" />
    <link rel="preload" as="script" fetchPriority="low" href="/_next/static/chunks/webpack.js?v=1737527072325" />
    <script src="/_next/static/chunks/main-app.js?v=1737527072325" async=""></script>
    <script src="/_next/static/chunks/app-pages-internals.js" async=""></script>
    <script src="/_next/static/chunks/app/layout.js" async=""></script>
    <script src="/_next/static/chunks/app/cart/page.js" async=""></script>
```



GET http://localhost:3000/products

Query Params

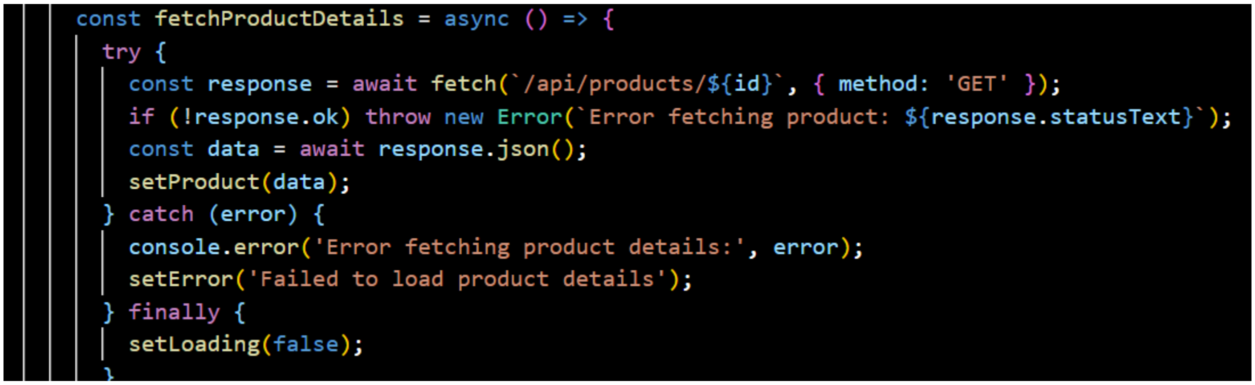
Key	Value
Key	Value

Status: 200 OK Time: 10.55 s Size: 89.86 KB

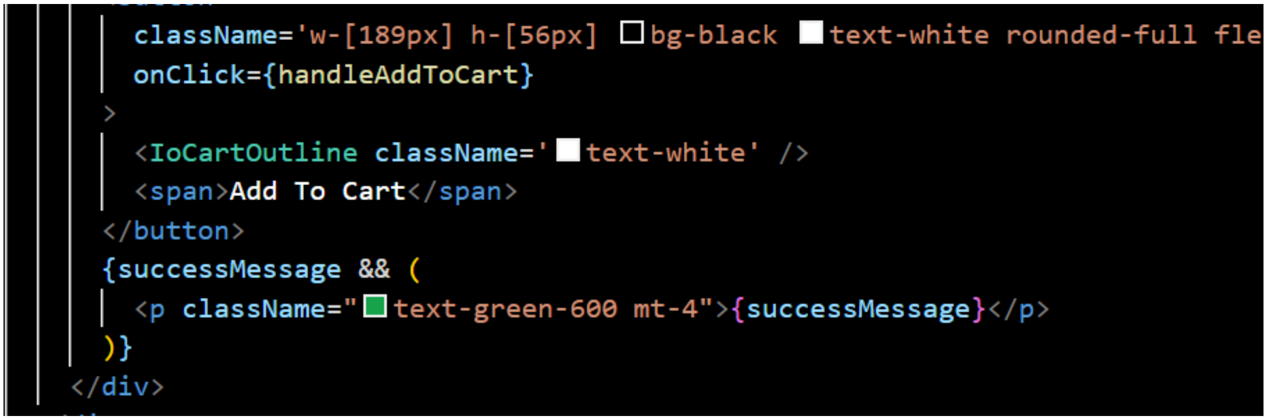
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charSet="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="/_next/static/css/app/layout.css?v=1737527169313"
        data-precedence="next_static/css/app/layout.css" />
    <link rel="preload" as="script" fetchPriority="low" href="/_next/static/chunks/webpack.js?v=1737527169313" />
    <script src="/_next/static/chunks/main-app.js?v=1737527169313" async=""></script>
    <script src="/_next/static/chunks/app-pages-internals.js" async=""></script>
    <script src="/_next/static/chunks/app/layout.js" async=""></script>
    <script src="/_next/static/chunks/app/products/page.js" async=""></script>
```

API Error Handling and Response Messages:

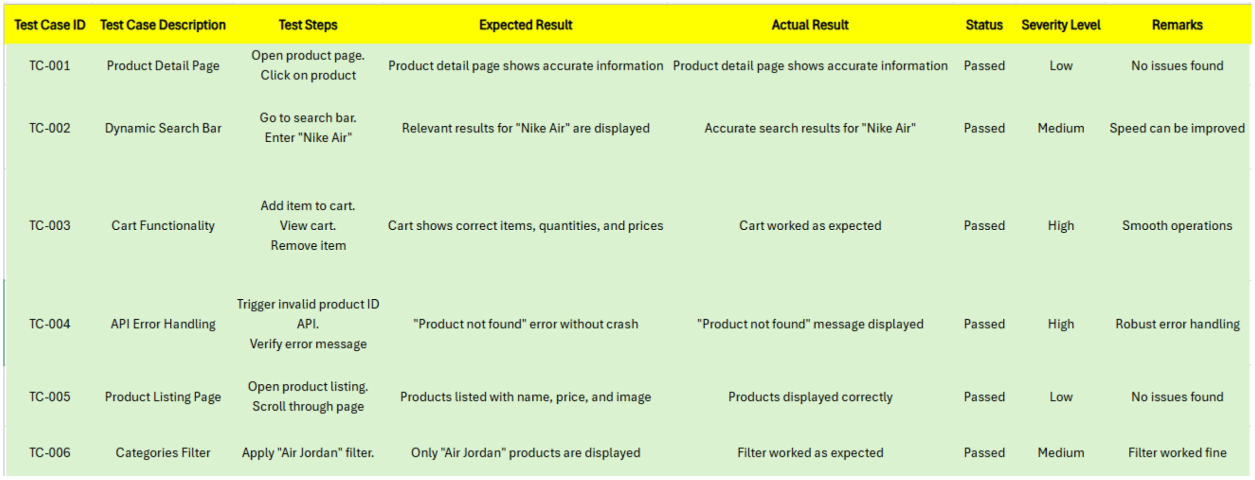


```
const fetchProductDetails = async () => {
  try {
    const response = await fetch(`/api/products/${id}`, { method: 'GET' });
    if (!response.ok) throw new Error(`Error fetching product: ${response.statusText}`);
    const data = await response.json();
    setProduct(data);
  } catch (error) {
    console.error('Error fetching product details:', error);
    setError('Failed to load product details');
  } finally {
    setLoading(false);
  }
```



```
    className='w-[189px] h-[56px] ⬛bg-black ⬜text-white rounded-full fle
    onClick={handleAddToCart}
  >
    <IoCartOutline className='⬜text-white' />
    <span>Add To Cart</span>
  </button>
  {successMessage && (
    <p className="🟩text-green-600 mt-4">{successMessage}</p>
  )}
</div>
```

CVS Functional Report:



Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC-001	Product Detail Page	Open product page. Click on product	Product detail page shows accurate information	Product detail page shows accurate information	Passed	Low	No issues found
TC-002	Dynamic Search Bar	Go to search bar. Enter "Nike Air"	Relevant results for "Nike Air" are displayed	Accurate search results for "Nike Air"	Passed	Medium	Speed can be improved
TC-003	Cart Functionality	Add item to cart. View cart. Remove item	Cart shows correct items, quantities, and prices	Cart worked as expected	Passed	High	Smooth operations
TC-004	API Error Handling	Trigger invalid product ID API. Verify error message	"Product not found" error without crash	"Product not found" message displayed	Passed	High	Robust error handling
TC-005	Product Listing Page	Open product listing. Scroll through page	Products listed with name, price, and image	Products displayed correctly	Passed	Low	No issues found
TC-006	Categories Filter	Apply "Air Jordan" filter.	Only "Air Jordan" products are displayed	Filter worked as expected	Passed	Medium	Filter worked fine

Security Measures:

- **Secure API Endpoints**: Implemented HTTPS and JWT for authentication.
- **CSRF Protection**: Applied anti-CSRF tokens for state-changing requests.
- **Access Control**: Used role-based access to ensure proper user permissions.

Challenges and Resolutions:

- **Slow Search Results**: Optimized search algorithm and added debouncing.
- **Cart Item Removal Issue**: Fixed session handling to sync cart state correctly.
- **Mobile Compatibility**: Adjusted responsive design for better mobile layout.
- **Payment Gateway Issues**: Fixed API key configuration errors for successful payments.