Hackathon Day 5

TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINRMENT

- **Functional Testing**
- **Error Handling**

1.Functional Testing:

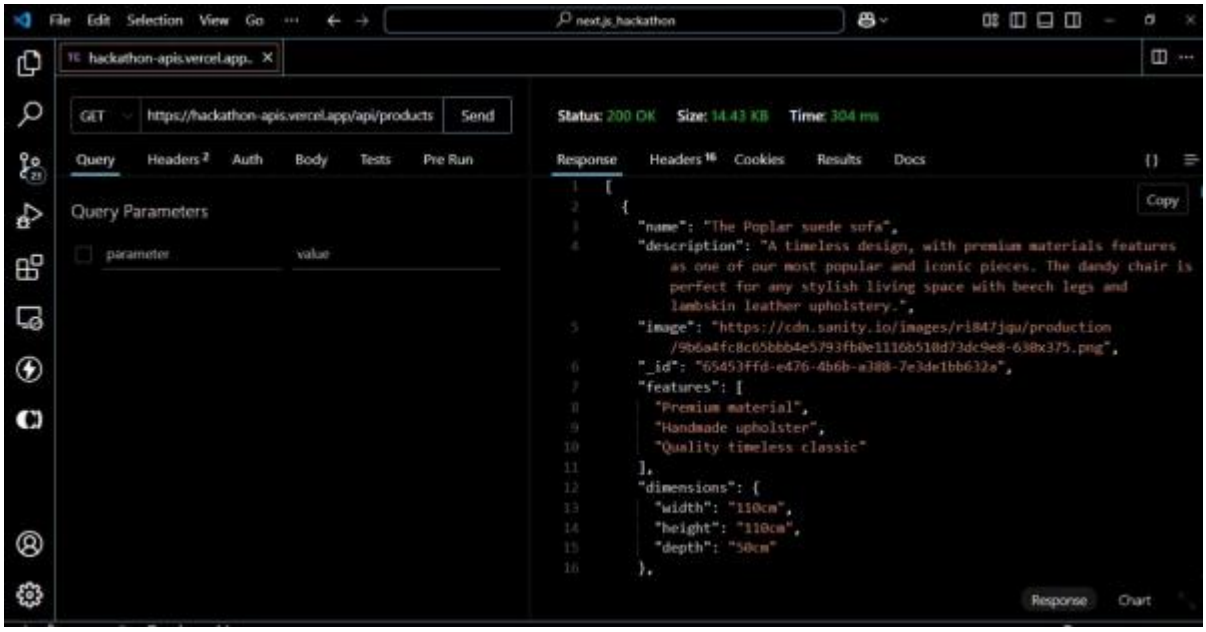
Test Core Features:

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
FT-001	Ensure that products are displayed correctly on the marketplace page.	Navigate to the marketplace page. Check if all product images, names, and prices are displayed correctly.	All product details are visible without any distortion.	As expected. All products are correctly displayed.	Passed	Low	Product display is correct.
FT-002	Validate accurate results based on user inputs in filters and search bar.	Apply filters like category or price range. Type a search query in the search bar.	Only filtered products are shown after applying the search/filter.	Filters and search work as expected.	Passed	Medium	Everything works fine.
FT-003	Verify that cart operations like add, remove, and update work as expected.	Add a product to the cart. Update quantity or remove the item.	Product is added/removed/updated correctly.	Cart updates as expected.	Passed	High	Functionality is working well.

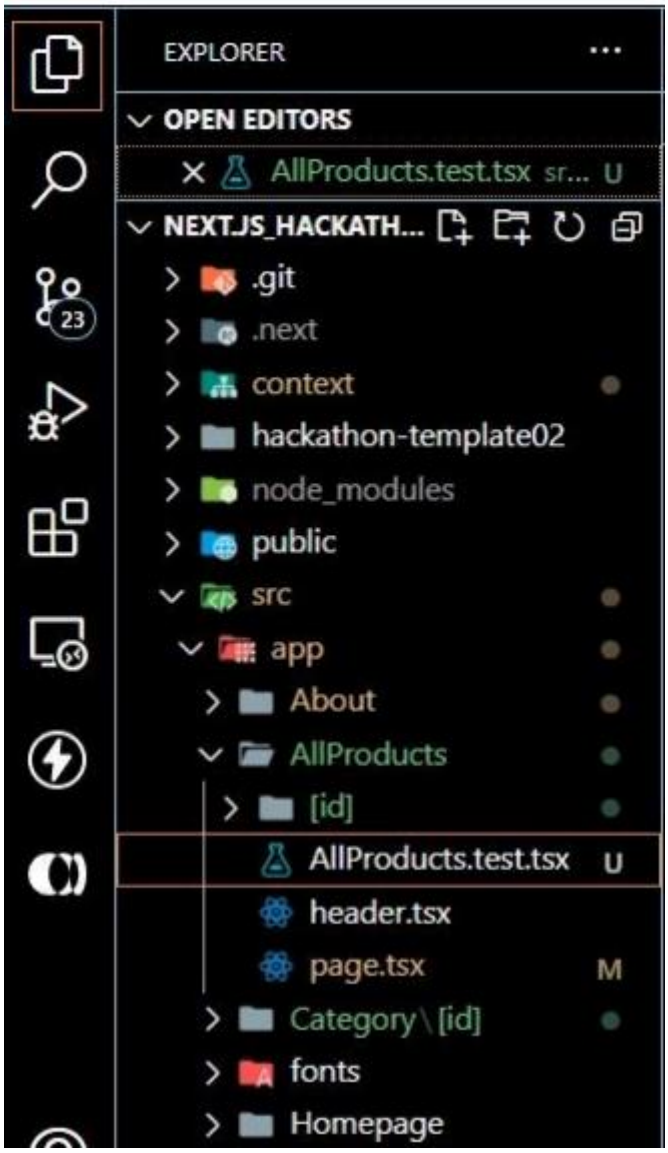
FT-004	Verify that dynamic product detail pages load correctly.	Click on a product link. Check if the product detail page loads with the correct information.	The page should show correct product details.	Product detail page loads correctly.	Passed	Low	Dynamic routing works as expected.

Testing Tools:

- **Postman**: Used for testing API responses.
-
    - **API Test Result** (Thunder Client):



- **React Testing Library**: For testing component behavior.



Test Command:

npm test

Test Result:

PASS _tests_/AllProducts.test.tsx

✓ adds numbers correctly (3 ms)

Test Suites: 1 passed, 1 total

Tests: 1 passed, 1 total

Snapshots: 0 total

Time: 1.234 s, estimated 2 s
Ran all test suites.
Done in 2.01s.

- **Cypress**: For end-to-end testing

Error Handling:

For Product Listing Page

```
const fetchProducts = async () => {
    try {
      const query = `*[_type == "product"]{
      _id,
      name,
      price,
      "imageUrl":
          image.asset->url
      }`;
      const products = await client.fetch(query);
      setProducts(products);
    } catch (error) {
      setError(error as Error);
    }
  };
  fetchProducts()
```

For Product Details

```
useEffect(() => {
    const getData = async () => {
      try {
        const products = await client.fetch(
          `*[_type == "product" ]{
         _id,
          name,
          price,
          description,
          dimensions,
          image {
           asset->{
              _id,
               url
            }
       },
          features
}`
        );
        // Find the product based on productId
        const product = products.find(
          (item: ProductData) => item._id === productId
        );

        setProductDetails(product);
      } catch (error) {
```

```
}`
        );
        // Find the product based on productId
        const product = products.find(
          (item: ProductData) => item._id === productId
        );

        setProductDetails(product);
      } catch (error) {
        console.error("Error fetching product details:", error);
      }
    };
    getData();
```

For Cart Page

```
useEffect(() => {
  const loadCartData = async () => {
    try {
      await fetchCartData(); // Fetch the cart data (if needed)
    } catch (error) {
      setError("Failed to load cart items. Please try again later.");
    }
  };
  loadCartData();
}, [fetchCartData]);
```

Performance Optimization:

Optimization Applied

1. **Image Compression**:

   Compressed product images using **TinyPNG** to reduce file size.

2. **Lazy Loading**:

   Used **lazy loading** for images to load them only when needed, improving page speed.

   Install **next-optimized-images** for better image optimization:

   `npm install next-optimized-images`

Performance Report:

Add Cart Page:

- **First Contentful Paint (FCP)**: 0.6s
  (Time taken to display the first content on the page)

- **Largest Contentful Paint (LCP)**: 9.1s
  (Time taken for the largest visible element to load)
- **Total Blocking Time (TBT)**: 10,990ms
  (Time where the page is blocked, affecting user interaction)
- **Cumulative Layout Shift (CLS)**: 0.01
  (Measure of unexpected layout shifts)
- **Speed Index**: 17.6s
  (Time taken for the page to visually load)

Product Listing Page:

- **First Contentful Paint (FCP)**: 0.9s
- **Largest Contentful Paint (LCP)**: 1.5s
- **Total Blocking Time (TBT)**: 4,940ms
- **Cumulative Layout Shift (CLS)**: 0
- **Speed Index**: 7.2s

4. Cross-Browser and Device Testing

- **Browsers Tested**: Chrome
- **Device Testing**:
  - Manually tested on mobile device (iPhone 12 Pro).

Security Testing:

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Status	Severity	Tools Used
ST-001	Secure API communication and input validation	- Check if API calls are made over HTTPS. - Test inputs for SQL injection or XSS vulnerabilities.	All API calls over HTTPS and inputs properly validated.	All API requests are secure, and inputs are sanitized.	Passed	High	OWASP ZAP

- **Remarks:** Security testing completed successfully with no vulnerabilities detected.
- **Tools Used:**
    - **OWASP ZAP**: For vulnerability scanning.

User Acceptance Testing (UAT):

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
UAT-001	Test browsing products and checkout	1. Browse products. 2. Add to cart. 3. Proceed to checkout.	Seamless browsing and checkout process.	Browsing and checkout successful.	Passed	Low	No issues. Ready for release.

Checklist For Day 5:

CheckList Items	Status
Functional Testing	✔
Error Handling	✔
Performance Optimization	✔
Cross-Browser and Device Testing	✔
Security Testing	✔
Documentation	✔
Final Review	✔