DAY 5

NAME: ALI HASAN

ROLL NO: 0044644

TIMING: SAT 9-12 AM

Day 5 - Testing, Error Handling, and Backend Integration Refinement

Steps for Implementation:

Here's the transformed version from a tester's perspective, providing feedback on the **Sanity CMS & Next.js** project:

Testing Report for Hecto:

Step 1: Functional Testing

1. Testing Core Features:

- **Product Listing:** The products are displayed correctly, but I noticed some images take a bit longer to load. This could be improved with image optimization. ✔
- **Filters and Search:** The filters work well, returning relevant results. However, the search functionality could be more responsive when handling multiple keywords. ✔
- **Cart Operations:** Adding, updating, and removing items from the cart works seamlessly. No issues encountered. ✔
- **Dynamic Routing:** Individual product detail pages load correctly, but there's a slight delay when fetching data from Sanity CMS. ✔

2. Testing Tools Used:

- **Postman:** API responses were tested, and all endpoints returned expected results.
- **React Testing Library:** Component testing was performed, and most elements behaved as expected.
- **Cypress:** Basic end-to-end tests confirmed smooth user interactions.

General Opinion: The core features function properly, but performance can be improved, especially in dynamic data retrieval.

Step 2: Error Handling

1. Error Messages:

- Tried simulating API failures. The try-catch blocks captured errors correctly, and appropriate fallback messages were displayed.

- Example error: "Failed to fetch products." This was logged properly in the console.

2. Fallback UI:

- When no products are available, a **"No items found"** message appears as expected.
- However, it would be better to add a **"Try Again"** button to improve user experience.

General Opinion: Error handling is well-implemented, but adding more user-friendly recovery options would enhance usability.

Step 3: Performance Optimization

1. Asset Optimization:

- Large images were slowing down initial load times. Compressed some using **TinyPNG**, which significantly improved performance.
- Implemented lazy loading for images, and it helped reduce unnecessary data loading.

2. Performance Analysis:

- Ran **Lighthouse tests**, and the **performance score was around 80**.
- Issues detected:
    - Unused CSS detected, which slightly affects rendering speed.
    - Some third-party scripts impact performance.

3. Load Time Testing:

- The homepage initially took **3+ seconds** to load but dropped below **2 seconds** after optimizations.

General Opinion: Performance is decent but can be further improved by refining asset loading and reducing unnecessary scripts.

Step 4: Cross-Browser and Device Testing

1. Browser Testing:

- Tested on **Chrome, Firefox, Safari, and Edge**. No major UI issues found, but Safari displayed slight inconsistencies in button styles.

2. Device Testing:

- Used **BrowserStack** for mobile simulation. The layout was responsive, but some text elements needed better scaling on smaller screens.
- Tested on a **physical Android device**, and the checkout process worked without issues.

General Opinion: The site is well-optimized for different browsers and devices, but minor UI tweaks could enhance mobile responsiveness.

Step 5: Security Testing

1. Input Validation:

- Tried entering malicious scripts in the search bar. The system properly sanitized inputs, preventing XSS attacks.
- Tested invalid email formats in forms, and error messages were displayed correctly.

2. Secure API Communication:

- All API requests were made over **HTTPS**, which is good.
- Checked environment variables, and API keys were **not exposed in the frontend**, confirming secure handling.

3. Security Tools Used:

- Ran an **OWASP ZAP scan**, and no critical vulnerabilities were detected.
- Used **Burp Suite** to test API security, and everything seemed secure.

General Opinion: Security measures are solid, but periodic testing should be done to prevent vulnerabilities.

Step 6: User Acceptance Testing (UAT)

1. Simulating Real-World Usage:

- Browsed products, added them to the cart, and completed a purchase. The overall flow was smooth.
- Some users might find the checkout process slightly long, so simplifying it would improve user experience.

2. Feedback Collection:

- Shared the platform with peers for feedback.
- Common suggestions:
    - Improve mobile navigation.
    - Add more product filtering options.

General Opinion: The platform is user-friendly, but refining the checkout process and adding more filter options would enhance usability.

Testing Results Summary:

- Issues found: Slow image loading, minor UI inconsistencies, and a slightly lengthy checkout process.
- Fixes applied: Image compression, lazy loading, and minor UI adjustments.

Final Checklist for Day 5:

✔ **Functional Testing:** ✅
✔ **Error Handling:** ✅
✔ **Performance Optimization:** ✅
✔ **Cross-Browser and Device Testing:** ✅
✔ **Security Testing:** ✅
✔ **Documentation:** ✅
✔ **Final Review:** ✅

Final Opinion:

Overall, There were some performance improvements needed, particularly in image loading and reducing unused scripts. Mobile responsiveness is good but could be refined for smaller screens. The checkout process works well but could be streamlined for better UX. Security measures are in place, and no major vulnerabilities were found.

With these refinements, the project will provide a **seamless and efficient shopping experience** for users.