

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

**E-Commerce Website**

**Product: Pudding & Jelly Powders**

**Name: Umehani**

**Roll No: 00364678**

**Day/Time: Friday – 09:00 – 12:00**

**Teacher: Sir Hamzah Syed**

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

**Steps for Implementation:**

**Step 1: Functional Testing**

**1. Test Core Features:**

* Product listing: Ensure products are displayed correctly.
* Filters and search: Validate accurate results based on user inputs.
* Cart operations: Add, update, and remove items from the cart.
* Dynamic routing: Verify individual product detail pages load correctly.

**2. Testing Tools:**

* Postman: For API response testing.
* React Testing Library: For component behavior testing.
* Cypress: For end-to-end testing.

**3. How to Perform Functional Testing:**

* Write test cases for each feature.
* Simulate user actions like clicking, form submissions, and navigation.
* Validate the output against expected results.

**Step 2: Error Handling**

**1. Add Error Messages:**

* Use try-catch blocks to handle API errors.

Example:

try {

const data = await fetchProducts();

setProducts(data);

} catch (error) {

console.error("Failed to fetch products:", error);

setError("Unable to load products. Please try again later.");

}

**2. Fallback UI:**

* Display alternative content when data is unavailable.
* Example: "No items found" message for an empty product list.

**Step 3: Performance Optimization**

**1. Optimize Assets:**

* Compress images using tools like TinyPNG or ImageOptim.
* Use lazy loading for large images or assets.

**2. Analyze Performance:**

* Use Lighthouse to identify speed and performance issues.
* Implement fixes such as reducing unused CSS, enabling browser caching, and optimizing JavaScript bundles.

**3. Test Load Times:**

* Measure initial load and interaction times.
* Aim for an initial page load time under 2 seconds.

**Step 4: Cross-Browser and Device Testing**

**1. Browser Testing:**

* Test on Chrome, Firefox, Safari, and Edge.
* Verify consistent rendering and functionality.

**2. Device Testing:**

* Use responsive design tools like BrowserStack to simulate diƯerent devices.
* Manually test on at least one physical mobile device.

**Step 5: Security Testing**

1. **Input Validation**

* Sanitize inputs to prevent SQL injection or XSS attacks.
* Use regular expressions to validate email, phone, and other inputs.

**2. Secure API Communication:**

* Ensure API calls are made over HTTPS.
* Store sensitive data like API keys in environment variables.

**3. Testing Tools:**

* OWASP ZAP: For automated vulnerability scanning.
* Burp Suite: For advanced penetration testing.

**Step 6: User Acceptance Testing (UAT)**

**1. Simulate Real-World Usage:**

* Perform tasks like browsing products, adding items to the cart, and checking out.
* Identify and fix any usability issues.

**2. Feedback Collection:**

* Ask peers or mentors to test your marketplace and provide feedback.

**Step 7: Documentation Updates**

**1. Include Testing Results:**

* Summarize key issues found and how they were resolved.
* Provide before-and-after screenshots for fixes.

**2. Submission Format:**

* Submit documentation in PDF or Markdown format.
* Include test case details, testing tools used, and optimization steps.

**Checklist for Day 5:**

* Functional Testing:

✔ ✘

* Error Handling:

✔ ✘

* Performance Optimization:

✔ ✘

* Cross-Browser and Device Testing:

✔ ✘

* Security Testing:

✔ ✘

* Documentation:

✔

* Final Review:

✔ ✘