**Day-5**

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

## **1. Overview**

On Day 5, we focused on refining our marketplace project by performing comprehensive testing, implementing robust error handling, and optimizing backend integration. The goal was to ensure a fully functional, efficient, and secure application.

## **2. Objectives**

* Conduct **functional testing** to validate core features.
* Implement **error handling** for API failures and unexpected issues.
* Optimize the **performance** of frontend and backend components.
* Ensure **cross-browser and device compatibility**.
* Perform **security testing** to safeguard user data.
* Compile a **CSV-based test report** documenting test cases and results.

## **3. Functional Testing**

### 3.1 Test Cases

| **Test Case ID** | **Description** | **Steps** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC-001 | Verify product listing loads correctly | Open homepage → Check product list | Products should display | Products display correctly | ✅ |
| TC-002 | Ensure product search works | Enter a search term → Click search | Correct products display | Expected products appear | ✅ |
| TC-003 | Check cart functionality | Add product → Open cart | Product appears in cart | Cart updates correctly | ✅ |
| TC-004 | Test checkout process | Proceed to checkout | Order confirmation screen | Checkout successful | ✅ |
| TC-005 | Validate error handling on API failure | Disconnect API → Reload | Error message displayed | Error handled properly | ✅ |

### 3.2 Tools Used

* **Postman** for API testing
* **React Testing Library** for component testing
* **Cypress** for end-to-end UI testing
* **Lighthouse** for performance testing

## **4. Error Handling Implementation**

### 4.1 API Error Handling

Implemented error handling for API failures:

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.");

}

### 4.2 Fallback UI Implementation

* Display user-friendly error messages.
* Provide retry options where applicable.

## **5. Performance Optimization**

### 5.1 Optimization Techniques

* **Lazy loading** for images and assets.
* **Minified CSS and JavaScript** to reduce load time.
* **Optimized API calls** with pagination to limit payload size.

### 5.2 Lighthouse Performance Report

* **Performance Score:** 92/100
* **Key Fixes:** Improved image compression, enabled caching, reduced unused JavaScript.

## **6. Cross-Browser & Device Testing**

### 6.1 Browser Compatibility

| **Browser** | **Compatibility** |
| --- | --- |
| Chrome | ✅ |
| Firefox | ✅ |
| Safari | ✅ |
| Edge | ✅ |

### 6.2 Responsive Testing

Tested across various screen sizes:

* **Mobile (375px, 768px)** - Fully responsive ✅
* **Tablet (1024px)** - Fully responsive ✅
* **Desktop (1440px, 1920px)** - Fully responsive ✅

## **7. Security Testing**

### 7.1 Key Security Measures

* **Input validation** to prevent SQL Injection & XSS.
* **HTTPS enforcement** for secure API communication.
* **Environment variables** used to store sensitive data.

## **8. User Acceptance Testing (UAT)**

Conducted UAT by simulating real user interactions:

* **Browsing & searching:** Smooth navigation and accurate search results.
* **Cart & checkout:** Seamless experience with clear order confirmation.
* **Error handling:** Displayed appropriate messages on failures.

## **9. Documentation & Submission**

### 9.1 Final Submission Includes:

* **CSV-based Test Report** with test cases, results, and remarks.
* **Updated Project Code** with error handling and optimizations.
* **Lighthouse Performance Report** showcasing improvements.
* **GitHub Repository Link** with structured README.

### 9.2 GitHub Repository Structure

📂 marketplace-project

┣ 📂 src

┃ ┣ 📂 components

┃ ┣ 📂 pages

┃ ┗ 📂 utils

┣ 📂 tests

┣ 📜 README.md

┗ 📜 test-report.csv

## **10. Conclusion**

By the end of Day 5, the marketplace project was thoroughly tested, optimized for performance, and secured against vulnerabilities. The final deliverables were submitted with well-documented testing reports and optimizations.