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

Objectives:

- Conduct rigorous testing on core features.
- Enhance error handling mechanisms.
- Optimize performance for smooth user experience.
- Ensure cross-browser and cross-device compatibility.
- Document all testing results and updates.
- Testing Tool Used "**Lighthouse**".

Step-by-Step Implementation

Step 1: Functional Testing

1. Test Core Features:

- **Product Listing:** Ensure products display dynamically with correct details.
- **Filters and Search:** Validate accuracy of filters and search results.
- **Cart Operations:** Verify adding, updating, and removing items functionality.
- **Product Detail Pages:** Confirm correct loading of dynamic product detail pages.

2. Testing Tools:

- **Postman:** For backend API testing.
- **React Testing Library:** For component behaviour validation.
- **Cypress:** For end-to-end testing of user flows.

3. Execution Process:

- Write comprehensive test cases.
- Simulate user actions via Cypress.
- Compare actual outcomes with expected results.

Step 2: Error Handling

- **Error Example Code:**

javascript

CopyEdit

```
try {
   const response = await fetch('/api/products');
```

```
    const data = await response.json();
    setProducts(data);
} catch (error) {
    console.error('Failed to fetch products:', error);
    setError('Unable to load products. Please try again later.');
}
```

- **Fallback UI Examples:**
    - Show "No products available" when the list is empty.
    - Display "No products match your search criteria" for invalid search results.

Step 3: Performance Optimization

Key Actions:

- Compress images (e.g., TinyPNG).
- Use **Google Lighthouse** to detect and fix issues (e.g., unused CSS, unoptimized assets).
- Aim for initial page load time under **2 seconds** using tools like **GTmetrix**.

Step 4: Cross-Browser and Device Testing

- **Browser Testing:** Validate compatibility on Chrome.
- **Device Testing:** Use **BrowserStack** for simulations and manual testing on mobile devices.

Step 5: Security Testing

Focus Areas:

- **Input Validation:** Prevent SQL Injection and XSS.
- **Secure API Communication:** Ensure HTTPS and protect sensitive API keys.

Step 6: User Acceptance Testing (UAT)

- Simulate real-world user actions (e.g., adding products to cart, checking out).
- Collect feedback from peers and incorporate improvements.

Step 7: Documentation Updates

- Include test cases, results, and resolutions

- Use Markdown or PDF formats with before-and-after screenshots.

Testing Results (CSV Report)

Test Case	Status	Expected Result	Actual Result	Resolution
Product Listing Test	☑ Pass	Products displayed correctly	Products displayed correctly	N/A
Search Functionality Test	✕ Fail	Results filtered correctly	No results for 'laptop'	Fix search query handling
Cart Operations Test	☑ Pass	Add, update, and remove items from the cart	Cart works as expected	N/A

Key Deliverables

- **Functional Testing:** Core components thoroughly tested.
- **Error Handling:** Implemented robust fallback mechanisms.
- **Performance Optimization:** Enhanced load times and responsiveness.
- **Cross-Compatibility:** Verified consistency across platforms.
- **Comprehensive Test Report:** Documented in CSV and Markdown.