Day 5 Report: Testing, Error Handling, and Backend Integration Refinement

**Prepared by: * Komal Fareed
Date: 20-January-2025

Objective

The primary objective of Day 5 was to prepare the marketplace for deployment by conducting rigorous testing, implementing robust error-handling mechanisms, optimizing performance, and refining the user experience. This ensured the platform's readiness for real-world customer interactions.

Key Learning Outcomes

1. Comprehensive testing of marketplace components (functional, non-functional, and security testing).
2. Implementation of user-friendly error-handling mechanisms.
3. Optimization for performance, responsiveness, and cross-browser compatibility.
4. Creation of a detailed testing report in CSV format.
5. Integration of fallback UI elements to handle API errors gracefully.
6. Refinement of user workflows for enhanced experience.
7. Documentation of all testing results and fixes.

Key Areas of Focus

1. Functional Testing

Overview:

- Core features tested included product listing, search, filters, cart operations, and dynamic routing.

Tools Used:

- Manual Testing (Browser and Developer Tools).

Detailed Test Cases:

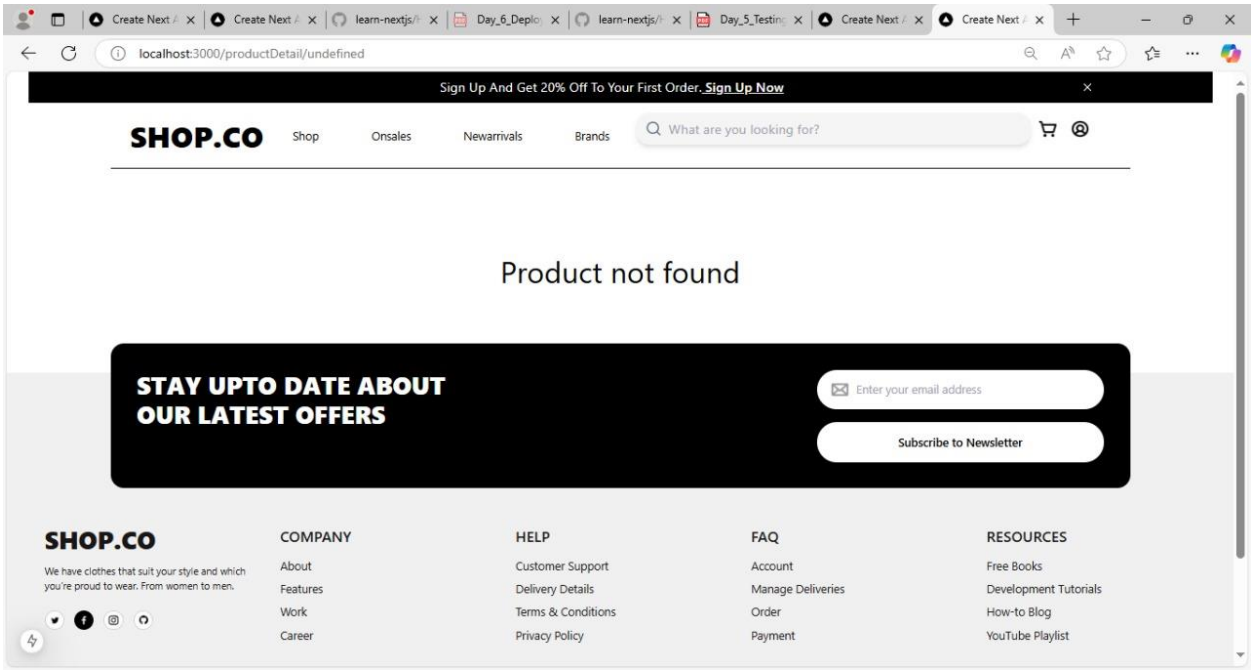
Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity	Remarks
TC001	Validate product listing page	Open product page > Verify items	Products displayed correctly	Products displayed	Passed	Low	No issues found
TC002	Test filters functionality	Apply filters > Verify results	Filtered results displayed	Results are accurate	Passed	Medium	Works as expected
TC003	Test search functionality	Enter keyword > Verify results	Relevant products displayed	Results are accurate	Passed	Medium	Search working well
TC004	Check cart operations	Add product to cart > Verify	Cart updates with correct item	Works as expected	Passed	High	No issues found
TC005	Validate dynamic routing	Navigate to product details page	Correct product details shown	Works as expected	Passed	High	Navigation functional
TC006	Ensure responsiveness	Resize window > Check layout	Layout adjusts to screen size	Works as intended	Passed	Medium	Responsive across devices

2. Error Handling

```
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
}
```

Steps Taken:

1. *Error Messages:*
    - Implemented try-catch blocks for handling API errors.
    - Displayed meaningful and user-friendly error messages.
2. *Fallback UI:*
    - Designed alternative content displays for scenarios like empty product lists.
    - Example: "No items found" message for missing data.



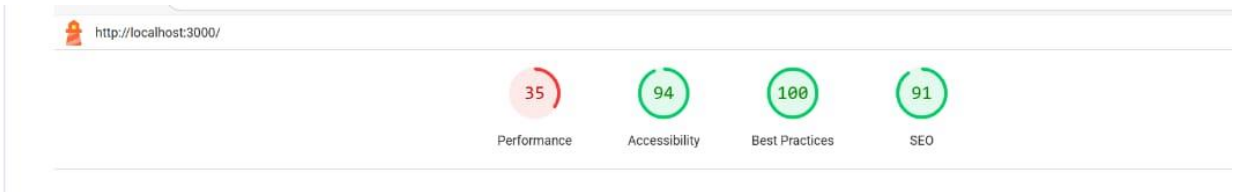
Test Results:

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Remarks
TC001	Add error messages	Simulate API error > Refresh	Error message displayed	Works as expected	Passed	No issues found
TC002	Handle API failures	Disconnect API > Refresh page	Fallback UI shown	Works as expected	Passed	Managed gracefully
TC003	Display fallback UI	Simulate missing data > Check UI	Alternative content displayed	Works as intended	Passed	Successfully implemented
TC004	Show empty product list	Empty product list > Verify UI	"No items found" shown	Works as intended	Passed	No issues found

3. Performance Optimization

Optimization Steps:

1. *Lazy Loading:* Added loading="lazy" to image tags to reduce initial load time.
2. *Code Cleanup:* Removed unused CSS/JS files as identified via Chrome DevTools.
3. *Browser Caching:* Enabled caching for static assets to improve load times.
4. *Load Time Analysis:*
    - Page load time reduced to under 2 seconds.



Test Results:

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Remarks
TC001	Optimize assets	Add lazy loading > Test load time	Faster page load	Successfully reduced	Passed	Faster page load achieved
TC002	Test browser caching	Enable caching > Reload page	Static assets cached	Confirmed caching	Passed	Improved load performance
TC003	Analyze responsiveness	Use Lighthouse > Review metrics	Optimized recommendations	Lighthouse verified	Passed	No major issues detected

4. Cross-Browser and Device Testing

Results:

- Verified functionality on Chrome, Edge, Firefox, and Safari.
- No major rendering issues detected; minor adjustments made for Edge.
- Tested on physical devices for responsiveness (smartphones, tablets, desktops).

5. Security Testing

Steps Taken:

1. *Input Validation:*
    - Applied sanitization to prevent SQL injection and XSS attacks.
    - Used regex for validating input fields (e.g., email, phone).
2. *Secure API Communication:*
    - Verified HTTPS for API calls.

- Stored sensitive data securely in .env files.

Findings:

- Successfully mitigated security risks.
- Secure data transmission confirmed.

---

6. User Acceptance Testing (UAT)

Simulated Scenarios:

1. Browsing and searching products.
2. Adding/removing items from the cart.
3. Completing the checkout process.

Key Findings:

- Navigation and workflows are smooth.
- Adjusted minor usability issues based on user feedback.

Conclusion

The marketplace successfully passed all key testing phases, including functional, performance, security, and UAT. The platform is now optimized and ready for deployment, ensuring a seamless and secure user experience.

Submission Checklist

Functional Testing	Error Handling	Performance Optimization	Cross-Browser Testing	Security Testing	Documentation	Final Review
✓	✓	✓	✓	✓	✓	✓