|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** | **Severity Level** | **Assigned To** | **Remarks** |
| 2 | TC001 | Validate product listing page | Open product page > Verify products | Products displayed correctly | Products displayed correctly | Passed | high | - | No issues found |
| 3 | TC002 | Test API error handling | Disconnect API > Refresh page | Show fallback UI with error message | Error message shown | Passed | Medium | - | No Issues found |
| 4 | TC003 | Check cart functionality | Add product to cart > Verify cart contents | Cart updates with added product | Cart updates as expected | Passed | High | - | Works as expected |
| 5 | TC004 | Ensure responsiveness on mobile | Resize browser window > Check layout | Layout adjusts properly to screen size | Responsive layout working as intended | Passed | Medium | - | Test successful |

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

**Refinement**

**Objective:**

Day 5 focuses on preparing your marketplace for real-world deployment by ensuring all

components are thoroughly tested, optimized for performance, and ready to handle

customer-facing traƯic. The emphasis will be on testing backend integrations,

implementing error handling, and refining the user experience.

Key Learning Outcomes:

1. Perform comprehensive testing, including functional, non-functional, user

acceptance, and security testing.

2. Implement robust error handling mechanisms with clear, user-friendly fallback

messages.

3. Optimize the marketplace for speed, responsiveness, and performance metrics.

4. Ensure cross-browser compatibility and device responsiveness for a seamless

user experience.

5. Develop and submit professional testing documentation that meets industry

standards, including a CSV-based test report.

6. Handle API errors gracefully with fallback UI elements and logs.

7. Optimize the marketplace for speed and responsiveness.

8. Ensure cross-browser and device compatibility.

9. Prepare detailed documentation for testing results and resolutions.

Key Areas of Focus:

1. Functional Testing

 Validate that all marketplace features work as intended.

 Test core functionalities like product listing, detail pages, cart operations, and

user profile management.

2. Error Handling

 Implement proper error messages for:

o Network failures.

o Invalid or missing data.

o Unexpected server errors.

 Display fallback UI elements (e.g., "No products available" when the API returns

no data).

3. Performance Testing

 Identify bottlenecks using tools like Lighthouse, GT metrix, We b PageTest or

google page speed.

 Optimize images, minimize JavaScript and CSS, and implement caching

strategies.

4. Cross-Browser and Device Testing

 Test your marketplace on popular browsers (Chrome, Firefox, Safari, Edge) and

devices (desktop, tablet, mobile).

 Use responsive design testing tools like Browser Stack or Lambda Test.

5. Security Testing

 Validate input fields to prevent injection attacks.

 Use HTTPS for secure communication.

 Avoid exposing sensitive API keys in your frontend code.

6. User Acceptance Testing (UAT)

 Simulate real-world scenarios by interacting with your marketplace as a user.

 Verify that workflows like browsing, searching, and checkout are intuitive and

error-free.

7. Documentation Updates

 Document testing results, fixes, and best practices followed.

 Include detailed test reports summarizing key findings and resolutions.

 Ensure test reports are formatted professionally and follow market standards.

 Maintain a consistent format for headings and subheadings.

 Include a table of contents for easy navigation.

Steps for Implementation:

Step 1: Functional Testing

1. Test Core Features:

o Product listing: Ensure products are displayed correctly.

o Filters and search: Validate accurate results based on user inputs.

o Cart operations: Add, update, and remove items from the cart.

o Dynamic routing: Verify individual product detail pages load correctly.

2. Testing Tools:

o Postman: For API response testing.

o React Testing Library: For component behavior testing.

o Cypress: For end-to-end testing.

3. How to Perform Functional Testing:

o Write test cases for each feature.

o Simulate user actions like clicking, form submissions, and navigation.

o Validate the output against expected results.

2. Fallback UI:

o Display alternative content when data is unavailable.

o Example: "No items found" message for an empty product list.

Step 3: Performance Optimization

1. Optimize Assets:

o Compress images using tools like Tiny PNG or Image Optim.

o Use lazy loading for large images or assets.

2. Analyze Performance:

o Use Lighthouse to identify speed and performance issues.

o Implement fixes such as reducing unused CSS, enabling browser

caching, and optimizing JavaScript bundles.

3. Test Load Times:

o Measure initial load and interaction times.

o Aim for an initial page load time under 2 seconds.

Step 4: Cross-Browser and Device Testing

1. Browser Testing:

o Test on Chrome, Firefox, Safari, and Edge.

o Verify consistent rendering and functionality.

2. Device Testing:

o Use responsive design tools like Browser Stack to simulate di Ưerent

devices.

o Manually test on at least one physical mobile device.

Step 5: Security Testing

1. Input Validation

o Sanitize inputs to prevent SQL injection or XSS attacks.

o Use regular expressions to validate email, phone, and other inputs.

2. Secure API Communication:

o Ensure API calls are made over HTTPS.

o Store sensitive data like API keys in environment variables.

3. Testing Tools:

o OWASP ZAP: For automated vulnerability scanning.

o Burp Suite: For advanced penetration testing.

Step 6: User Acceptance Testing (UAT)

1. Simulate Real-World Usage:

o Perform tasks like browsing products, adding items to the cart, and

checking out.

o Identify and fix any usability issues.

2. Feedback Collection:

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

Step 7: Documentation Updates

1. Include Testing Results:

o Summarize key issues found and how they were resolved.

o Provide before-and-after screenshots for fixes.

2. Submission Format:

o Submit documentation in PDF or Markdown format.

o Include test case details, testing tools used, and optimization steps.

Expected Output:

By the end of Day 5, students should have:

1. Fully tested and functional marketplace components, validated against

professional testing standards.

2. Clear and user-friendly error handling mechanisms implemented across all

functionalities.

3. Optimized performance with faster load times and smoother interactions.

4. A responsive design tested thoroughly on multiple browsers and devices.

5. A comprehensive CSV-based testing report documenting test cases, results, and

resolutions.

6. Well-structured documentation summarizing all testing and optimization eƯorts.

7. Error handling mechanisms with clear messages and fallback UI.

8. Optimized performance for faster page load times.

9. A responsive design verified on multiple browsers and devices.

10.Comprehensive documentation of testing and fixes.

Submission Requirements:

Document Title: "Day 5 - Testing and Backend Refinement - [Your Marketplace Name]"

What to Submit:

1. Functional Deliverables:

 Screenshots or recordings showcasing functional and responsive components.

 Logs or reports from testing tools (e.g., Lighthouse, Postman).

2. Testing Report (CSV Format):

 Submit a detailed testing report in CSV format, including the following columns:

o Test Case ID: A unique identifier for each test case.

o Test Case Description: A concise explanation of what is being tested.

o Test Steps: Step-by-step procedure to execute the test case.

o Expected Result: The anticipated outcome of the test case.

o Actual Result: The observed outcome of the test case.

o Status: Mark as "Passed," "Failed," or "Skipped."

o Severity Level: Categorize issues as High, Medium, or Low based on

impact.

o Assigned To: Name of the person responsible for fixing issues (if

applicable).

o Remarks: Additional notes or comments for clarity.