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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** | **Severity Level** | **Assigned To** | **Remarks** |
|  |  |  |  |  |  |  |  |  |
| **TC00 1** | **Validate product listing** | 1. Open the product listing page. 2. Verify that all products are displayed correctly. | Products are displayed with correct images, names, and prices. | Items shown accurately. | **Passed** | Low | **\_** | "No errors identified." |
| **TC00 2** | **Validate filters functionality** | 1. Apply a filter on the product page (e.g., price range). 2. Verify the displayed results. | Products matching the selected filter criteria are displayed accurately. | Filters functioning as intended. | **Passed** | medium | **\_** | "Working without issues." |
| **TC00 3** | **Test search functionality** | 1. Enter a keyword in the search bar. 2. Press "Search" and review the results | Products matching the keyword appear in the results list | Results shown accurately. | **Passed** | medium | **\_** | "Runs as expected." |
| **TC00 4** | **Test API response for product data** | 1. Test API endpoints using Postman. 2. Verify the response matches the expected schema. | API returns correct product data, including ID, name, and price. | API responses are correct | **Passed** | high | **\_** | "The API is working without discrepancies." |
| **TC00 5** | **Test dynamic routing for products** | 1. Click on a product from the listing page. 2. Verify the product detail page loads. | The product detail page loads with correct details (name, description, price). | Pages load as anticipated. | **Passed** | high | **\_** | "Dynamic routing operates smoothly." |
| **TC00 6** | **Validate error handling for API failures** | 1.Simulate API failure (e.g., disconnect API)  2.Trigger API call 3. Check error handling logic | Error message should be displayed gracefully | Error message shown as expected. | **Passed** | high | **\_** | "Error handling performs correctly." |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TC00 7** | **1. Simulate empty API response 2. Check UI for fallback message** | "No items found" message should be shown | "No items found" message should be shown | "No products found" message displayed. | **Passed** | | medium | **\_** | "Fallback UI has been properly implemented." |
| **TC00 8** | **Validate image compression and lazy loading** | . Compress images using Tiny PNG/ImageSoft 2. Implement lazy loading for large images | mages should load faster, and page performance should improve | Images compressed; lazy loading enabled. | **Passed** | | Medium | **\_** | "Page load time improved by 30%." |
| **TC00 9** | **Validate cross-browser compatibility** | 1. Test on Chrome, Firefox, Safari, and Edge 2. Verify consistent rendering and functionality | Consistent rendering and functionality across all browsers | Uniform behavior across different browsers. | **Passed** | | High | **\_** | "No problems with rendering or performance." |
| **TC00 10** | **Validate responsiveness on different devices** | 1. Use BrowserStack to simulate devices (e.g., iPhone, iPad, Android) 2. Manually test on at least one physical mobile device | Responsive design should work seamlessly on all devices | Responsive design functions as intended. | **Passed** | | High | **\_** | "No issues detected with layout or functionality." |
| **TC00 11** | **Validate input sanitization and validation** | 1. Test input fields for SQL injection and XSS attacks 2. Use regular expressions to validate email, phone, etc. | inputs should be sanitized and validated, preventing malicious inputs | Inputs cleaned and validated successfully. | **Passed** | | high | **\_** | "The system is free of vulnerabilities." |
| **TC00 12** | **Validate secure API communication** | 1. Check if API calls are made over HTTPS 2. Verify sensitive data (e.g., API keys) is stored in environment variables | API calls should use HTTPS, and sensitive data should be secure | API requests must use HTTPS, ensuring sensitive data is secure. | | **Passed** | Critical | **\_** | "Secure communication protocols are in place." |
| **TC00 13** | **Validate wishlist functionality** | 1.Browse products  2.Add items to the wishlist.  3. View wishlist. | All tasks should be completed without errors | Tasks finished successfully. | | **Passed** | High | **\_** | "No issues with user experience." |
| **TC00 14** | **Collect feedback from peers/mentors** | 1. Share the marketplace with peers/mentor s 2. Collect feedback on usability and functionality | Feedback should highlight any usability or functionality issues | Feedback gathered and examined. | | **Passed** | medium | **\_** | "Useful feedback provided for further improvements." |

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