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

Documentation:

Testing and Optimization Report

1. Test Cases Executed and Their Results:

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
TC001	Home Page Load Test	1. Open the website in a browser 2. Ensure the homepage loads fully	Homepage should load in under 3 seconds	Homepage loaded in 2.5 seconds	Passed	Low	-	No issue found
TC002	Product Listing Page	1. Navigate to the product page 2. Verify product images and details load correctly	Products should be displayed with images & details	Images loaded in 2.32 seconds	Passed	Medium	-	Handled gracefully
TC003	Product Search Functionality	1. Enter product name in search bar 2. Click search	Relevant products should be displayed	Correct products displayed	Passed	Low	-	Handled gracefully
TC004	Add to Cart Functionality	1. Select a product 2. Click 'Add to Cart' button	Product should be added to the cart	Product added successfully	Passed	Low	-	No issue found
TC005	Responsive Design Test	1. Open website on mobile/tablet 2. Check layout responsiveness	Layout should adjust correctly	Mobile layout successfully	Passed	Medium	-	Worked as expected
TC006	Test API error handling	Disconnect API > refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	-	Handled gracefully

2. Performance Optimization Steps Taken:

- **Code Splitting:** Implemented dynamic imports to reduce initial page load times.
- **Image Optimization:** Utilized Next.js `next/image` for lazy loading and responsive images.
- **Server-side Rendering (SSR):** Used SSR for SEO-critical pages to improve performance.
- **Caching Strategies:** Implemented caching for API responses to enhance page speed.
- **Minification and Compression:** Enabled automatic minification of JS and CSS files.
- **Database Query Optimization:** Reduced query execution time by adding proper indexes.

3. Security Measures Implemented:

- **Data Encryption:** Sensitive data encrypted using industry-standard encryption methods.
- **CORS Policy:** Configured to prevent unauthorized cross-origin requests.
- **Input Validation:** Prevented XSS and SQL injection via strict input validation.
- **Rate Limiting:** Applied to prevent abuse of login and API endpoints.
- **Regular Security Audits:** Conducted vulnerability scanning using automated tools.

4. Challenges Faced and Resolutions Applied:

Challenge	**Resolution**
Slow load time due to large images	Implemented image optimization using Next.js image component
Mobile responsiveness issues	Applied flexible grid layouts and media queries
State management inconsistencies	Introduced Redux for better global state management
Cross-browser compatibility issues	Conducted testing on multiple browsers and fixed UI inconsistencies

---