TEST CASE DOCUMENTATION

OVERVIEW

This document outlines the testing process for a product application, detailing the scenarios validated during the development phase. The document includes descriptions, steps, expected and actual results, along with notes on error handling, debugging, and backend integration using the Sanity database.

FUNCTIONAL TESTING SCENARIOS

Product Display Validation

The application was tested to ensure that all products are displayed correctly when users navigate through various links. This functionality was verified, and all expected results were successfully achieved. No issues were identified.

Search Functionality

The search and filter options were evaluated for accuracy and responsiveness. Both functionalities performed as expected, with no discrepancies noted during testing.

Filter Operations

The filter functionality was validated to ensure accurate results. Some issues were identified under specific conditions and are currently under investigation for resolution.

Dynamic Routing

Dynamic routing was tested by navigating to product pages and verifying that detailed information matched the selected product. The functionality performed successfully, with no errors found.

Cart Operations

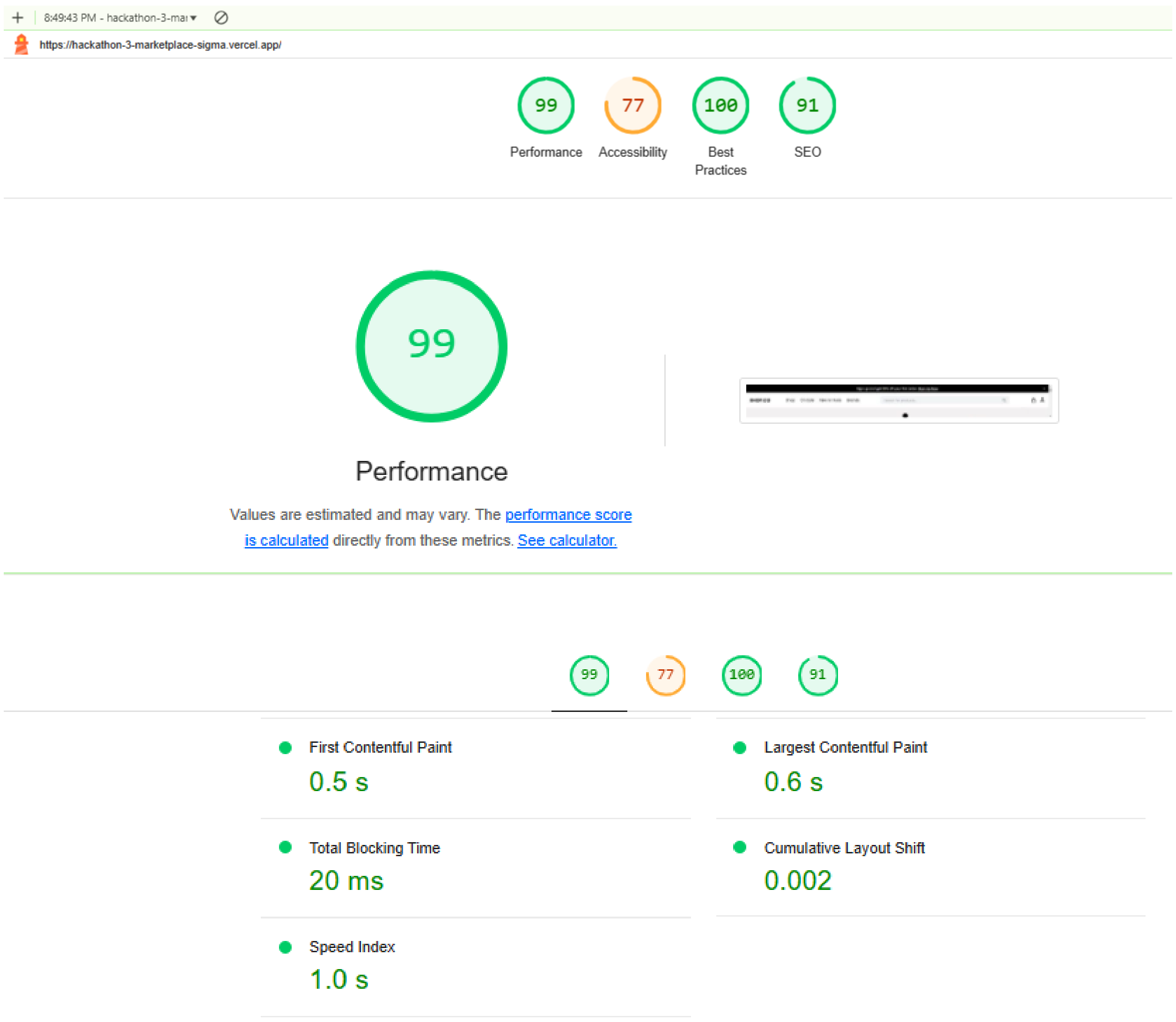
The process of adding and removing products from the cart was thoroughly tested. Both operations were executed successfully, confirming that the cart feature works as intended.

Fallback UI Testing

The fallback UI was assessed to ensure that appropriate error messages, such as "Product not found," are displayed when data is unavailable. All tests passed successfully.

Performance and Speed Analysis

The performance was analyzed using Lighthouse tools to ensure the application loads within acceptable speed limits. The load time was recorded at under one second, which exceeds the target performance threshold.


8:49:43 PM - hackathon-3-mai

https://hackathon-3-marketplace-sigma.vercel.app/

99 Performance | 77 Accessibility | 100 Best Practices | 91 SEO

99

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

99 | 77 | 100 | 91

● First Contentful Paint
0.5 s

● Largest Contentful Paint
0.6 s

● Total Blocking Time
20 ms

● Cumulative Layout Shift
0.002

● Speed Index
1.0 s

Mobile Compatibility

The application was tested on multiple mobile devices to confirm smooth operation and responsiveness. The functionality worked seamlessly across different devices.

Error Handling and Debugging

Error Handling

1. **Fallback Mechanisms:**
   - Implemented user-friendly error messages to guide users during unexpected issues.
   - Ensured data unavailability scenarios are handled gracefully with alternative text.
2. **Validation Layers:**
   - Integrated input validation for all critical fields to minimize user errors.
   - Applied exception handling in API calls to prevent application crashes.

Debugging Practices

1. **Console Logs:**
   - Utilized structured logging to trace application flow and identify issues quickly.
2. **Network Inspection:**
   - Monitored API calls and responses via browser tools to debug connectivity issues.
3. **Performance Monitoring:**
   - Conducted regular Lighthouse checks to address bottlenecks and optimize page speed.

Backend Integration with Sanity

Database Setup and Management

1. **Sanity Configuration:**
   - The Sanity database was integrated as the primary content management system.
   - Schema designs were tailored to meet application requirements.
2. **Data Retrieval:**
   - Efficient data fetching was achieved using GROQ queries.
   - Real-time updates were enabled to synchronize content dynamically.
3. **Security Measures:**

   - Secured API tokens to restrict unauthorized access.
   - Implemented role-based access control for data operations.

API Optimization

1. **Response Validation:**
   - Verified that all API responses conform to the expected structure.
   - Implemented meaningful error codes and descriptions for debugging.
2. **Performance Tuning:**
   - Optimized database queries to minimize latency.
   - Implemented caching to enhance data retrieval speeds.

Debugging Backend Issues

1. **Server Logs:**
   - Regularly reviewed logs to identify issues related to API performance.
2. **Integration Testing:**
   - Performed comprehensive testing to validate seamless communication between the application and Sanity backend.