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

Today's focus is on ensuring that our website provides a seamless experience for users by thoroughly testing its functionality. The key objective is to validate that the site performs optimally when accessed by users, which is crucial for any website. We will conduct tests to evaluate error handling, ensuring that every page loads correctly and consistently across different devices. Additionally, we will verify that the site is responsive and that data is properly fetched from our backend (Sanity CMS).

We will also simulate various user scenarios to identify potential errors that might arise in different situations. Performance optimization is another key task, where we aim to improve the site's loading speed and responsiveness. To ensure cross-browser compatibility, we'll test the website across different platforms, including Google Chrome, Microsoft Edge, Bing, and others.

Finally, we will compile a comprehensive documentation report detailing any errors encountered, their solutions, and strategies for improvement. This report will serve as a reference for ongoing development and ensure the website's stability and efficiency.

Now, let's break down each aspect. I have thoroughly examined my website from every angle to ensure its performance and functionality.

✓ Pages Rendering:

First, we will thoroughly test all the pages of our website, including Home, Contact, About, Shop, Login, Cart, and others, to ensure that they are loading correctly. This step is crucial to verify that each page displays as intended without any issues.

✓ Functional Testing:

In this step, we will ensure that each function is working correctly, such as product rendering, product details, the cart page, search functionality, and more. Let's begin with product rendering. As shown in the image below, we have demonstrated how we fetch data from Sanity and render it within a component. This component is then rendered on the Shop page, allowing all products to be displayed properly on the page.

Product Rendering:

- **Data Fetching from Sanity**:
    - client.fetch(...): This part of the code fetches product data (including _id, title, price, description, and productImage) from Sanity. This is crucial for ensuring that the data displayed on the page is fetched correctly.

- **Rendering Products**:
    - The fetched data is stored in the products state using setProducts(data). This allows dynamic rendering of products on the page.
    - The filteredProducts.map(...) loop iterates over the filtered products and renders them on the page.
    - For each product, the Image component renders the product image using urlFor(product.productImage)?.width(500).url().
    - The product title, truncated description, and price are displayed for each product, making sure all relevant details are visible to the user.

Component Rendering on Shop Page:

- **Shop Page Integration**:
    - In the second code (Page component), the ItemsPage component is imported and rendered within the page. This ensures that the products rendered by ItemsPage are properly displayed on the Shop page.
    - The ItemsPage component is placed directly in the layout, ensuring that all the products from Sanity are visible on the Shop page without any issues.

Search Functionality:

- Today I have also used Search functionality as I said in yesterday document. The search bar (<input type="text" />) allows users to search for products by title. The filteredProducts array dynamically updates based on the search term, which is crucial for product filtering and ensuring that only relevant products are displayed.

✓ Error Handling Scenarios:

1. **Network Failures:**
    - Display a message like "Failed to fetch products. Please try again later."
2. **Missing or Invalid Data:**
    - If no data is available, show the message "No products available at the moment."
3. **Unexpected Server Errors:**
    - Display a message indicating an issue and suggest trying again later.

- **Handling Product Fetching:**
    1. When the user clicks on "Shop" and a request is sent to the server:
        - Show the message: "Loading Products..." while fetching data.
    2. If no products are available:
        - Display: "No products available at the moment."
    3. If there's a network issue or server error:
        - Show: "Failed to fetch products. Please try again later."
- **Product Detail View:**
    1. If a user clicks on a product that is not found:
        - Trigger the notFound() function to display a "Product not found" message.

Above approach ensures clear communication to users about any issues with loading products or viewing product details.

✓ Performance Testing:

For performance testing, we will utilize Lighthouse, a pre-built tool provided by Google Chrome. This tool helps us assess the overall performance of the website by generating detailed reports on factors such as load speed, accessibility, best practices, and SEO.

We will conduct separate performance tests for both mobile and desktop environments to ensure the website performs optimally across different devices. This will allow us to identify any potential bottlenecks or areas for improvement in terms of speed, responsiveness, and user experience.

Desktop Report:

Lighthouse Performance Report

Captured on: Jan 20, 2025, 5:03 PM GMT+5
Emulated Device: Desktop
Lighthouse Version: 12.2.1
Tool: Lighthouse 12.2.1 | Generated with Chromium 131.0.0.0

Performance: 72

- **First Contentful Paint (FCP):** 0.4 s
- **Largest Contentful Paint (LCP):** 0.5 s
- **Total Blocking Time (TBT):** 980 ms
- **Cumulative Layout Shift (CLS):** 0
- **Speed Index (SI):** 0.7 s

Recommendations for Improvement:

1. **Reduce JavaScript Execution Time:**
   - Current savings potential: **2.2 s**
2. **Minimize Main-Thread Work:**
   - Current savings potential: **3.0 s**
3. **Optimize Images:**
   - Save **183 KiB** by serving images in next-gen formats (e.g., WebP).
4. **Minify JavaScript:**
   - Potential savings: **30 KiB**.
5. **Eliminate Render-Blocking Resources:**
   - Potential savings: **20 ms**.
6. **Defer Offscreen Images:**
   - Potential savings: **191 KiB**.
7. **Remove Duplicate Modules in JavaScript Bundles:**
   - Potential savings: **11 KiB**.

Page Load Issues:

1. **Avoid Long Main-Thread Tasks:**
   - 8 long tasks found.
2. **Ensure Proper Image Sizing:**
   - Potential savings: **32 KiB**.
3. **Minimize Network Payloads:**
   - Total size: **2,193 KiB**.

4. **Avoid Enormous Network Payloads:**
   - Review large JavaScript files and assets.

Accessibility: 82

Accessibility Issues:

1. **Contrast:**
   - Improve the contrast between background and foreground colors for better legibility.
2. **Names and Labels:**
   - Document lacks a element.
   - Links do not have discernible names for accessibility tools.
3. **Touch Targets:**
   - Improve touch target sizes and spacing for better user interaction.
4. **Navigation:**
   - Ensure headings are in a sequentially descending order for better keyboard navigation.

Best Practices: 100

- **CSP (Content Security Policy):** Ensure it's effective to protect against XSS attacks.
- **Missing Source Maps for Large JavaScript Files:** Ensure source maps are available for easier debugging.

SEO: 82

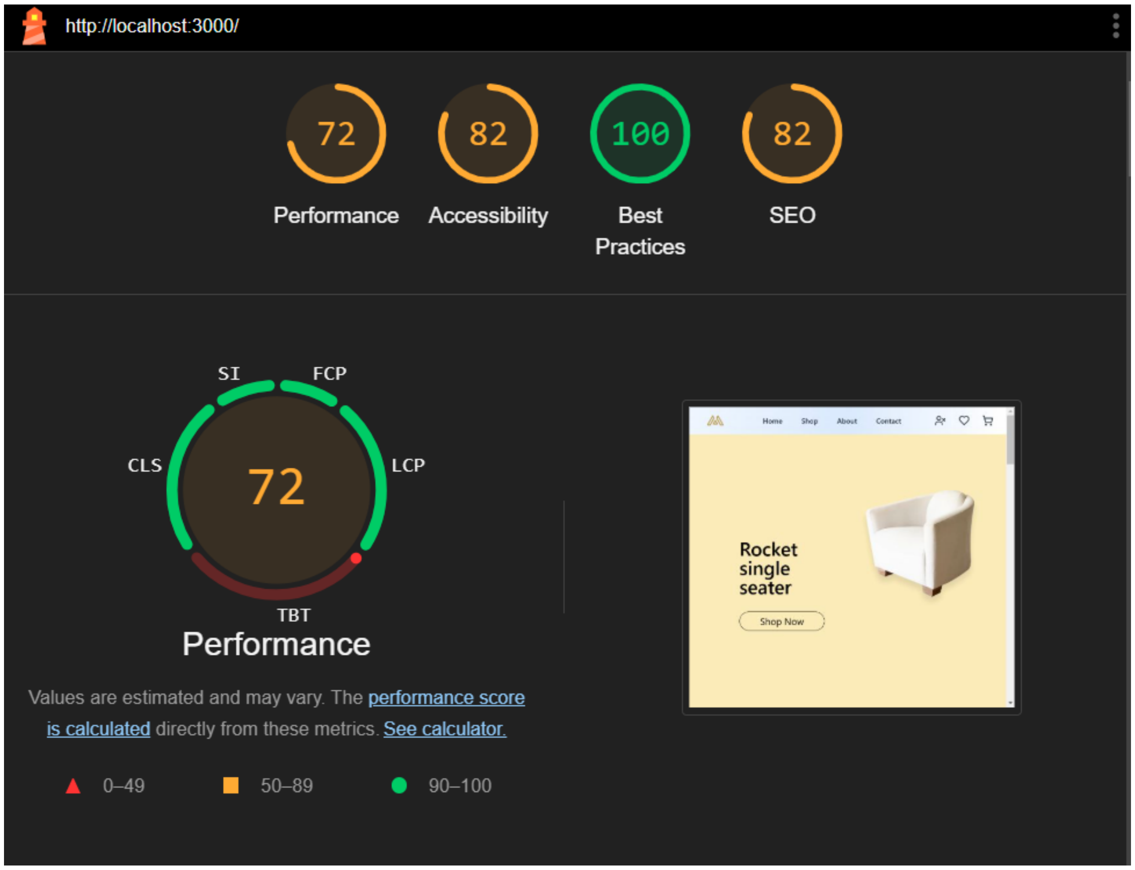
SEO Issues:

1. **Content Best Practices:**
   - Document lacks a element and a meta description, which is crucial for search engine optimization.
2. **Manual Checks:**

- ○ Format your HTML to make it more accessible for crawlers to understand.

Passed Audits:

- **Performance Audits Passed:** 17
- **Accessibility Audits Passed:** 19
- **Best Practices Audits Passed:** 13
- **SEO Audits Passed:** 6



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

Mobile performance:

Lighthouse Performance Report

Captured on: Jan 20, 2025, 5:13 PM GMT+5
Emulated Device: Moto G Power
Lighthouse Version: 12.2.1
Tool: Lighthouse 12.2.1 | Generated with Chromium 131.0.0.0

Performance: 69

- **First Contentful Paint (FCP):** 1.1 s
- **Largest Contentful Paint (LCP):** 2.0 s
- **Total Blocking Time (TBT):** 3,660 ms
- **Cumulative Layout Shift (CLS):** 0.022
- **Speed Index (SI):** 1.5 s

Recommendations for Improvement:

1. **Reduce JavaScript Execution Time:**
   - Current savings potential: **6.2 s**
2. **Minimize Main-Thread Work:**
   - Current savings potential: **8.3 s**
3. **Optimize Images:**
   - Save **59 KiB** by properly sizing images.
   - Save **183 KiB** by serving images in next-gen formats (e.g., WebP).
4. **Minify JavaScript:**
   - Potential savings: **30 KiB**.
5. **Eliminate Render-Blocking Resources:**
   - Potential savings: **130 ms**.
6. **Defer Offscreen Images:**
   - Potential savings: **191 KiB**.
7. **Remove Duplicate Modules in JavaScript Bundles:**
   - Potential savings: **11 KiB**.

Page Load Issues:

1. **Avoid Long Main-Thread Tasks:**
   - 14 long tasks found.
2. **Ensure Proper Image Sizing:**
   - Potential savings: **59 KiB**.

3. **Minimize Network Payloads:**
    - Total size: **2,280 KiB**.
4. **Avoid Large Layout Shifts:**
    - 1 layout shift found.
5. **Avoid Enormous Network Payloads:**
    - Review large JavaScript files and assets.

Accessibility: 86

Accessibility Issues:

1. **Contrast:**
    - Improve the contrast between background and foreground colors for better legibility.
2. **Names and Labels:**
    - Document lacks a element.
3. **Touch Targets:**
    - Improve touch target sizes and spacing for better user interaction.
4. **Navigation:**
    - Ensure headings are in a sequentially descending order for better keyboard navigation.

Best Practices: 100

- **CSP (Content Security Policy):** Ensure it's effective to protect against XSS attacks.
- **Missing Source Maps for Large JavaScript Files:** Ensure source maps are available for easier debugging.

SEO: 82

SEO Issues:

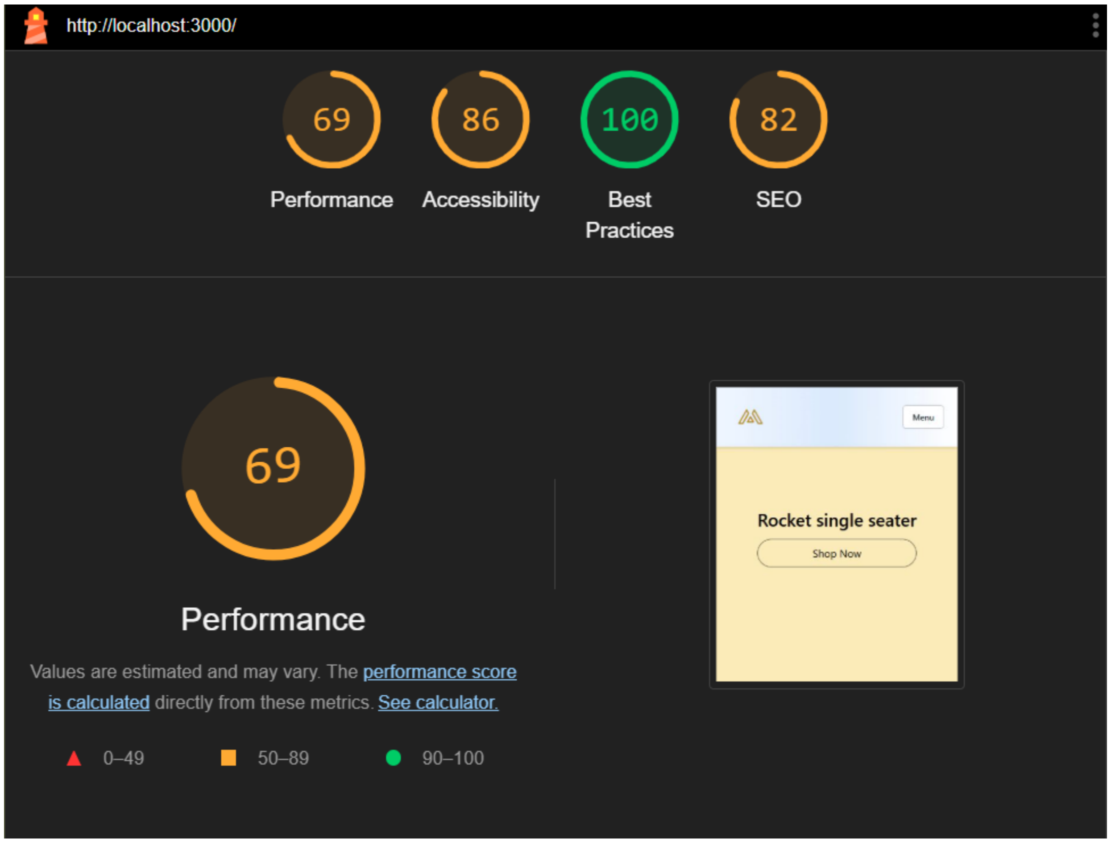
1. **Content Best Practices:**

    - Document lacks a element and a meta description, which is crucial for search engine optimization.

2. **Manual Checks:**
    - Format your HTML to make it more accessible for crawlers to understand.

Passed Audits:

- **Performance Audits Passed:** 17
- **Accessibility Audits Passed:** 20
- **Best Practices Audits Passed:** 14
- **SEO Audits Passed:** 6



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

Above is the performance report of the site.

✓ Cross-Browser and Device Testing:

I have tested my website thoroughly on Google Chrome and Microsoft Edge to ensure cross-browser compatibility. Additionally, I have evaluated its responsiveness across various devices, including desktops, laptops, tablets, and mobile phones. This comprehensive testing ensures that the website delivers a seamless and consistent user experience, regardless of the device or browser being used.

✓ Security Testing:

- Validate input fields to prevent injection attacks.
- Use HTTPS to ensure secure communication.
- Avoid exposing sensitive API keys or credentials in frontend code.

To enhance security, I have concealed critical credentials in the frontend, ensuring they are not visible to users or potential attackers. Furthermore, I have refrained from pushing these credentials to GitHub, maintaining strict confidentiality and safeguarding the project's integrity.

✓ User Acceptance Testing (UAT):

- Simulated real-world scenarios by interacting with the marketplace as a user.
- Verified that workflows such as browsing, searching, and navigation are intuitive and error-free.

To further validate the user experience, I shared the website with friends and family members to test it as real-world users. Their feedback was positive, and everything functioned smoothly. However, the checkout button is currently non-functional since the checkout process has not yet been implemented.