

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

Objective:

The main goal of **Day 5** is to ensure that all components are thoroughly tested, optimized for performance, and ready for real-world deployment. This includes testing backend integrations, implementing error handling, refining user experiences, and preparing documentation.

Key Learning Outcomes:

1. Perform comprehensive testing: functional, non-functional, user acceptance, and security testing.
 2. Implement robust error handling mechanisms with clear and 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, including a CSV-based test report.
 6. Handle API errors gracefully with fallback UI elements and logs.
-

Key Areas of Focus:

1. **Functional Testing:**
 - Validate marketplace features (e.g., product listing, cart, user profiles).
2. **Error Handling:**
 - Implement proper error messages and fallback UI for network failures, missing data, or unexpected server errors.
3. **Performance Testing:**
 - Identify bottlenecks, optimize images, minimize CSS/JS, and implement caching strategies.
4. **Cross-Browser and Device Testing:**
 - Ensure compatibility with popular browsers and devices.
5. **Security Testing:**
 - Validate inputs to prevent attacks and use HTTPS for secure communication.
6. **User Acceptance Testing (UAT):**

- Simulate real-world user interactions to ensure smooth workflows.
 - 7. **Documentation Updates:**
 - Document testing results, fixes, and best practices followed.
-

Steps for Implementation:

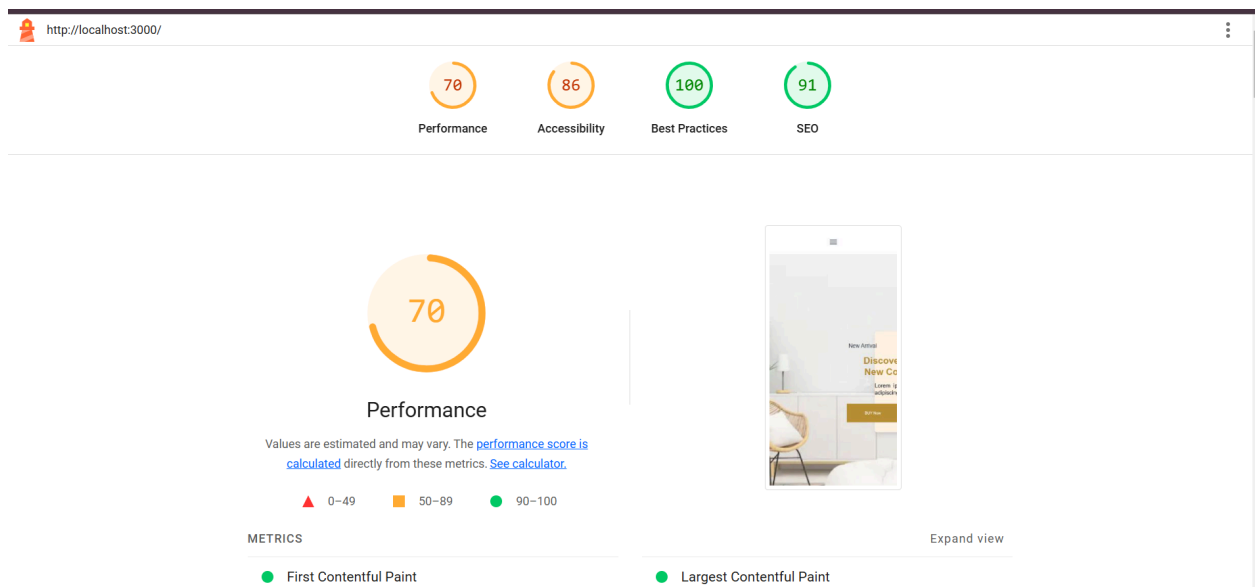
1. **Functional Testing:**
 - Test features like product listings, filters, cart operations, and routing.
 - Use tools like **Postman**, **React Testing Library**, and **Cypress** for testing.
2. **Error Handling:**
 - Add **try-catch** blocks and implement fallback UI for unavailable data.
3. **Performance Optimization:**
 - Use **Lighthouse** to analyze and improve performance.
4. **Cross-Browser Testing:**
 - Test across browsers (Chrome, Firefox, Safari, Edge) and devices.
5. **Security Testing:**
 - Ensure secure communication via **HTTPS** and input validation.
6. **User Acceptance Testing (UAT):**
 - Simulate real-world tasks (e.g., browsing, adding to cart) to identify usability issues.
7. **Documentation Updates:**
 - Provide detailed reports on testing, fixes, and challenges.

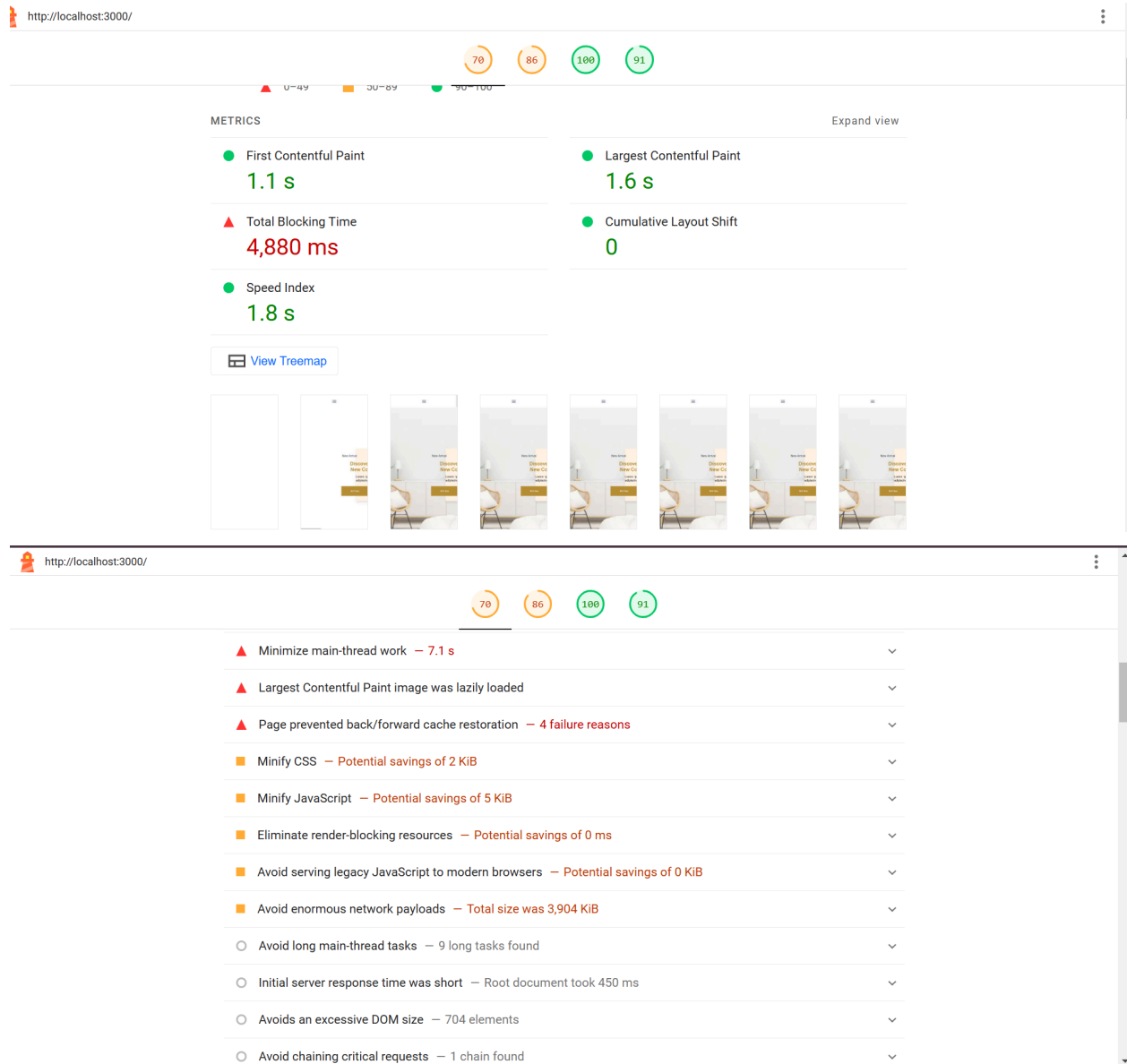
API Response Testing

The screenshot shows the Thunder Client interface. The top bar includes the application name 'THUNDER CLIENT' and several tabs: 'hero.tsx M', 'page.tsx ...\products U', 'page.tsx ...\id U', 'Extension: Thunder Client', 'TC Release Notes', and 'TC New Request X'. The left sidebar has a 'New Request' button and tabs for 'Activity', 'Collections', and 'Env'. The main area is divided into 'Query' and 'Response' sections. The 'Query' section shows a GET request to 'https://template6-six.vercel.app/api/products' with a 'Send' button. The 'Response' section shows the status '200 OK', size '45.98 KB', and time '791 ms'. The response body is a JSON object with the following structure:

```
1 [
2   {
3     "imageUrl": "https://cdn.sanity.io/images
4       /7xt4qcah/production
5       /2219cafc285ec13a2ed3f88aa36cbea852a11735
6       -305x375.png",
7     "price": 210,
8     "tags": [
9       "rustic ",
10      "vase ",
11      "home decor",
12      "vintage ",
13      "interior design"
14    ],
15     "discountPercentage": 10,
16     "description": "Bring the charm of nature into
17       your home with the Rustic Vase Set. Perfect
18       for those who appreciate timeless beauty and
19       a warm, inviting atmosphere, this set of
20       vases adds a touch of rustic elegance to any
21       space. Crafted with care and attention to
22       detail, these vases are designed to evoke
23       the essence of vintage craftsmanship while
24       seamlessly complementing both modern and
```

Performance Testing Results for Marketplace








Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

CONTRAST

-  Background and foreground colors do not have a sufficient contrast ratio.

These are opportunities to improve the legibility of your content.


NAMES AND LABELS

CONTRAST

-  Background and foreground colors do not have a sufficient contrast ratio.


These are opportunities to improve the legibility of your content.

NAMES AND LABELS

-  Links do not have a discernible name

These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

TABLES AND LISTS

-  List items (``) are not contained within ``, `` or `<menu>` parent elements.

These are opportunities to improve the experience of reading tabular or list data using assistive technology, like a screen reader.



Best Practices

TRUST AND SAFETY

- ☐ Ensure CSP is effective against XSS attacks ▼
- ☐ Use a strong HSTS policy ▼
- ☐ Ensure proper origin isolation with COOP ▼



SEO

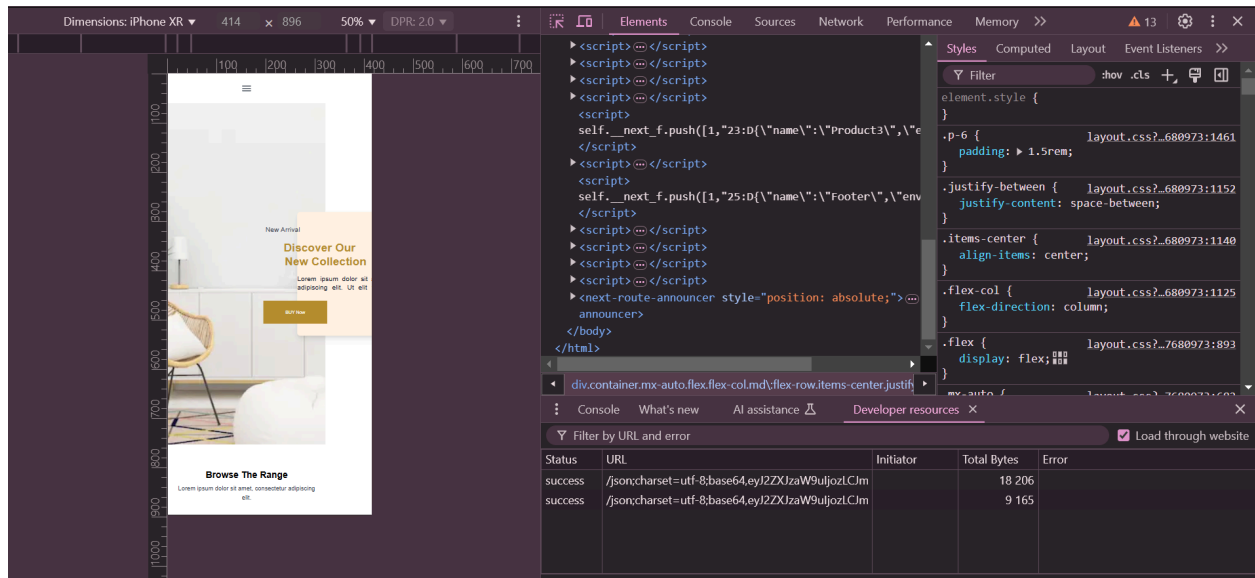
These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on [Core Web Vitals](#). [Learn more about Google Search Essentials](#).

CRAWLING AND INDEXING

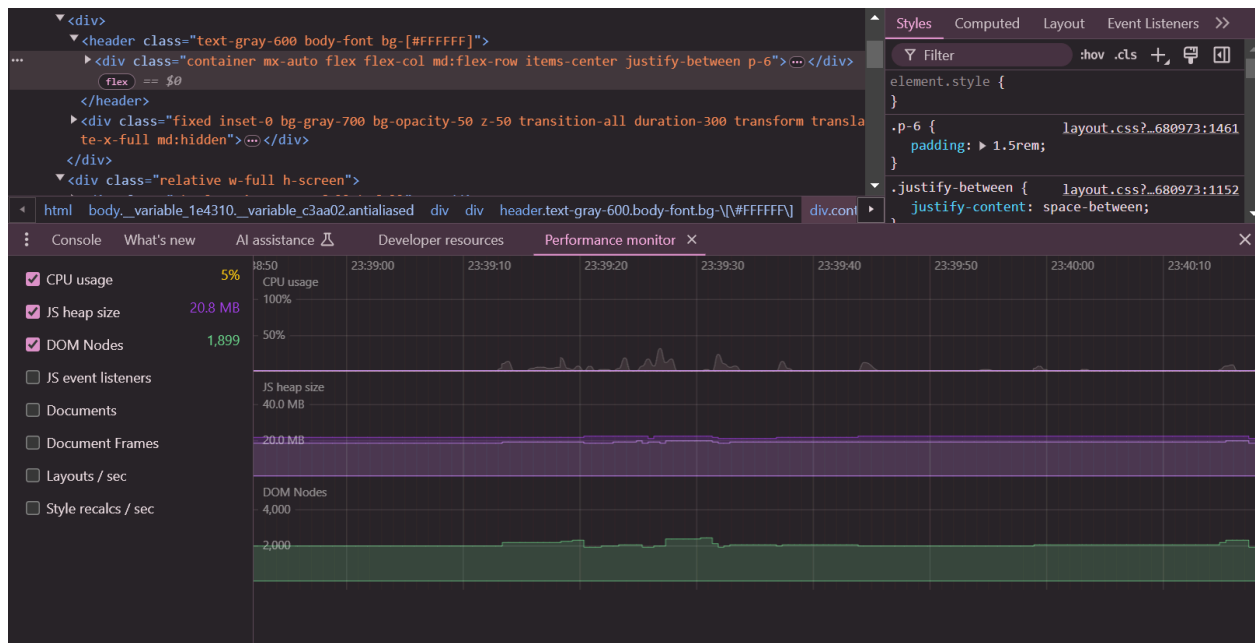
- ▲ Links are not crawlable ▼

To appear in search results, crawlers need access to your app.

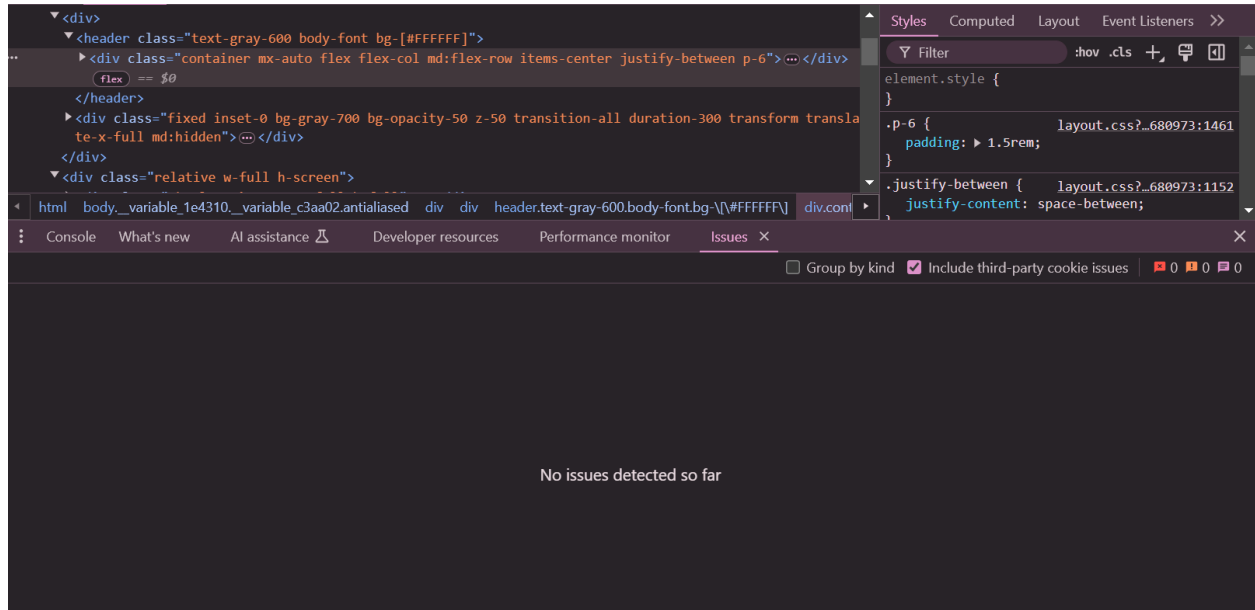
Developer Resources



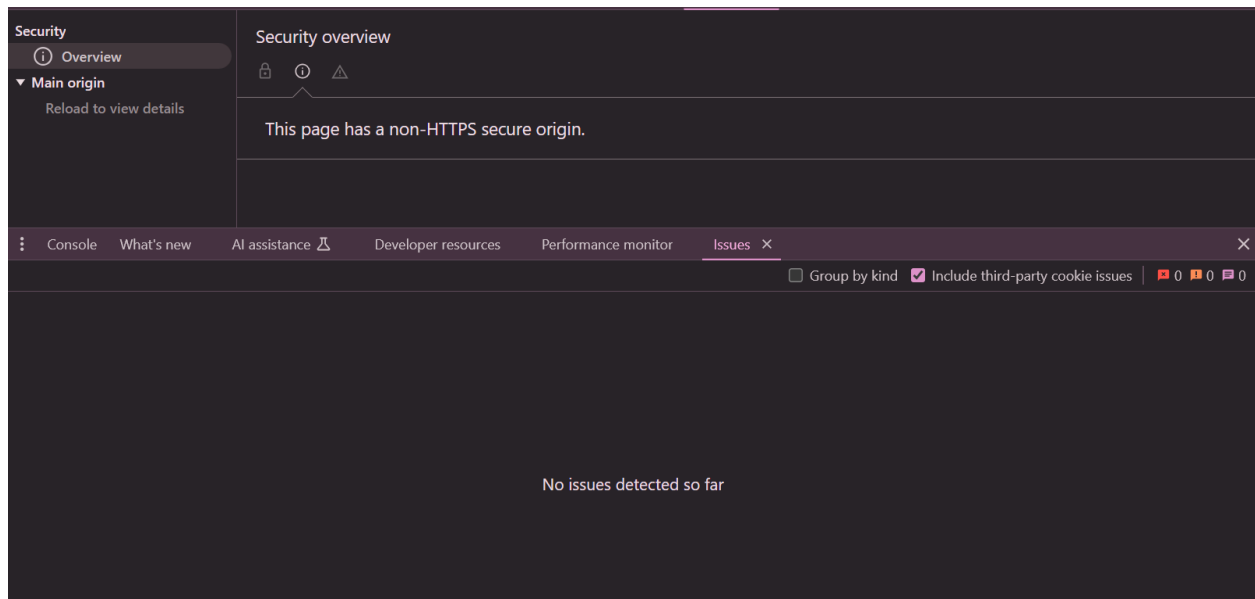
Performance Monitor




Issues Testing



HTTPS Testing



Testing Report Sample csv

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Status	Severity	Assigned	Remarks
TC001	Validate listing	Open > Verify	Products displayed	Products displayed	Passed	Low	-	No issues found
TC002	Test API error	Disconnect > Refresh	Show error UI	Error message shown	Passed	Medium	-	Handled gracefully
TC003	Cart functionality	Add to cart > Verify	Cart updates	Cart updates as expected	Passed	High	-	Works as expected
TC004	Responsiveness	Resize > Check layout	Layout adjust* 	Responsive layout issue	Failed	High	-	Issue with responsiveness

Day 5 Summary

On **Day 5**, I successfully:

- Performed Comprehensive Testing:** Validated the functionality of the marketplace features, including product listing, search, cart operations, and user profiles.
- Implemented Error Handling:** Added error messages and fallback UI for network failures, invalid data, and unexpected server errors to ensure a seamless user experience.
- Optimized Performance:** Used tools like **Lighthouse** and **GTmetrix** to identify and resolve performance bottlenecks, improving the page load time and responsiveness.

4. **Conducted Cross-Browser and Device Testing:** Ensured the marketplace was responsive and consistent across various browsers (Chrome, Firefox, Safari, Edge) and devices (mobile, tablet, desktop).
5. **Performed Security Testing:** Implemented input validation to prevent security issues like SQL injection and XSS, and ensured secure communication over **HTTPS**.
6. **Simulated User Acceptance Testing (UAT):** Tested the marketplace by simulating real-world user scenarios, verifying that browsing, searching, and checkout workflows were intuitive and error-free.
7. **Updated Documentation:** Created detailed documentation of the testing results, including a CSV-based testing report and a comprehensive summary of the fixes and optimizations.

By the end of **Day 5**, the marketplace was fully tested, optimized for performance, and prepared for real-world deployment.