LivingStyle

Hackathon Day 5

Testing, Error Handling, and Backend Integration Refinement

Objective:

Prepare "LivingStyle" for real-world deployment by thoroughly testing, optimizing performance, and refining backend integrations. Ensure readiness to handle customer-facing traffic while delivering a seamless and error-free user experience.

Comprehensive Testing:

1.Validate all features and backend integrations.

2.Ensure functional, non-functional, and user acceptance testing is completed. Error Handling:

- Implement user-friendly error messages and fallback mechanisms.

3.Gracefully handle API errors with fallback UI elements. Performance Optimization:

- Enhance speed, responsiveness, and reliability.

4.Conduct performance testing to identify and fix bottlenecks. Cross-Browser and Device Compatibility:

- Test for seamless functionality across multiple browsers and devices. Security Testing:
- Identify vulnerabilities and secure sensitive data. Documentation:
- Create professional testing reports (CSV-based) summarizing results and resolutions.

5.Prepare deployment ready documentation with best practices.

Functional Testing

Ensure the marketplace's key features operate as intended, delivering a smooth and error-free user experience.

Key Features to Test

Product Listing: Verify products are displayed accurately with proper details**.**

Product Details: Ensure product detail pages show correct information like price, description, images, and availability.

Dynamic Routing: Validate seamless navigation to individual product detail pages.

Responsive Design: Validate that all features and pages adapt seamlessly to various screen sizes (mobile, tablet, and desktop).

Performance Optimization

Optimize the marketplace to improve load times, responsiveness, and overall performance. This step involves identifying bottlenecks and applying strategies to enhance the user experience.

Image Optimization: Compress images using tools like TinyPNG or ImageOptim to reduce size without losing quality.

Lazy Loading: Implement lazy loading for images and assets to defer loading until needed, improving initial load times.

Minimize JavaScript and CSS:

Minification: Minify JavaScript (using Terser) and CSS (using CSSNano) to reduce file sizes for faster loading.

Remove Unused Code: Eliminate unused CSS and JavaScript to further reduce file sizes.

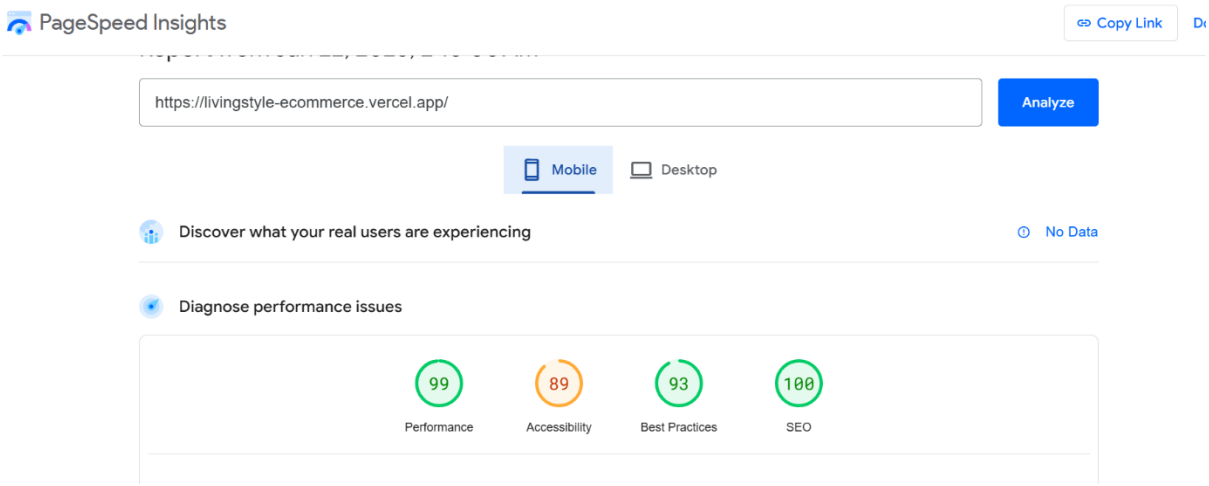
Implement Caching Strategies: Browser Caching: Store static assets in the user's browser to avoid repeated network requests.

Service Workers: Cache resources and improve offline functionality using service workers.

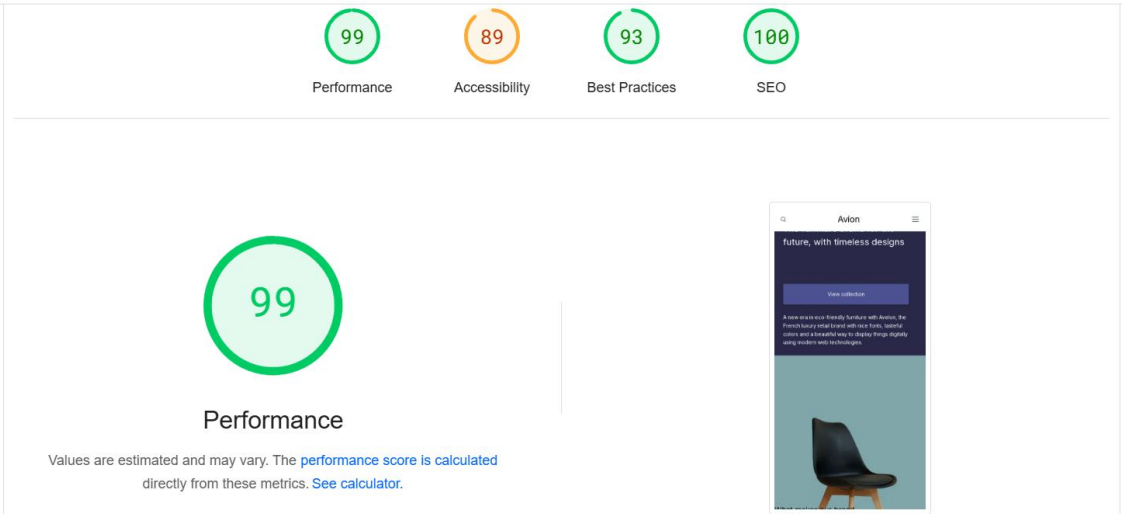
Analyze Performance: Google Lighthouse: Use Lighthouse to audit the website's performance and identify areas to improve, such as image optimization and resource loading.

WebPageTest & GTmetrix: Use these tools to identify performance bottlenecks and improve page load speeds.

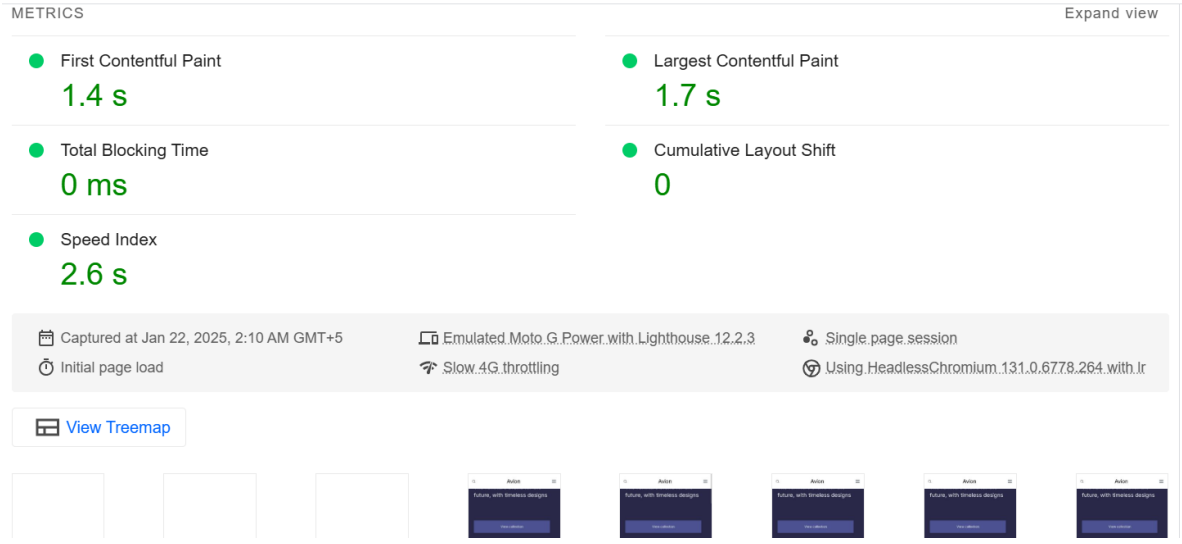
Analyze Performance Testing



Overview



Speed Test Performance at Mobile



Metrics

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.

NAMES AND LABELS

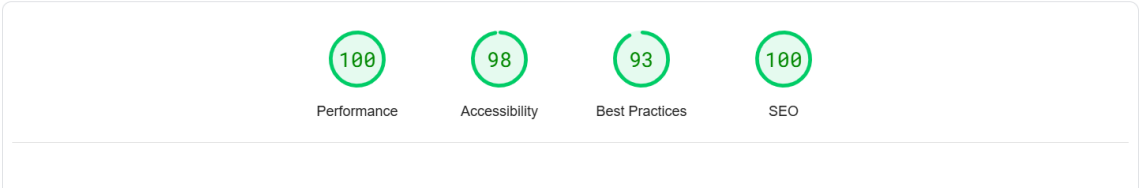
Overall Accessibility score at mobile

https://livingstyle-ecommerce.vercel.app/ Analyze

Mobile Desktop

Discover what your real users are experiencing No Data

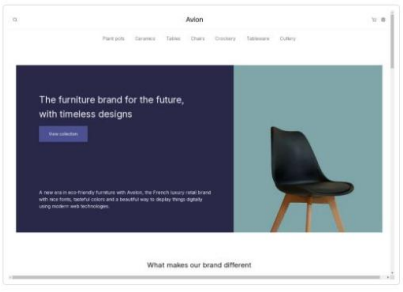
Diagnose performance issues



Speed performance at Desktop

Performance

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



Performance Score