

Optimizing website performance: A comprehensive Google Lighthouse study on desktop and mobile modes

Ahmed S. Al-Majidi¹, Sameer I. Al-Majidi²

¹ Al-Waely Secondary School for Distinguished Boys, Babylon, Iraq

² Babylon Sewerage Department, Babylon, Iraq

SUMMARY

Website technical performance is a major factor for user satisfaction and experience, especially in areas where the internet is slow or the operating device has limited capabilities. Therefore, Google LLC developed the Lighthouse tool to improve website performance; it measures five audit categories: Performance, Accessibility, Search Engine Optimization (SEO), Best Practices, and Progressive Web App (PWA). However, the consistency and reliability of its results, particularly via the Command Line Interface (CLI), have not been thoroughly explored. Therefore, this study aimed to investigate the reliability (stability across repeated runs) and consistency (agreement of scores across different tools (CLI vs DevTools) and modes (desktop vs mobile) of Lighthouse results. We hypothesized the Performance scores would vary significantly depending on the mode and website complexity, unlike Accessibility, SEO, and Best Practices scores. We ran Lighthouse on a real website (Detective) using both CLI and Chrome DevTools under desktop and mobile modes 3 times. To provide a point for comparison, we ran two additional sites: Wikipedia and the New York Times using DevTools one time. Performance scores significantly fluctuated: CLI runs on the Detective site ranged from 86 to 95, and DevTools tests revealed values between 21 and 100 across various modes and sites, while Accessibility, SEO, and Best Practices scores remained consistent (≥ 90). These findings suggest that developers may need to repeat tests for more stable outputs. This paper highlights the features and limitations of Lighthouse as an auditing tool and offers insights into best-use practices for web developers.

INTRODUCTION

In the digital age, website loading and performance are a main factor in determining user satisfaction and experience. A delay in page loading for a few seconds results in reduced user engagement and increased bounce rate (i.e., users leave the website after viewing only one page), especially in regions with limited network bandwidth or high network latency. Website performance optimization is also important because it can improve Search Engine Optimization (SEO), increasing a website's visibility in search engine results.

Google developed Lighthouse, a programming tool, to assist developers in raising website quality. It is open-source and can be used on web pages that require authentication. There are various ways of running the tool, including Chrome DevTools, which provides an interactive graphical interface for manually auditing webpages directly within the browser;

the Lighthouse Command Line Interface (CLI), which enables automated and repeatable audits from the terminal; and CI (Continuous Integration) environments, where Lighthouse audits can be executed automatically as part of software development.

This study focuses on Google Lighthouse because of its uniform, category-based scoring system and its availability in both DevTools and the CLI. Lighthouse evaluates Performance, Accessibility, Best Practices, Search Engine Optimization (SEO), and Progressive Web App (PWA). Unlike Lighthouse, GTmetrix results are less sensitive to CPU variability because tests are run from remote, controlled servers, which can generate more stable performance measurements across repeated runs. Compared to Lighthouse, WebPageTest tends to be slower for quick audits and can be more complex for beginner users. It lacks the tight integration with browser developer tools that makes Lighthouse more convenient for rapid, iterative testing.

Alternative analytics tools such as GTmetrix, WebPageTest, and PageSpeed Insights each offer different capabilities. GTmetrix is powered by Lighthouse, using its audits to evaluate site architecture and suggest performance improvements (1). WebPageTest provides network-level diagnostics, simulation of various networks (2G, 3G, 4G), and multi-device field testing (2, 3). PageSpeed Insights includes field testing performance data (field data) and lab testing from the Chrome User Experience Report (4). Unlike these tools, Lighthouse has a clear advantage: it provides unified category-based scoring, integrates directly with DevTools, and can be organized across CLI and CI pipelines.

Although each of these tools can give an idea of the site performance, Lighthouse is unique in its category-based scoring system, DevTools integration, and versatile use across various platforms, including CLI and CI pipelines. This study is based on Lighthouse because of its *All-in-One Tool*, where it doesn't just measure performance or SEO, but also evaluates accessibility, best practices, and PWA, and measures important Google metrics such as Largest Contentful Paint (LCP), Total Blocking Time (TBT), and Cumulative Layout Shift (CLS). The Performance score is calculated using metrics such as: First Contentful Paint (FCP), which measures the time until the first piece of content is rendered; Speed Index (SI), which measures how quickly content is visually displayed during page load; Largest Contentful Paint (LCP), which measures the time until the largest above-the-fold element is visible; Total Blocking Time (TBT), which measures the total time the main thread is blocked and unable to respond to user input; and Cumulative Layout Shift (CLS), which measures unexpected visual movement of page elements. Each raw metric value is

converted to a score between 0 and 100 using a log-normal distribution derived from real-world HTTP Archive data. The overall Performance score is then computed as a weighted average of these individual metric scores:

$$\text{Performance Score} = (0.10 \times \text{FCP}) + (0.10 \times \text{SI}) + (0.25 \times \text{LCP}) + (0.30 \times \text{TTT}) + (0.25 \times \text{CLS})$$

The Accessibility score is calculated from automated checks aligned with recognized accessibility standards (e.g., sufficient color contrast between text and background, descriptive labels for images and form fields, and clear heading hierarchy). The Best Practices score evaluates compliance with modern web development standards, such as using secure HTTPS connections, avoiding outdated browser features, and following current security recommendations. The SEO score reflects technical search-readiness signals such as meta tags, crawlability, and mobile compatibility. Each category score is computed as a weighted average of its individual audits according to Lighthouse's official scoring model. A PWA is a type of application software delivered through the web, built using common web technologies including HTML, CSS, and JavaScript, and works on any platform that uses a standards-compliant browser.

We hypothesized that Performance scores would differ significantly depending on the testing modes (desktop vs. mobile; DevTools vs. CLI) and website complexity, while Accessibility, SEO, and Best Practices would remain comparatively stable. We ran Lighthouse multiple times on a real website using both CLI and DevTools in desktop and mobile modes. We also ran additional DevTools tests on two well-known sites, Wikipedia and the New York Times (NYTimes). The results indicated inconsistency in Performance scores

across testing modes and websites, while the other categories remained constant (mostly ≥ 90). These results suggest that researchers and developers should perform repeated runs and report Lighthouse version and mode explicitly to improve reliability and reproducibility

RESULTS

To test our hypothesis that Lighthouse Performance scores would vary by testing mode and website complexity while Accessibility, SEO, and Best Practices remain stable, we compared three different websites (Detective, Wikipedia, and NYTimes). We selected these websites because of their different complexity levels, performance levels, and real-world relevance (i.e., how commonly each site is encountered by everyday users in actual browsing). The Detective site is static and has a lightweight design, Wikipedia has a medium-complexity design, and The NYTimes site has a complex design.

Running the Detective site using CLI

This subsection reports three consecutive Lighthouse CLI audits conducted on the Detective site. Although the Performance scores differed greatly (86–95), the Accessibility, SEO, and Best Practices scores were the same in all 3 runs. These results provide a raw comparison of 3 run scores between CLI and DevTools in Lighthouse categories on both desktop and mobile modes on the same site (**Figure 1**). These scores were considered reasonable because the non-Performance categories remained mostly stable, and the Performance variation was minor enough to attribute to the test device's limited hardware capabilities. The average scores for the four main categories were as follows: Performance: 89.3, Accessibility: 93.0, SEO: 100.0, and Best Practices: 93.0.

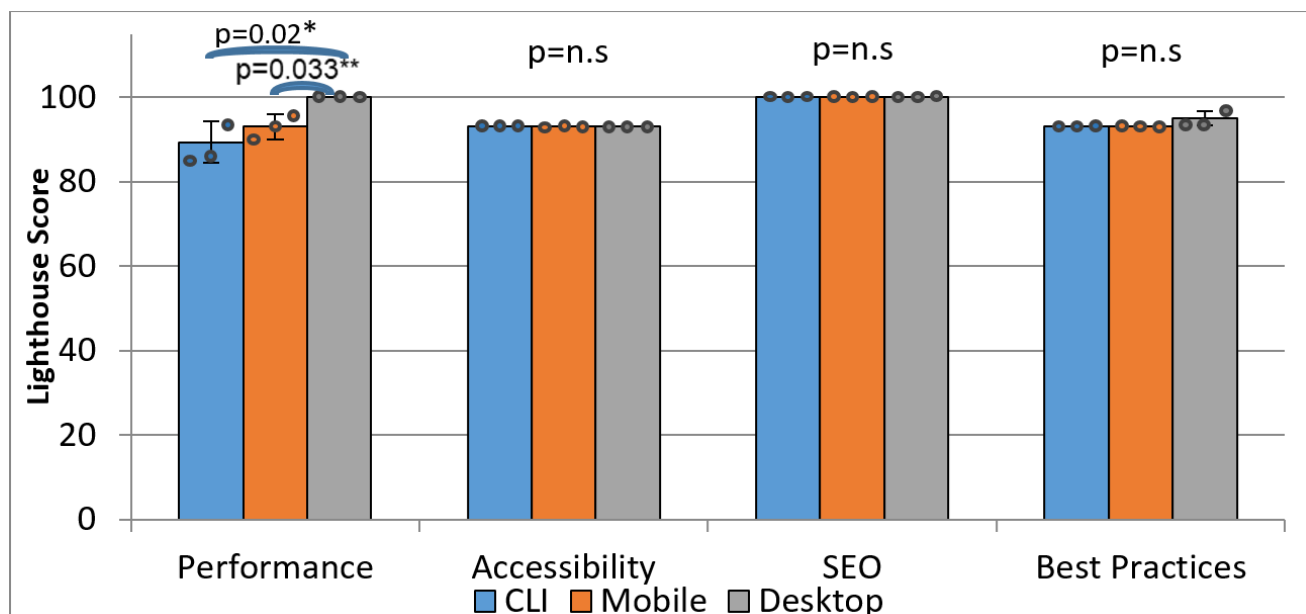


Figure 1. CLI and DevTools (Mobile and Desktop) Lighthouse scores of the Detective site. Bars represent the mean of three independent runs; error bars indicate standard deviation. Colored dots represent individual run scores (blue = CLI, orange = Mobile, gray = Desktop). An independent-samples t-test revealed a significant difference between CLI and Desktop in the Performance category ($p = 0.02^*$). A paired t-test also showed a significant difference between Desktop and Mobile modes for Performance ($p = 0.033^{**}$). No significant differences were observed in Accessibility, SEO, or Best Practices ($p > 0.05$).

Running the Detective site using DevTools – mobile mode

In mobile mode, the Detective site's Performance scores showed minor variation (93.0 ± 3.0). The Accessibility, SEO, and Best Practices scores were the same in all three runs. The average scores for the four main categories were as follows: Performance: 93.0, Accessibility: 93.0, SEO: 100.0, and Best Practices: 93.0 (**Figure 1**).

Running the Detective site using DevTools – desktop mode

In desktop mode, the Detective site's Best Practices scores showed minor variation (96–93). The Accessibility, SEO, and Performance scores were the same in all three runs. The average scores for the 4 main categories were as follows: Performance: 100.0, Accessibility: 93.0, SEO: 100.0, and Best Practices: 95.0. Compared to CLI, the Performance categories in DevTools results were more stable and complete (**Figure 1**).

Running the Wikipedia & The NYTimes site using DevTools – desktop & mobile mode

To confirm whether the results obtained from the Detective site were consistent across a broader range of web content, we conducted additional Lighthouse audits on two publicly available websites (Wikipedia and the NYTimes) using Chrome DevTools. Each site was tested in both Desktop and Mobile modes (**Figure 2**). We tested Wikipedia and NYTimes using DevTools only, as the CLI-DevTools comparison was already recognized through repeated runs on the Detective site. These additional sites served to examine complexity-driven score variation rather than execution-mode effects.

The statistical analysis revealed a significant difference in the Performance category between the CLI and Desktop modes ($p = 0.02$) (**Figure 1**). This suggests that the environment in which Lighthouse is executed can impact the Performance metrics. However, no significant differences were observed in Accessibility, SEO, or Best Practices ($p > 0.05$). The paired t-test analysis showed a statistically significant difference between desktop and mobile modes for the Performance category ($p = 0.033$). In contrast, no statistically significant differences were observed for Accessibility, SEO, or Best Practices ($p \geq 0.10$). Generally, the desktop mode has a higher

Performance score than the mobile mode.

DISCUSSION

We observed variation in Performance scores across testing modes and websites, while Accessibility, SEO, and Best Practices remained stable. The variations in Performance scores may be because of the variability in network speed, CPU activity, and differences in the execution environment between Chrome DevTools and the Lighthouse CLI. Also, because CLI audits run without a graphical browser interface, they can be more sensitive to background CPU load and system resource scheduling. When CPU resources are contested, TBT and LCP increase, and since these two metrics together carry 55% of the Performance score, the overall score drops. Accessibility, SEO, and Best Practices scores remained stable across all runs, as these categories are much less influenced by run-time execution factors. These fluctuations suggested that relying on a single Lighthouse run may lead developers to misjudge website performance and prioritize the wrong optimizations, especially when differences larger than 10 points occur. These findings imply that Lighthouse can be relied on to assess structural and semantic integrity but may require repeated runs with averaging to ensure accuracy in Performance metrics. Also, it is worth noting that geographic factors may contribute to the observed Lighthouse score fluctuations. With their main infrastructure located in the United States, Wikipedia and The NYTimes are worldwide-distributed websites that depend on extensive Content Delivery Networks (CDNs). The Detective site is hosted by GitHub and is also located in the United States. Geographical factors like server proximity, CDN optimization, and initial connection delays may still influence loading-related Performance metrics by affecting resource retrieval times, even though Lighthouse uses simulated network conditions for many audits. Therefore, rather than only website code, some of the observed performance variability might be due to underlying hosting and delivery differences.

The additional tests using Chrome DevTools in both mobile and desktop modes produced fully generated reports with no timeout errors or missing data. In CLI, errors and incomplete results can occur. This was observed previously in an analysis of Indian Institutes of Technology sites, because many of these sites lack mobile responsiveness, affecting

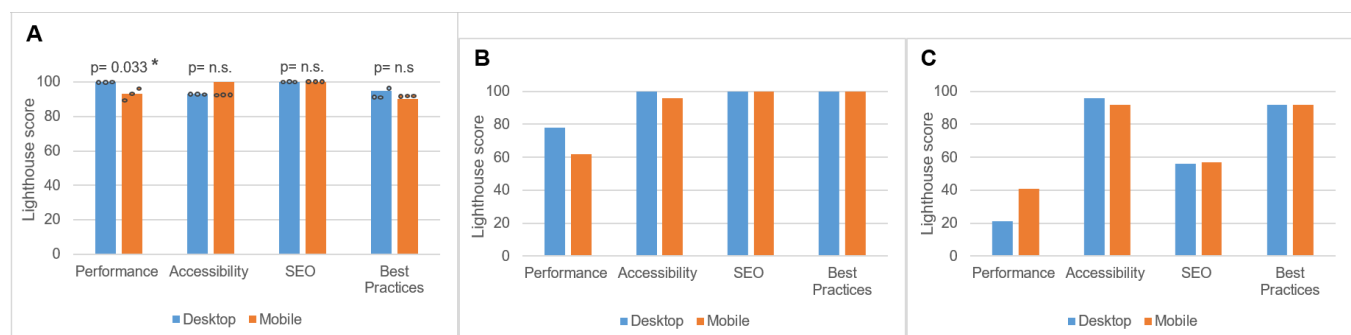


Figure 2. DevTools Lighthouse scores of the three websites (Detective, Wikipedia, and NYTimes) in Desktop and Mobile modes. Panel A shows the three independent runs for the Detective site, with error bars indicating standard deviation and dots representing individual run scores. Panels B and C show raw scores from single runs for Wikipedia and NYTimes, respectively. A paired t-test revealed a significant difference between Desktop and Mobile modes for the Detective site Performance category ($*p = 0.033$); no significant differences were observed in Accessibility, SEO, or Best Practices ($p = n.s.$).

accessibility and usability on smartphones and tablets (10). In addition, Desktop mode showed the highest scores across all categories compared with the other reports. Due to fewer constraints (e.g., reduced network or processing speed) and better resource availability during the audit, such as RAM and CPU, which are freer during analysis compared to Mobile or CLI. These results indicate that DevTools may offer more representative insights for developers targeting typical user environments. Lighthouse Performance metrics associated with factors that reduce measurement noise of Performance scores (such as image optimization, code size, compression techniques, and render-blocking resources) may contribute to variability in Performance scores if these factors are not controlled consistently across audit runs.

Interestingly, we observed an exception with the NYTimes site, where the Performance score was 41 in the Mobile mode and 21 in the Desktop mode. This difference could be explained by variations in how the website loads on mobile devices, such as providing a lighter or more optimized mobile version. These results indicate that while most categories are consistent, the Performance score is more sensitive to execution conditions and resource availability at the time of testing. This indicates that website design itself (not just the testing environment) affects audit results.

The report for the Detective site in the CLI showed that the tested device has a slow CPU. This can reduce the Performance score because of the hardware limitations or the system load. This may skew the results, especially in the Performance category. The Lighthouse report for the NYTimes site in mobile mode in Chrome DevTools showed that the results may be incomplete because the page loaded too slowly to finish within the time limit. However, this issue rarely appears when using Lighthouse via Chrome DevTools, which produced more stable and complete reports.

The mobile and desktop scores of the tested sites show that the structure and complexity of the website have a big impact on Lighthouse scores. Simple static sites (like the Detective site) typically receive higher Performance scores, while content-heavy websites like the NYTimes are penalized for large images, third-party scripts, and ad-related elements, consistent with the study by Dubey et al. (10).

It has previously been noted that Performance metrics for HTTPS websites, such as load time and responsiveness, were influenced by factors such as content size and server response time (11). In our Lighthouse analysis, the lower scores (especially in the Performance scores) were related to heavy JavaScript usage, large images, and delayed server responses. Similar results between the two studies support the conclusion that both server-side and client-side optimizations are important for improving website performance.

Our findings indicated that Lighthouse Performance scores varied across different sites. This is consistent with previous work (13); they reported that framework choice (e.g., Astro JS vs. Nuxt JS) influences performance and stability. This is consistent with our findings: both studies identify site-level characteristics (framework choice in their case, and content complexity in ours) as major determinants of Lighthouse Performance variation. Therefore, framework selection should be based on the specific performance requirement (e.g., system stability or quick content rendering).

Previous studies have explored extensions and enhancements to Google Lighthouse with the aim of supporting large-scale and repeated website audits. For example, one study suggested an enhanced Lighthouse-based system that enables audits using multiple input formats (e.g., URLs, CSV, JSON, XML) and produces structured outputs for large datasets (12). Their method allows for automated execution and average score calculation to facilitate more consistent and reproducible measurements across repeated runs. This aligns with our work: we likewise performed repeated runs and calculated average scores to reduce single-run variability, confirming that automation and execution consistency are critical for stable Lighthouse measurements.

In the context of Best Practices audits, inconsistent application of semantic HTML structures, mixed usage of inline and external resources, or varying security configurations (e.g., HTTPS enforcement and content security policies) may introduce fluctuations in Lighthouse results between runs (7, 8). Also, differences in validation practices, code organization, and progressive enhancement strategies can affect audit results. Therefore, maintaining consistent development and validation practices is important not only for achieving higher scores, but also for reducing variability and improving the reliability of repeated Lighthouse measurements (9). In our study, however, Best Practices scores remained stable across all runs and sites, suggesting that the tested pages did not vary in these structural or security-related factors.

For the PWA category in the Google Lighthouse audit to appear successfully, the web application must adhere to specific modern web standards. A primary reason for the PWA section not appearing is that a valid Web App Manifest is not present. This is a JSON configuration file that provides the browser with information about the web application (e.g., its name, icons, and start URL), which is important for allowing the app to be installed on a user's home screen. Furthermore, the site must be served over HTTPS to ensure security, and a registered Service Worker, which is a script that runs in the background, must be present to control the page and enable offline capabilities (14). Lighthouse checks whether the site meets these conditions and assigns a "PWA score" based on compliance. This score is not always included in default Lighthouse reports; it only appears when the page includes a valid manifest and Service Worker.

One of the study's limitations is that the study compares Lighthouse CLI and DevTools for a single website only; it does not assess the two methods across all tested sites due to time constraints. Furthermore, Lighthouse scores may fluctuate based on external factors like network speed (~30 Mbps), server response time, and the device constraints (4 GB RAM), which may marginally affect the results and the interpretation of Performance scores. Since v12.6.0 and v12.7.1 Lighthouse versions are part of the same major series (v12x) and have the same audit standards, it is possible that version-specific factors may have slightly influenced the results. However, according to official release notes, no basic audit mechanisms were changed between these versions; therefore, this is a negligible limitation. Understanding these version changes is important for researchers; an audit run on Lighthouse 5.0 may result in significantly different scores than one run with Lighthouse 6.0 or later, especially in Performance and Accessibility. Therefore,

any comparative study must explicitly state which Lighthouse version was used to ensure reproducibility.

Our results show that reproducible performance testing requires a standardized reporting protocol for developers and researchers. Developers should always state the tool and the tool version (e.g., Lighthouse 12.6.0), and the mode used (CLI vs DevTools). Also, developers should report device specs (CPU, RAM), network settings, or throttling profile. Finally, it is recommended that developers provide the number of independent runs and a summary statistic, raw values, or SD.

Future studies may expand this analysis by testing more diverse websites that vary in complexity levels via different tools in order to better generalize our results. By using GTmetrix or WebPageTest, they can also provide insights into the accuracy and dependability of different Performance metrics compared to Lighthouse results. Tests of the results of particular optimizations, like image compression, lazy loading, and JavaScript reduction, may also give developers useful guidance on how to enhance the performance of real-world websites, especially in areas with low network load and device hardware capabilities.

This study assessed Google Lighthouse's consistency and dependability using Chrome DevTools and CLI. We concluded that the results of a number of tests run on a real-world website indicate that the Performance category showed notable variations, especially when tested using CLI. On the other hand, SEO, Accessibility, and Best Practices scores remained stable. These results show how sensitive Performance metrics are to runtime factors and suggest that Chrome DevTools might provide developers with more stable and representative final results. DevTools may be more suitable for general use as it generates more complete reports with fewer audit failures and errors.

MATERIALS AND METHODS

Tools

In this study, the Lighthouse audits were run using two closely related versions from the v12.x series. Chrome DevTools ran with version v12.6.0, while the CLI reports were run using version v12.7.1. These versions use nearly identical scoring models and audit definitions. As such, no important differences are expected to arise due to version differences. No changes to the core score calculation model were documented between v12.6.0 and v12.7.1 (5, 6).

Environment

A web page was subjected to three consecutive Lighthouse audits through the CLI using Lighthouse 12.7.1 and Chrome DevTools using Lighthouse 12.6.0, with environmentally consistent settings. Additional Lighthouse audits on two publicly available websites (Wikipedia and the NYTimes) were conducted using Chrome DevTools (Lighthouse 12.6.0) in order to confirm whether the results obtained from the Detective site were consistent across a broader range of web content. Each site was tested in both Desktop and Mobile modes.

All reports were generated on a laptop with 4GB RAM (DDR3 1600 MHz) and an Intel Core i5-330M CPU running Windows 10, connected via a home Wi-Fi network with a contractual speed of 40 Mbps; the actual measured speed

was 30 Mbps during testing.

All sites were tested in DevTools on the same version (12.6.0); extensions were disabled, the browser cache was cleared, and the device remained on an identical network. This ensured that every Lighthouse test was conducted under constant conditions.

All Lighthouse audits were run in July 2025. The CLI experiments on the Detective site were performed on July 10, 2025. DevTools audits on the Detective site were performed on July 17, 2025. Additional DevTools audits on Wikipedia and NYTimes were performed on July 19, 2025. Because website content may change over time, these dates were provided to ensure reproducibility. The short period between testing dates reduces the possibility of significant changes affecting the results.

Data set selection

Three websites were selected according to their complexity, performance, and real-world relevance. The Detective site (itachi-15.github.io/detective/) was run three times in each mode. It has a lightweight structure, loads about 2.7 MB of resources, 14 requests, and zero scripts, making it a low-complexity site. Wikipedia (www.wikipedia.org) was run one time in each mode as a medium-complexity site. It loads about 3.4 MB of resources, 77 requests, and 14 scripts. The NYTimes (www.nytimes.com) was run one time in each mode as a high-complexity site. It has heavy content, a large number of media and scripts, and loads about 41 MB of resources, 400 requests, and more than 50 scripts. Each audit category score was analyzed, the average scores were recorded, and tables were used for comparison between the scores of each mode and run. All Lighthouse reports were saved in HTML format.

Test procedure

Chrome DevTools: The three sites were run via DevTools; the Detective site was run three times on both Desktop and Mobile modes, and Wikipedia and the NYTimes were run one time on both Desktop and Mobile modes. Although Chrome DevTools lacks automation, it provides reports in an easy-to-use format straight from the browser and determines which pages need authentication. Websites were opened in the Chrome browser. Then, the Developer Tools were accessed by pressing F12, and the Lighthouse section was selected. The categories to be analyzed (Performance, Accessibility, Best Practices, and SEO) were chosen. After that, the test type (mobile or desktop) was selected. Finally, the "Analyze page load" option should be clicked, and the results should be generated.

Lighthouse CLI: The highest amount of configuration and reporting flexibility for Lighthouse runs is offered by the Node CLI. The Node CLI is recommended for users who desire advanced functionality or who prefer to run Lighthouse automatically. One of the CLI's features is its headless, local execution, and it is beneficial for integration into automated pipelines for building, testing, and deploying code (CI/CD workflows). Lighthouse was installed via npm (Node Package Manager, a tool for installing and managing JavaScript packages) through the CLI to compare the Detective site result generated via the CLI with results generated via DevTools. The results were saved in HTML format. The complete list of

commands was provided in the Appendix.

Data are presented as the mean of three independent replicates. For comparisons involving three groups, a one-way Analysis of Variance (ANOVA) was performed, followed by Tukey's Honestly Significant Difference (HSD) to classify specific differences between pairs (**Figure 2**). To evaluate differences between desktop and mobile Lighthouse scores for the Detective site, paired t-tests were performed for each audit category (Performance, Accessibility, Best Practices, and SEO). For each category, scores from three independent runs in desktop mode were paired with corresponding runs in mobile mode. Statistical significance was assessed using a significance threshold of $\alpha = 0.05$. All statistical tests were conducted using Excel.

ACKNOWLEDGMENTS

We would like to express our gratitude to Dr. Mohammed Ibrahim for his general guidance and for providing valuable feedback on our work. His advice was instrumental in improving the quality of this research.

Received: August 19, 2025

Accepted: May 8, 2026

Published: October 2, 2026

REFERENCES

1. GTmetrix. "Everything You Need to Know About the New GTmetrix Report Powered by Lighthouse." *GTmetrix.com*. 10 Nov. 2020. gtmetrix.com/blog/everything-you-need-to-know-about-the-new-gtmetrix-report-powered-by-lighthouse/.
2. WebPageTest. "WebPageTest FAQs." *Docs.webpagetest.org*. docs.webpagetest.org/webpagetest-faqs/. Accessed 7 Aug. 2025.
3. WebPageTest. "API Reference." *Docs.webpagetest.org*. docs.webpagetest.org/api/reference/. Accessed 7 Aug. 2025.
4. Google Developers. "Get Started with PageSpeed Insights." *Developers.google.com*. developers.google.com/speed/docs/insights/v5/get-started. Accessed 7 Aug. 2025.
5. GoogleChrome/lighthouse. *Release 12.6.0*. *GitHub*. 29 Apr. 2025. github.com/GoogleChrome/lighthouse/releases/tag/v12.6.0.
6. GoogleChrome/lighthouse. *Release 12.7.1*. *GitHub*. 7 July 2025. github.com/GoogleChrome/lighthouse/releases/tag/v12.7.1.
7. Google. "Use Semantic HTML." *Web.dev*. Accessed 6 Aug. 2025. web.dev/learn/html/semantic-html/.
8. Mozilla Developer Network. "Content-Security-Policy." *Mozilla*. Accessed 7 Aug. 2025. developer.mozilla.org/en-US/docs/Web/HTTP/CSP.
9. W3C. "WAI Web Site Redesign Project: User Centered Design (UCD)." *W3C*. Accessed 7 Aug. 2025. www.w3.org/WAI/redesign/ucd.
10. Dubey, Vikrant and Shilpi Verma. "Evaluating The Digital Performance and Accessibility of IIT Library Websites Using Google Lighthouse." *International Journal of Computer Sciences and Engineering*, Vol. 13, 31 Mar. 2025, pp.16–23. doi:10.26438/ijcse/v13i3.1623 <https://ijcseonline.org/index.php/article/view/6687>
11. Gupta, Abhinav and Radim Bartos. "Improving Web Content Delivery with HTTP/3 and Non-Incremental EPS," *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, 22 Aug. 2024, pp. 1–9. doi.org/10.1109/ICCCN61486.2024.10637641
12. McGill, Thomas, et al. "Towards Improving Accessibility of Web Auditing with Google Lighthouse." *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2 Aug. 2023, pp. 1594–1599. doi.org/10.1109/COMPSAC57700.2023.00246
13. Zaidan, Ahmad and Dwi Suyatno. "Rendering Performance Analysis of Astro JS, Next JS, Nuxt JS, and SvelteKit Frameworks Using Google Lighthouse, PageSpeed Insight, and JMeter." *Journal of Emerging Information System and Business Intelligence*, Vol. 6, No. 1, 9 Apr. 2025, p.1–13. doi.org/10.26740/jeisbi.v6i1.64283
14. Google Developer. "Progressive Web App audits." *developer.chrome.com*. Accessed 7 Aug. 2025. developer.chrome.com/docs/lighthouse/pwa/

Copyright: © 2026 Al-Majidi and Al-Majidi. All JEI articles are distributed under the Creative Commons Attribution Noncommercial No Derivatives 4.0 International License. This means that you are free to share, copy, redistribute, remix, transform, or build upon the material for any purpose, provided that you credit the original author and source, include a link to the license, indicate any changes that were made, and make no representation that JEI or the original author(s) endorse you or your use of the work. The full details of the license are available at <https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en>.

Appendix

How to use CLI:

Install via npm:

```
npm install -g lighthouse
```

Run a basic audit:

```
lighthouse https://example.com --view
```

Export reports to JSON and HTML:

```
lighthouse https://example.com --output=json --output=html --output-path=./report
```

All lighthouse reports are available at:

<https://github.com/itachi-15/lighthouse->