UDC 004.056.55:004.312.2 **doi: 10.26906/SUNZ.2026.1.107**

Dmytro Tyrtyshnyi

National Technical University “Kharkiv Polytechnic Institute”, Kharkiv, Ukraine

COMPARATIVE ANALYSIS OF AUTOMATED TOOLS FOR CLIENT-SIDE PERFORMANCE TESTING IN MODERN WEB ENVIRONMENTS

Abstract. Relevance. In the era of Single Page Applications (SPA), the traditional approach to performance testing, focusing solely on server response times, is no longer sufficient. The logic of modern web applications has shifted to the client side. However, standard automated tools often produce misleading results under unstable network conditions, failing to capture the true user experience. **Object of research:** The process of testing and analyzing the performance of the client-side of web applications. **Subject of research:** Methods and software tools for automated analysis of web performance metrics. **Purpose of the article:** To conduct a comparative experimental analysis of automated testing tools (Google Lighthouse vs. Sitespeed.io) under constrained network conditions and on real-world e-commerce applications, identifying specific reliability gaps in standard auditing approaches. **Research results:** Experiments on a controlled slow application revealed that Google Lighthouse timed out, reporting a critical "False Positive" (Score 98/100) despite a 50-second load time. Conversely, Sitespeed.io correctly captured the full load duration with high statistical stability (RSD < 10%). Further tests on real-world platforms (Nike, Zara) demonstrated "False Negative" behavior in Lighthouse, which reported inflated LCP values significantly higher than manual observation. **Conclusions:** The analysis confirms that while Lighthouse is useful for general audits, a robust CI/CD framework requires the flexibility of Sitespeed.io to handle custom *pageCompleteChecks*, OS-level controlled network throttling, and complex visual elements without generating misleading pass/fail results.

Keywords: computer system, web performance; Core Web Vitals; Sitespeed.io; Lighthouse; false positives; false negatives; RSD; Visual Progress; UI performance.

Introduction

Problem Statement. The rapid development of web technologies has led to a fundamental paradigm shift in software architecture. With the mass adoption of Single Page Applications (SPA) and Progressive Web Apps (PWA), the primary computational load and rendering logic have shifted from the server side to the client side (browser). In such conditions, traditional performance testing approaches that focus exclusively on Server Response Time (TTFB) or backend throughput are becoming insufficient and often misleading.

User Experience (UX) today depends not on how fast the server sends data, but on how fast the browser can process it, build the DOM tree, and display an interactive interface. The complexity of modern JavaScript frameworks (React, Vue, Angular) creates a situation where a "fast" network request can lead to long blocking of the Main Thread on weak user devices. This directly affects business metrics: conversion rates, user retention, and SEO rankings. However, automated QA instrumentation often fails to keep pace with these changes. Standard auditing tools frequently ignore unstable mobile connectivity conditions (4G/LTE) or complex content loading scenarios, generating "false positive" results, which creates critical risks for the quality of the final product.

Analysis of Recent Research. The issues of client-side optimization and test automation are discussed in a number of modern scientific works. General approaches to optimizing mobile application performance and real-time UI are highlighted in works [1, 2]. The authors emphasize that for modern interfaces, not only load time but also the stability of visual presentation is critical. Research [3] analyzes front-end optimization methods using banking systems as an example, but the focus is shifted to architectural patterns rather than verification tools. Significant attention in the scientific community is paid to Core Web Vitals metrics. In particular, the Largest Contentful Paint (LCP) metric and the impact of JavaScript bundles and caching strategies on it are analyzed in detail in works [4–6]. The Interaction to Next Paint (INP) metric, which has recently become the standard for assessing interactivity, is considered in study [7]. The authors of work [8] propose a comprehensive framework for improving digital performance based on these metrics.

A separate layer of research concerns instrumentation. Work [9] investigates the correlation between the popularity of e-commerce resources and Google Lighthouse scores, confirming the importance of this tool as an indicator, but not as an absolute measure of quality. The issue of web performance automation is raised in [10], emphasizing the need for accessible tools for a wide range of developers. Works [11, 12] examine the use of modern methods for testing the client side, creating a basis for further comparative analysis.

However, despite the existence of research on individual metrics [5, 7] or general tool overviews [9, 10], the scientific literature does not sufficiently cover the problem of automation tool behavior (Lighthouse vs. Sitespeed.io) under "stress testing" conditions (limited network bandwidth, large media resources). Most studies consider "ideal" scenarios, ignoring the risks of obtaining unreliable data (False Positives/Negatives) during automated regression testing.

Purpose of the Article. The purpose of this work is to conduct a comparative experimental analysis of modern client-side performance automated testing tools (Google Lighthouse and Sitespeed.io). The research aims to identify the limitations of standard auditing tools under unstable network conditions and complex content, analyze mathematical models of visual readiness metrics (Visual Progress), and justify the choice of an architectural solution for building a reliable CI/CD testing system.

© Tyrtyshnyi D., 2026

1. Theoretical Basis and Mathematical Models

To build an effective testing framework, one must understand the mathematical models behind the metrics.

1.1. Core Web Vitals Models. Cumulative Layout Shift (CLS): Measures visual stability based on impact area and distance fraction:

$$CLSshift = ImpactFraction \times DistanceFraction, \quad (1)$$

Interaction to Next Paint (INP): Measures the full latency of user interaction:

$$INP = InputDelay + ProcessingTime + \\ + PresentationDelay, \quad (2)$$

Lighthouse Performance Score (v10): A logarithmic weighted average:

$$Score = 0.30(TBT) + 0.25(LCP) + 0.25(CLS) + \\ + 0.10(FCP) + 0.10(SI), \quad (3)$$

Crucially, if a tool stops recording early (timeout), TBT and CLS may be recorded as near-zero, artificially inflating the score.

1.2. Mathematical Model of Visual Completeness. Visual Completeness is determined by analyzing video frames using histogram intersection. To define these metrics, we first quantify the state of the screen at any given time *t*.

Let I_t be the video frame (image) captured at time *t*. The "Visual Complete" feature of Sitespeed use image histograms (frequency distribution of colors) to compare frames because they are robust against small pixel shifts.

- Let $H(I_t)$ be the RGB color histogram of the frame at time *t*.
- Let t_{start} be the time of the first frame (navigation start).
- Let t_{end} be the time of the final frame when the page has settled (no network/CPU activity for a set duration or any custom script defining this).

We define a Difference Function *D(A, B)* representing the difference between two frame histograms (typically the Sum of Absolute Differences):

$$D(A,B) = \sum_{i=1}^{n} |H(A)_i - H(B)_i| , \quad (4)$$

where *n* is the number of bins in the histogram.

The core component of all "VisualComplete" metrics is the Visual Progress function (5), denoted as *VP(t)*. It represents how close the current frame is to the final frame relative to the starting state.

$$VP(t) = \frac{D(I_{start},I_{end}) - D(I_t,I_{end})}{D(I_{start},I_{end})} \times 100\%, \quad (5)$$

- If *VP(t) = 0%:* The current frame F_t looks exactly like the empty white screen (F_{start}).
- If *VP(t) = 100%:* The current frame F_t looks exactly like the final loaded page (F_{end}).

VisualComplete[X] (e.g., 75, 85, 95, 99). This is the first point in time where the Visual Progress *VP(t)* meets or exceeds the threshold *X* and **remains** above that threshold for the remainder of the recording.

$$VisualComplete_x = min\{t | \forall_\tau \geq t, VP(\tau) \geq X\}, \quad (6)$$

VisualComplete95: The time *t* when the page is 95% visually complete.

$$t_{vc95} = \min\{t | VP(t) \geq 95\}, \quad (7)$$

In complex loading scenarios (e.g., a carousel loads, then a pop-up covers it or animated banners), *VP(t)* might drop temporarily.

The rigorous definition ensures we find the time after which it stays stable above *X*.

LastVisualChange. This is the timestamp of the final frame update before the page reaches its static, final state. It marks the moment when visual progress hits 100% and stops changing.

$$LastVisualChange = max\{t | VP(t) \neq VP(t - \delta t)\}, \quad (8)$$

where δt is the duration of a single video frame.

1.3. Statistical Reliability (RSD). To ensure that performance data is not result of network jitter, we calculate the Relative Standard Deviation (RSD), or Coefficient of Variation, across n iterations (where n=10 in our experiments):

$$RSD = \left|\frac{\sigma}{\mu}\right| \times 100\%, \quad (9)$$

where σ is the standard deviation and μ is the mean. An RSD<10% indicates a stable, controlled testing environment.

2. Methodology

To ensure a fair comparison, the network environment was standardized for both tools.

2.1. Network Profiles. Strict bandwidth and latency shaping were applied to simulate real-world conditions:

- Mobile 4G: Upload: 9000 kbps / Download: 9000 kbps / RTT: 85ms.
- Desktop 4G: Upload: 9000 kbps / Download: 9000 kbps / RTT: 85ms.
- WiFi 20Mb: Upload: 20 Mbps / Download: 20 Mbps / RTT: 20ms.
- WiFi 50Mb: Upload: 50 Mbps / Download: 50 Mbps / RTT: 20ms.
- Native: Unthrottled (~800 Mbps).

2.2. Tools & Configuration:

- Throttling Engine: Both tools utilized the "@sitespeed.io/throttle" npm package for machine-level packet shaping. This ensures that observed differences are due to the analysis engines, not the throttling method.
- Google Lighthouse: Chrome DevTools (v143) in Mobile and Desktop modes.
- Sitespeed.io (v39.2): Docker container with video and visualMetrics enabled. Configured for 10 iterations per run to calculate median values.

3. Experiment A: The "False Positive" (Controlled App)

Object: An unoptimized web application with large images (~10MB each) designed to load in approximately 50 seconds on a 4G network.

Lighthouse Results: Lighthouse reported a Performance Score (3) of 98/100 and an LCP of 1.5s. The report included an error: *"The page loaded too slowly to finish within the time limit"* (Fig. 1). Because the large images were slow to load, Lighthouse effectively timed out, analyzed only the initial header, which loaded fast, and ignored the rest of the page.

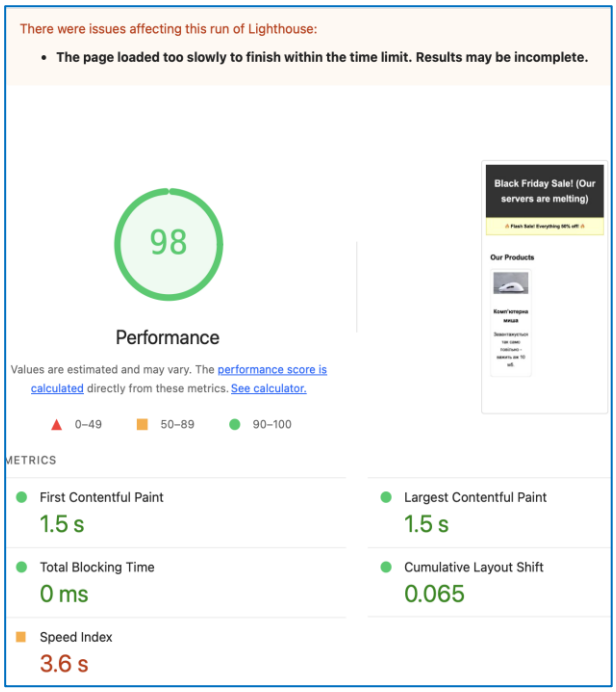

Fig. 1. Google Lighthouse falsely reporting a high score (98) because the test timed out before the heavy assets loaded

Sitespeed.io Results: Sitespeed.io was configured with a custom start flag: *--pageCompleteCheck: "return document.querySelectorAll('#product-grid.product-card'). length >= 9"*, waited for the product grid to populate (Fig. 2). It correctly reported LastVisualChange: 49.1s (8). Crucially, the statistical stability was high: the RSD (9) was < 10% across 10 iterations, proving the data was reliable.

Metric	1	2	3	4	5	6	7	8	9	10	Median	Mean	stddev
Largest Contentful Paint	1.036 s	916 ms	852 ms	1.020 s	984 ms	1.000 s	1.068 s	848 ms	952 ms	984 ms	984 ms	966 ms	70 ms
Interaction to Next Paint	440 ms	464 ms	432 ms	480 ms	424 ms	432 ms	504 ms	424 ms	528 ms	432 ms	436 ms	456 ms	35 ms
Last Visual Change	49.000 s	49.067 s	49.972 s	49.411 s	49.551 s	49.273 s	49.033 s	49.000 s	49.411 s	49.411 s	49.342 s	49.313 s	293 ms
Speed Index	6.154 s	5.581 s	5.585 s	5.524 s	5.551 s	5.447 s	5.730 s	5.170 s	5.333 s	5.526 s	5.539 s	5.560 s	245 ms
Visual Readiness	47.967 s	48.134 s	49.088 s	48.369 s	48.539 s	48.268 s	47.966 s	48.133 s	48.437 s	48.403 s	48.319 s	48.330 s	313 ms
Visual Complete 95	16.833 s	40.700 s	13.258 s	13.781 s	13.484 s	13.609 s	40.667 s	13.967 s	13.378 s	14.184 s	13.874 s	19.386 s	10.693 s
Visual Complete 99	48.967 s	49.033 s	49.938 s	49.378 s	49.517 s	49.240 s	49.000 s	48.967 s	49.378 s	49.378 s	49.309 s	49.280 s	292 ms

Fig. 2. Sitespeed.io correctly capturing the 49-second load time. Note the VisualComplete95 is 13.9s, indicating the user saw most content early, but the page was not fully finished

3.1. Analysis of Visual Progress. To understand the discrepancy, we analyzed the Visual Progress graph generated by Sitespeed.io (Fig. 3).

The curve shows a "stepped" progression. Lighthouse only captured the initial step (0-2s), while Sitespeed.io tracked the full progression to 100% at ~50s. The graph shows that the interface was ~97% complete for a long duration (15s–45s), whereas the histogram analysis in Sitespeed.io correctly identified that the final state had not yet been reached.

Additionally, the test was conducted on a Desktop viewport with the same 4g connection. In the Desktop Visual Progress Graph (Fig. 4) we could observe more gradual content visualisation on the page.

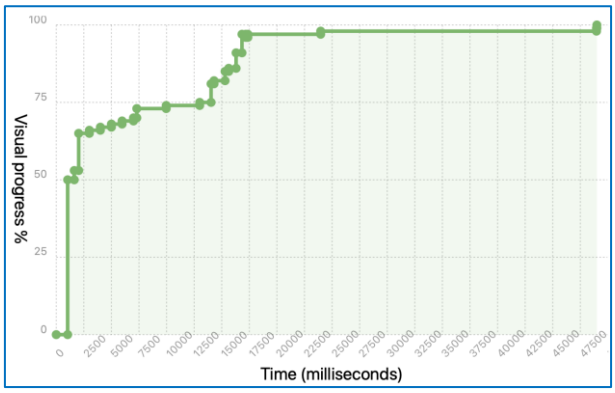

Fig. 3. Visual Progress Graph (VP(t)) - 4G - Mobile

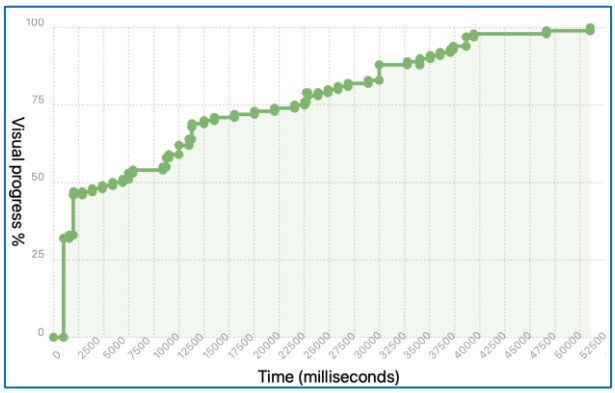

Fig. 4. Visual Progress Graph (VP(t)) - 4G - Desktop

The reason for that is the bigger viewport and more slow images (main bottleneck) of the slow web app impact the visible part of the screen more on Desktop, than on Mobile. Extra information for the Visual Progress could be also found in the “Filmstip” section of the Sitespeed report, there we could see screenshots of the app at different moments of the test execution (Fig. 5).

Fig. 5. Page Visualisation Progress (Filmstrip) - 4G - Desktop.

3.2. Impact of Network Conditions. We extended the experiment to faster networks to see if the tools converged (Tabl. 1, 2). Comparative Analysis:

1. Metric Accuracy: On the "Desktop WiFi 50Mb" profile, Sitespeed reported an LCP of 3.3s, which aligned with manual observation. Lighthouse reported an LCP of 10s. This discrepancy suggests that Lighthouse (in DevTools) may struggle with LCP timing identification.

2. Scaling: Sitespeed.io results scaled logically with bandwidth (LCP dropped from 13.4s → 1.3s as network improved). Lighthouse results were inconsistent, with LCP remaining high (10s) even on Native networks, potentially indicating a bug in the specific version or interference from browser extensions/overhead not present in the isolated Docker container.

3. VisualMetrics are available for the Sitespeed.io only, limiting the capabilities of the Lighthouse to analyse page rendering speed of the web sites under such constrained conditions.

4. Diagnostic Parity: Despite the metric differences, both tools correctly identified the "Opportunities" for optimization: "Properly size images" and "Serve images in next-gen formats."

Table 1 – **Sitespeed Results (Median of 10 runs, RSD <10%)**

Testing setup	LCP	VisualComplete95	LastVisualChange
Mobile - 4g (up:9000 down:9000 rtt:85)	1 sec (LCP is a <h1> on mobile, instead of img, which is loading very slowly)	13.8 sec (since viewport is smaller than on desktop)	49 sec
Desktop - 4g (up:9000 down:9000 rtt:85)	13.4 sec	40.7 sec	53 sec
Desktop - Wifi 20Mb (up:20000 down:20000 rtt:20)	6.3 sec	20.3 sec	24.9 sec
Desktop - Wifi 50Mb (up:50000 down:50000 rtt:20)	3.3 sec	8.8 sec	11.1 sec
Desktop Native - 800 Mbps	1.3 sec	2.65 sec	3.3 sec

Table 2 – **Lighthouse Results (Manual runs)**

Testing setup	LCP	SpeedIndex (Performance score)
Mobile - 4g (up:9000 down:9000 rtt:85)	1.5 sec (LCP is a h1 on mobile, instead of img, which is loading very slowly)	3.6 sec (98) with the error
Desktop - 4g (up:9000 down:9000 rtt:85)	13.3 sec	6.8 sec (63)
Desktop - Wifi 20Mb (up:20000 down:20000 rtt:20)	10 sec	2 sec (71)
Desktop - Wifi 50Mb (up:50000 down:50000 rtt:20)	10 sec (much higher than in Sitespeed and real-life. In reality the LCP img has been loaded in 3-4 sec)	1.3 sec (74)
Desktop - Native - 800 Mbps	10 sec (looks like a bug with LCP in Lighthouse with some Websites)	2.65 sec

Additionally, the Sitespeed gives an opportunity to watch the network requests waterfall and match it with the Visual metrics (Fig. 6). The correlation between sluggish load of .png images and slow LCP and Visual metrics (6,9) is obvious.

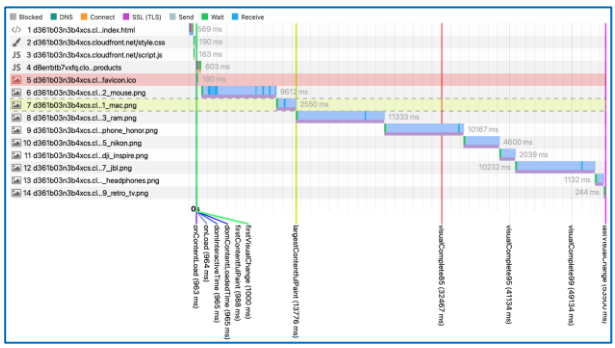

Fig. 6. Network requests waterfall - 4G - Desktop

Comparing this to the waterfall of the test with 800Mbps cable network, we could see that the resources download happens much faster by the browser, which leads to better Visual metrics (Fig. 7).

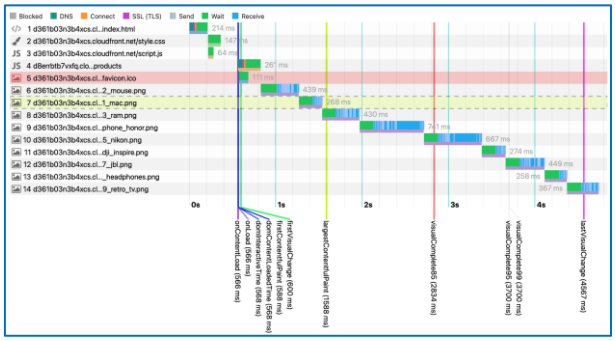

Fig. 7. Network requests waterfall - Native - Desktop.

4. Experiment B: Real-World E-Commerce Analysis

To validate these findings, we tested popular e-commerce platforms (Nike, Zara, Rozetka) under the same 4G Mobile conditions (Tabl. 3).

Table 3 – **Comparison of Real-World Metrics between Sitespeed and Lighthouse.**

URL tested	LCP (Sitespeed)	LCP (Lighthouse)	VisualComplete95(7)
https://www.nike.com/	4.9 sec	16.2 sec - innacurate (real ~5 sec)	10.3 sec*
https://www.zara.com/ua/uk/s-man-sale-l10847.html	5.6 sec	19.6 sec – innacurate (real ~5-6 sec)	9.9 sec*
https://rozetka.com.ua/ua/notebooks/c80004/	1.2 sec	2.3 sec	1.7 sec

4.1. The "False Negative" Phenomenon. On Nike.com, manual observation confirmed the main content loaded in 4-5 seconds.

- Sitespeed.io: Reported LCP of 4.9s, accurately reflecting reality (Fig. 8).
- Lighthouse: Reported LCP of 16.2s. This demonstrates a False Negative - Lighthouse flagged a performant site as slow (Fig. 9), likely misidentifying the LCP candidate element during the load process. A similar issue occurred with Zara.com (19.6s reported vs 5-6s real).

URL	Last Visual Change	Visual Complete 95	First Visual Change	First Contentful Paint	Largest Contentful Paint	Cumulative Layout Shift	Total Transfer Size	Total Requests
https://www.nike.-com/	11366	10396	1805	1792	4904	0.0056	4880.2	252
https://www.zara.-com/ua/uk/s-man-sale-l10847.html?v1=2439352	10074	9972	1867	1676	5668	0.0126	4886.2	139
https://rozetka.-com.ua/ua/note-books/c80004/	3017	1750	1242	1212	1212	0	2754.8	406

Fig. 8. Sitespeed – LCP and other metrics for Experiment B

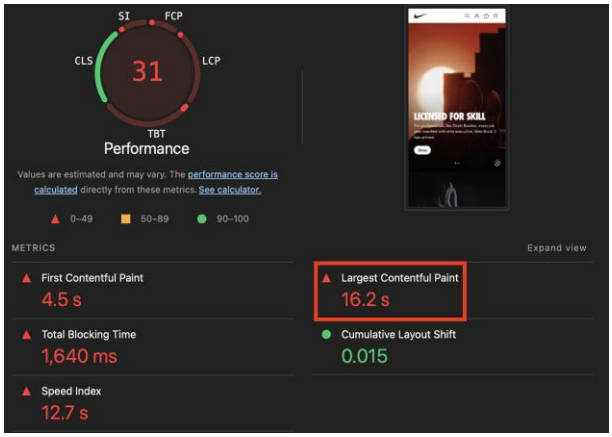

Fig. 9. Lighthouse – LCP and other metrics for Nike

4.2. The Challenge of Animation. While Sitespeed.io was more accurate, it highlighted a specific challenge with VisualComplete. On Nike.com, an animated banner caused the VisualComplete95 (7) metric to be 10.3s, even though the site was interactive much earlier.

The histogram analysis (*VP(t)*) interpreted the animation as "unfinished loading" (Fig. 10).

Solution: This proves the advantage of Sitespeed's flexibility. By analyzing the generated Filmstrip or configuring a custom *pageCompleteCheck* to ignore the banner, we can obtain the true metric. Lighthouse does not offer this granular control.

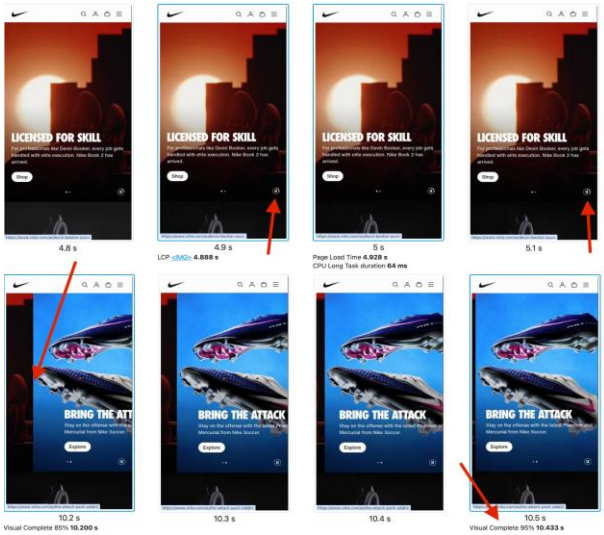

Fig. 10. Animated banner, impacting VisualComplete metrics

4.3. Statistical Validity (RSD):

- Nike/Rozetka: RSD (9) was ~10%, indicating stable test conditions.
- Zara: RSD (9) was ~25%. This high deviation indicates that the instability lies within the Zara application itself (e.g., variable API response times), not the tool. Sitespeed's ability to report RSD is critical for distinguishing between "tool noise" and "platform instability."

Conclusions

A comparative experimental analysis of methods and tools for automated client-side performance testing was conducted under constrained network conditions on controlled unoptimized web application and on real-world e-commerce applications. This result will expand the possibilities of designing robust continuous integration (CI/CD) systems capable of detecting performance regressions in complex web environments without generating false positive validation results.

Experimental data confirmed that, compared to standard auditing tools (such as Google Lighthouse) which demonstrated "False Positive" results (Score 98 vs 50s load time) and "False Negatives" (inflated LCP values), the Sitespeed.io framework significantly increases the reliability of quality control. The obtained high statistical validity (RSD < 10% across 10 iterations) and the accurate capture of Visual Progress metrics allow for the precise identification of "bottlenecks" that are invisible to standard timeout-based algorithms.

Furthermore, the architectural flexibility of the Sitespeed.io engine, specifically the ability to implement custom completion logic (*pageCompleteCheck*) and analyze histogram-based visual metrics, allows it to be integrated as the core component of a comprehensive

automated testing framework with historical data aggregation, which is the primary objective of further research.

Use of Artificial Intelligence Tools. The author confirm that artificial intelligence technologies were not used in the creation of the presented work.

REFERENCES

1. G. C. Vegineni, "Real-time Performance Optimization in Modern UI Applications," *2025 3rd International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*, Coimbatore, India, 2025, pp. 528-533, doi: https://doi.org/10.1109/ICoICI65217.2025.11252913
2. Hort, M., Kechagia, M., Sarro, F. and Harman, M. (2021), "A Survey of Performance Optimization for Mobile Applications", *IEEE Transactions on Software Engineering*, vol. 47, no. 8, pp. 1–26, doi: https://doi.org/10.1109/TSE.2021.3071193
3. Ericsson, T. (2013), "Front-end website performance optimisation: Optimising the front-end performance of Swedbank's website", Dissertation, *DIVA Portal*, URL: https://www.diva-portal.org/smash/get/diva2:645641/FULLTEXT01.pdf
4. Edgar, M. (2023), "Largest Contentful Paint", in *Speed Metrics Guide: Choosing the Right Metrics to Use When Evaluating Websites*, Apress, Berkeley, CA, pp. 137–152, doi: https://doi.org/10.1007/979-8-8688-0155-6_8
5. Д. А. Тиртишний, С. Ю. Леонов (2024), "Вплив JavaScript-бандлів на метрику Largest Contentful Paint (LCP) та рекомендації щодо оптимізації", *NTU "KhPI" Repository*, URL: https://repository.kpi.kharkov.ua/items/50f96f0a-f773-4ebb-b852-d76e080efc3a.
6. Д. А. Тиртишний, С. Ю. Леонов (2024), "Стратегії кешування контенту для оптимізації LCP у динамічних веб-додатках", *NTU "KhPI" Repository*, URL: https://repository.kpi.kharkov.ua/items/f429dd8c-f97d-42f9-843a-dad89ee7c4b5.
7. Д. А. Тиртишний, С. Ю. Леонов (2024), "INP як метрика оцінки взаємодії користувача з вебсторінкою", *NTU "KhPI" Repository*, URL: https://repository.kpi.kharkov.ua/items/6c41404c-1df4-409a-a6a8-a8661b44e7c4.
8. Ranjith Reddy Gaddam (2025), "Optimizing Core Web Vitals: A Comprehensive Framework for Enhanced Digital Performance", *Sarcouncil Journal of Engineering and Computer Science*, pp. 704–711, URL: https://sarcouncil.com/download-article/SJECS-218_-_2025-704-711.pdf.
9. Thomas, I. (2024), "The Correlation of the Popularity of E-commerce Websites and the 6 Metrics of the Lighthouse Performance Score", *Furman University Scholar Exchange*, URL: https://scholarexchange.furman.edu/scjas/2024/all/478/.
10. Marx, R. (2018), "Web Performance Automation for the People", in *Companion Proceedings of the The Web Conference 2018 (WWW '18)*, ACM, New York, NY, USA, pp. 825–829, doi: https://doi.org/10.1145/3184558.3186570.
11. Д. А. Тиртишний, С. Ю. Леонов (2024), "Використання сучасних методів тестування та аналізу клієнтської частини веб-додатків", *NTU "KhPI" Repository*, URL: https://repository.kpi.kharkov.ua/items/c4f8bf52-2684-43d2-a2ae-6aa1b64b1923.
12. Leonov, S., Tyrtyshnyi, D. (2025), "Development of a software platform for testing the performance of the client part of a web application", *Control, Navigation and Communication Systems*, Vol. 1, No. 79, pp. 111–115. doi: https://doi.org/10.26906/SUNZ.2025.1.111-115

Received (Надійшла) 07.01.2026
Accepted for publication (Прийнята до друку) 11.02.2026
Publication date (Дата публікації) 27.02.2026

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Тиртишний Дмитро Андрійович – аспірант, Навчально-науковий інститут комп'ютерних наук та інформаційних технологій, кафедра Комп'ютерна інженерія та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Dmytro Tyrtyshnyi – PhD student, Postgraduate, Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: dmytro.tyrtyshnyi@gmail.com; ORCID Author ID: https://orcid.org/0009-0000-4935-7156;

Порівняльний аналіз автоматизованих інструментів для тестування продуктивності клієнтської частини в сучасних веб-середовищах

Д. А. Тиртишний

Анотація. Актуальність. В епоху односторінкових додатків (SPA) традиційний підхід до тестування продуктивності, що фокусується виключно на часі відповіді сервера, більше не є достатнім. Логіка сучасних веб-додатків змістилася на клієнтську частину. Однак стандартні автоматизовані інструменти часто дають хибні результати в умовах нестабільної мережі, не відображаючи справжній досвід користувача. **Об'єкт дослідження:** процес тестування та аналізу продуктивності клієнтської частини веб-додатків. **Предмет дослідження:** методи та програмні засоби для автоматизованого аналізу метрик веб-продуктивності. **Мета статті:** провести порівняльний експериментальний аналіз інструментів автоматизованого тестування (Google Lighthouse проти Sitespeed.io) в умовах обмеженої пропускної здатності мережі та на реальних додатках електронної комерції, ідентифікувавши специфічні прогалини в надійності стандартних підходів до аудиту. **Результати дослідження:** Експерименти на контрольованому додатку зі сповільненою мережею виявили, що Google Lighthouse переривався через таймаут, видаючи критичний «хибно-позитивний» результат (оцінка продуктивності 98/100), незважаючи на 50-секундний час завантаження сторінки. Натомість Sitespeed.io коректно зафіксував повну тривалість завантаження з високою статистичною стабільністю (RSD < 10%). Подальші тестування на реальних платформах (Nike, Zara) продемонстрували «хибно-негативну» поведінку Lighthouse, який звітував про завищені значення LCP, що значно перевищували результати ручного спостереження. **Висновки:** Аналіз підтверджує, що хоча Lighthouse є корисним для загальних аудитів, побудова надійного фреймворку CI/CD вимагає гнучкості Sitespeed.io для обробки специфічних умов завершення завантаження сторінки (pageCompleteChecks), керованого обмеження мережі на рівні ОС та складних візуальних елементів без генерації оманливих результатів проходження тестів.

Ключові слова: комп'ютерна система, веб-продуктивність; Core Web Vitals; Sitespeed.io; Lighthouse; хибно-позитивні результати; хибно-негативні результати; RSD; Visual Progress; продуктивність UI.