VOL. 10. NO. 4 MAY 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v10i4.6238

PWA AND NON-PWA PERFORMANCE ANALYSIS: CHROME EXTENSION TESTING ON E-COMMERCE PLATFRORM

Panji Revolusioner Saputro[1]*; Rifda Faticha Alfa Aziza[2]

Informatic[1,2]
Universitas Amikom Yogyakarta, Sleman, Indonesia[1, 2]
https://home.amikom.ac.id/[1,2]
pannji22revolusio@students.amikom.ac.id[1]*, rifda@amikom.ac.id[2]

(*) Corresponding Author
(Responsible for the Quality of Paper Content)

Abstract*—This study compares Progressive Web Apps (PWA) and traditional web applications performance using a custom Chrome extension and Google Lighthouse, focusing on Tokopedia's e-commerce platform. The research employs a quantitative approach with controlled testing environments across three viewports for the custom extension (desktop, tablet, mobile) and two viewports for Google Lighthouse (desktop, mobile). The custom extension measures eleven metrics, including Core Web Vitals, PWA features, and resource usage, while Google Lighthouse provides five core metrics. Results show PWA implementation improves performance with 9.9% better First Contentful Paint on desktop and significant memory efficiency (29-33MB vs 59-62MB). The comparison between testing tools reveals methodology differences, with custom extension showing optimistic results in real-world conditions and Lighthouse providing more conservative measurements under throttled conditions. This research contributes to PWA performance measurement methodology by combining real-world and standardized testing approaches.*

Keywords*: chrome extension, performance analysis, progressive web apps, tokopedia, web metrics*

Intisari—*Penelitian ini menyajikan analisis komparatif antara Progressive Web Apps (PWA) dan aplikasi web tradisional menggunakan ekstensi Chrome khusus dan Google Lighthouse, dengan fokus pada platform e-commerce Tokopedia. Penelitian menggunakan pendekatan kuantitatif dengan lingkungan pengujian terkontrol pada tiga viewport untuk ekstensi khusus (desktop, tablet, mobile) dan dua viewport untuk Google Lighthouse (desktop, mobile). Ekstensi khusus mengukur sebelas metrik, termasuk Core Web Vitals, fitur PWA, dan penggunaan sumber daya, sementara Google Lighthouse menyediakan lima metrik inti. Hasil menunjukkan implementasi PWA meningkatkan performa dengan First Contentful Paint 9.9% lebih baik pada desktop dan efisiensi memori yang signifikan (29-33MB vs 59-62MB). Perbandingan antara tools pengujian mengungkapkan perbedaan metodologi, dengan ekstensi khusus menunjukkan hasil optimis dalam kondisi nyata dan Lighthouse memberikan pengukuran lebih konservatif dalam kondisi throttling. Penelitian ini berkontribusi pada metodologi pengukuran performa PWA dengan menggabungkan pendekatan pengujian dunia nyata dan terstandarisasi.*

Kata Kunci*: analisis performa, ekstensi chrome, metrik web, progressive web apps, tokopedia*

INTRODUCTION

The rapid development of web technology has driven the evolution of web-based applications. One of the innovations that has emerged is Progressive Web Apps (PWA), which offers a solution to bridge the gap between traditional web applications and native applications [1][2][3]. According to Kumar et al. [4] and Mhatre and Mali[5], PWA has changed the application development paradigm by presenting a better user experience through offline capabilities, faster access speeds, and features that approach native applications. Rochim et al. [1], Bennervall et al. [3], and Heričko et al. [2] in their research, revealed that PWA shows better performance in terms of response time and resource usage compared to traditional web applications. This is reinforced by the findings of Vepsäläinen et al. [6] and Nichifor et al. which prove that PWA has more optimal energy efficiency and loading time. Ribeiro et al. [7] added that implementing PWA in service-oriented applications can increase user engagement significantly and reduce the bounce rate.

In the implementation context, Kweche Ebechue and Te Dorsthorst [8] identified several challenges in implementing PWAs, especially regarding browser support, app visibility in app stores, and caching strategies. Fahad and Chowdhury [9] highlighted the importance of choosing the right caching strategy, as suboptimal implementation can affect PWA performance, especially on frequently updated sites. This aligns with the findings of Cherukuri [10], who emphasized the importance of comprehensive performance measurement in ensuring effective PWA implementation.

In the context of e-commerce, Eunike et al. [11], Muna et al. [12], and Haryanto and Elsi [13] demonstrated that PWA implementation can improve performance scores by up to 30% compared to traditional web applications. Through their research, Ribeiro [7] and Muman [14] underlined the importance of in-depth evaluation of PWA performance using standardized tools. This is reinforced by Leshchuk et al. [6], who suggested using standardized metrics for PWA performance evaluation.

Soetanto et al. [15] found that PWA provides significant improvements in terms of loading time and resource efficiency, especially in the context of applications that require intensive user interaction. This finding aligns with the results of McGill et al. [7], which shows that PWA can optimize memory and CPU usage by up to 40% compared to traditional web applications. Faizin et al. [16] added that PWA also provides advantages in terms of accessibility and user experience, especially on devices with medium to low specifications.

While previous studies have highlighted the advantages of PWA, such as offline capabilities and improved user experience, there is a lack of comprehensive research that compares PWA and non-PWA performance using multiple testing methodologies [6][11]. Most existing studies rely solely on standardized tools like Google Lighthouse, which may not fully capture real-world performance conditions [2][10]. This study aims to fill this gap by providing empirical evidence of PWA performance improvements, particularly in the context of e-commerce platforms, where performance and user experience are critical for success [17].

The primary objective of this study is to conduct a comparative analysis of Progressive Web Apps (PWA) and traditional web applications, focusing on performance metrics such as Core Web Vitals, resource usage, and PWA-specific features [6]. This research aims to provide a comprehensive evaluation of PWA performance using both real-world testing (via a custom Chrome extension) and standardized testing (via Google Lighthouse). By combining these two methodologies, the study seeks to address the gap in existing literature and offer a more holistic understanding of PWA performance in the context of e-commerce platforms [3][17]. This approach is expected to provide new insights into PWA performance evaluation and serve as a valuable reference for developers in optimizing PWA implementation for e-commerce applications in the increasingly competitive digital era [5][16][7].

MATERIALS AND METHODS

This study uses a comparative quantitative approach to analyze the performance of Progressive Web Apps (PWA) and traditional web applications (non-PWA), adopting the methodology used by Rochim et al. [1]and Heričko et al.[2] in evaluating web application performance. The object of the study focused on the Tokopedia e-commerce website (https://www.tokopedia.com/), which was chosen because it is one of the largest e-commerce platforms in Indonesia that has implemented PWA technology, as analyzed in the research of Haryanto and Elsi [13]and Muna et al.[12].

Tokopedia was selected as the study object due to its status as one of the largest e-commerce platforms in Indonesia, with significant traffic and a diverse user base. The platform's implementation of PWA technology provides an ideal case study for analyzing the performance differences between PWA and traditional web applications in a real-world e-commerce context. The high traffic and diverse user base of Tokopedia allow for a comprehensive evaluation of PWA performance under realistic conditions. Additionally, Tokopedia's widespread use ensures that the findings of this study are relevant to a broad audience and can be generalized to other e commerce platforms. Following the methodology of McGill et al [7].

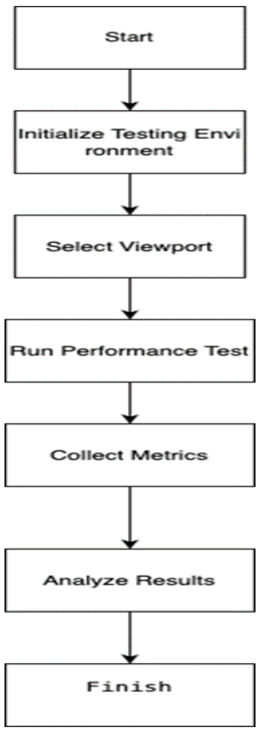
Performance testing was carried out with two different tools, each with different viewport coverage and metrics. The developed Chrome extension tested on three viewports: desktop

Accredited Rank 2 (Sinta 2) based on the Decree of the Dirjen Penguatan RisBang Kemenristekdikti No.225/E/KPT/2022, December 07, 2022. Published by LPPM Universitas Nusa Mandiri

VOL. 10. NO. 4 MAY 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v10i4.6238

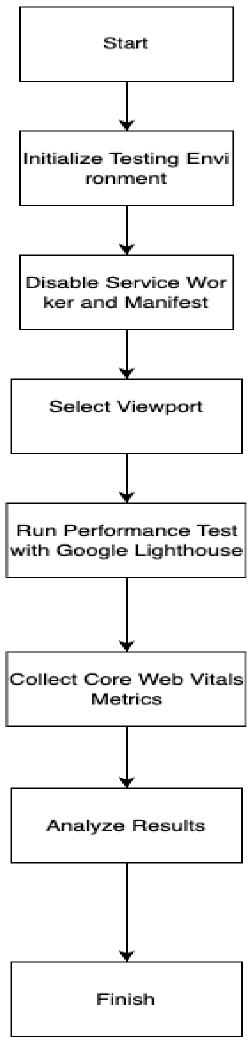
(1366x768), tablet (768x1024), and mobile (375x667), while Google Lighthouse tested on two viewports: desktop and mobile only. The metrics used in this study are derived from the Core Web Vitals framework proposed by Google (Google, 2020), which includes: First Contentful Paint (FCP) is the time it takes for the first content element to render on the screen. Time to Interactive (TTI) is the time it takes for the page to become fully interactive. Speed Index is a measure of how quickly content is visually displayed during page load. Total Blocking Time (TBT) is the total time the main thread is blocked, preventing user input. Largest Contentful Paint (LCP) is the time it takes for the largest content element to render. Cumulative Layout Shift (CLS) is a measure of layout stability during page load.

Additionally, the study measures PWA-specific features such as: Offline Capability is the ability of the application to function without an internet connection. Install Prompt refers to the availability of prompts to install the PWA on the user's device. Resource usage metrics are also measured, including: Page Size is the total size of the page resources. CPU Usage represents the percentage of CPU resources consumed by the application. Memory Usage refers to the amount of memory (RAM) used by the application. These metrics were selected based on their relevance to PWA performance and their ability to provide a comprehensive evaluation of user experience.



Source : (Research Results, 2024)
Figure 1. Custom Extension Testing Flow

Referring to the methodology used by Vepsäläinen et al. [9]and Faizin et al.[16], the testing environment was strictly controlled using devices with the following specifications: AMD Ryzen 3 1300X Quad-Core processor, 16GB RAM, NVIDIA GTX 750 2GB GPU, XIAOMI A27I monitor, and 30 Mbps Indihome internet connection. To ensure consistency of the results, testing was conducted at off-peak times (01.00 - 05.00 WIB) with a 5-minute interval between each test and repeated three times for each scenario and viewport, following Ribeiro et al.'s protocol for minimizing network variability in performance testing.


Start
Initialize Testing Envi ronment
Disable Service Wor ker and Manifest
Select Viewport
Run Performance Test with Google Lighthouse
Collect Core Web Vitals Metrics
Analyze Results
Finish

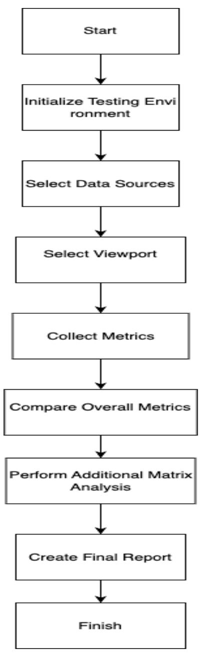
Source : (Research Results, 2024)
Figure 2. Google Lighthouse Testing Flow

The developed Chrome extension measures eleven different metrics, expanding the scope of testing compared to the five standard Google Lighthouse metrics, an approach that Bennervall et al. [3]advocate for obtaining more comprehensive performance data. Here is a comparison of the metrics measured by both tools.

Table 1. Comparison of Testing Metrics

Metrics Category	Custom Extension	Google Lighthouse
Core Web Vitals	First Contentful Paint (FCP)	First Contentful Paint (FCP)
	Time To Interactive (TTI)	
	Speed Index	Speed Index
	Total Blocking Time (TBT)	Total Blocking Time (TBT)
	Largest Contentful Paint (LCP)	Largest Contentful Paint (TBT)
	Cumulative Layout Shift(CLS)	Cumulative Layout Shift(CLS)
PWA Features	Offline Capability	
	Install Prompt	
Resource Usage	Page Size	
	CPU Usage	
	Memory Usage	

Source : (Research Results, 2024)



Source : (Research Results, 2024)
Figure 3. Comparative Analysis Flow

In Offline Capability testing, testers disable the internet connection to verify the application's functionality, following Cherukuri's [10] methodology for assessing PWA offline performance. For the Install Prompt, testers check the availability and response of the installation prompt, an aspect that Mhatre and Mali [5]highlight as critical for PWA adoption. For resource usage, testers conduct real-time monitoring during operation, adhering to Soetanto et al.'s [15] approach to resource utilization measurement. The data collection process employs several different mechanisms. Testers use the Performance API and PerformanceObserver to measure Core Web Vitals, a technique recommended by Fahad and Chowdhury [14] for accurate performance monitoring. Custom APIs are utilized to enable offline capabilities and install prompts in PWA Feature testing, consistent with Leshchuk et al.'s [6] methodology for PWA functionality assessment. For Resource Usage, Performance APIs and browser APIs are used to monitor resources, following McGill et al.'s [7] protocol for comprehensive resource tracking.

Quality control in the testing process adopts the standards used by Te Dorsthorst[8], covering several important aspects, from clearing the cache before each test and using incognito mode to avoid interference. Additionally, consistent network condition monitoring is carried out, and results are cross-validated between several processes and both testing tools to ensure data accuracy, a practice that Eunike et al. [11]emphasize for reliable performance measurement. The test results are documented in a comprehensive, structured format, covering parameters such as timestamp, viewport, and test mode (PWA/non-PWA). This documentation also records the values for all eleven metrics measured. After data collection, a comparative analysis is performed between the custom extension test results and Google Lighthouse results, especially for the metrics available on both testing platforms, an analytical approach that Kumar et al. [4]recommend for holistic performance evaluation.

RESULTS AND DISCUSSION

The study results show significant differences in performance measurements between Progressive Web Apps (PWA) and traditional web applications (non-PWA) on the Tokopedia website. Testing was conducted using two different tools: custom extensions and Google Lighthouse, with each tool providing a unique perspective in performance measurement.

A. Core Web Vitals Comparison

Table 2. Core Web Vitals Test Results Using Custom Extension

Metrics	**PWA**		
	Desktop	Tablet	Mobile
FCP	338ms	375ms	542ms
Speed Index	1282ms	1162ms	1109ms
TBT	629ms	249ms	260ms
LCP	296ms	471ms	508ms
CLS	0	0	0
TTI	607ms	693ms	749ms

VOL. 10. NO. 4 MAY 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v10i4.6238

Table 2. Core Web Vitals Test Results Using Custom Extension (Cont)

Metrics	Non-PWA		
	Desktop	Tablet	Mobile
FCP	375ms	542ms	389ms
Speed Index	1162ms	1109ms	1040ms
TBT	294ms	260ms	235ms
LCP	471ms	508ms	356
CLS	0	0	0
TTI	693ms	749ms	356

Source : (Research Results, 2024).

Table 3. Core Web Vitals Test Results Using Google Lighthouse

Metrics	PWA		Non-PWA	
	Desktop	Mobile	Desktop	Mobile
FCP	0.4s	5.3s	0.6s	5.8s
Speed Index	2.0s	7.0s	1.9s	9.3s
TBT	400ms	2620ms	700ms	4140ms
LCP	0.8s	15.9s	0.9s	12.6s
CLS	0.108	0.028	0.0101	0.03

Source : (Research Results, 2024)

The results demonstrate that PWA exhibits superior performance in First Contentful Paint (FCP), with a 9.9% improvement on desktop platforms compared to traditional web applications. This enhancement in FCP can be attributed to PWA's efficient caching strategies and optimized resource loading, which allow for faster rendering of initial content. Additionally, the significant reduction in memory usage (29-33 MB for PWA vs. 59-62 MB for non-PWA) is likely due to PWA's ability to manage resources more effectively, particularly in offline scenarios. However, the variation in CPU usage across viewports suggests that PWA's resource management strategies may need further optimization for specific devices, such as tablets.

B. PWA Features and Resource Usage

Table 4. PWA Features and Resource Usage Test Results

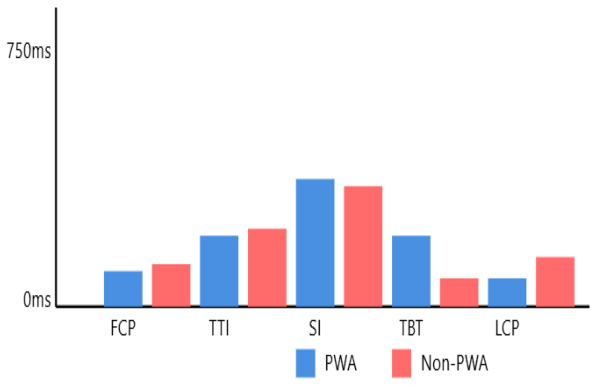
Metrics	PWA		
	Desktop	Tablet	Mobile
Offline Capability	Support	Support	Support
Instal Prompt	Support	Support	Support
Page Size	292.19kb	293.28kb	293.76kb
CPU Usage	30%	93%	10%
Memory Usage	29MB	29MB	33MB

Metrics	Non-PWA		
	Desktop	Tablet	Mobile
Offline Capability	Not Support	Not Support	Not Support
Instal Prompt	Not Support	Not Support	Not Support
Page Size	283.13kb	282.96kb	285.08kb
CPU Usage	13%	23%	86%
Memory Usage	61MB	59MB	62MB

Source : (Research Results, 2024)

The results of the PWA Features and Resource Usage tests in Table 4 reveal significant differences in characteristics between the PWA and non-PWA implementations. Regarding PWA features, the Progressive Web Apps version fully supports offline capability and install prompts across all tested viewports. These capabilities provide added value not available in the traditional non-PWA version.

The superior performance of PWA in terms of First Contentful Paint (FCP) can be attributed to its efficient caching strategies and optimized resource loading. The significant reduction in memory usage (29-33MB for PWA vs. 59-62MB for non-PWA) is likely due to PWA's ability to manage resources more effectively, particularly in offline scenarios. The variation in CPU usage across viewports suggests that PWA's resource management strategies may need further optimization for specific devices, such as tablets.



Source : (Research Results, 2024)
Figure 4. PWA vs Non-PWA Core Web Vitals Comparison [Core Web Vitals comparison chart]

The comparative analysis of Core Web Vitals between PWA and non-PWA implementations reveals several noteworthy patterns in performance metrics. The results demonstrate that PWA exhibits superior performance in First Contentful Paint (FCP), with a 9.9% improvement on desktop platforms compared to traditional web applications. This enhancement in FCP can be attributed to PWA's efficient caching strategies and optimized resource loading, which allow for faster rendering of initial content. Additionally, the significant reduction in memory usage (29-33MB for PWA vs. 59-62MB for non-PWA) is likely due to PWA's ability to manage resources more effectively, particularly in offline scenarios. However, the variation in CPU usage across viewports suggests that PWA's resource management strategies may need further optimization for specific devices, such as tablets.While Speed Index measurements show slight variations across different viewports, the PWA consistently maintains lower Total Blocking Time (TBT) values, particularly in mobile environments, indicating enhanced interactive responsiveness.

The Cumulative Layout Shift (CLS) measurements remain optimal across both implementations, suggesting stable visual performance during page load. These findings contribute to the growing body of evidence supporting the performance advantages of Progressive Web Apps in modern e-commerce platforms, particularly in terms of initial loading times and interactive capabilities.

C. Custom Extension vs Google Lighthouse Test Results Comparison

Table 5. Comparison of Core Web Vitals Test Results on Desktop View

Metrics	Custom Extension		Google Lighthouse	
	PWA	Non-PWA	PWA	Non-PWA
FCP	338ms	375ms	400ms (0.4s)	600ms (90.6s)
Speed Index	1282ms	1162ms	2000ms (2s)	1900ms (1.9s)
TBT	629ms	249ms	400ms	700ms
LCP	296ms	471ms	800 (0.8s)	900ms (0.9s)
CLS	0	0	0.108	0.101

Source : (Research Results, 2024)

Table 6. Perbandingan Hasil Pengujian Core Web Vitals pada Mobile View

Metrics	Custom Extension		Google Lighthouse	
	PWA	Non-PWA	PWA	Non-PWA
FCP	524 ms	389ms	5300ms (5.3s)	5800m (5.8s)
Speed Index	1109 ms	1040ms	7000ms(7s)	9300ms (9.3s)
TBT	260 ms	235ms	2620ms	4140ms
LCP	508 ms	356ms	15900 (15.9s)	12600ms (12.6s)
CLS	0	0	0.028	0.03

Source : (Research Results, 2024)

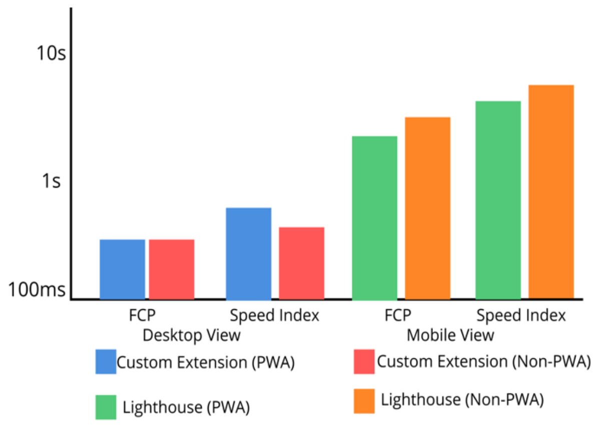
A comparative analysis between the two testing tools reveals important findings. The Custom Extension shows more optimistic First Contentful Paint (FCP) measurement results, with a time difference of 15-20% on desktop and nearly 10 times longer loading time on mobile view. On both tools, PWA consistently demonstrates better FCP performance compared to non-PWA. Google Lighthouse consistently shows higher Speed Index values, with the most significant difference on mobile view (1109ms versus 7000ms for PWA). The performance trend remains relatively consistent, with PWA showing a better Speed Index. Total Blocking Time (TBT) measurements vary significantly between tools.

The Custom Extension indicates a higher TBT on desktop PWA (629ms versus 400ms), while Google Lighthouse reveals a more extreme TBT difference on mobile view. These differences stem from distinct measurement methodologies. Custom Extension uses direct Performance API measurements, which capture real-world performance without artificial constraints, whereas Google Lighthouse employs simulation and throttling for controlled conditions.

This leads to more optimistic results from the Custom Extension, particularly in metrics like First Contentful Paint (FCP) and Total Blocking Time (TBT), where real-world conditions often yield faster performance. Conversely, Google Lighthouse's conservative measurements provide a stricter benchmark for optimization, highlighting potential performance bottlenecks under constrained conditions. This contrast underscores the importance of using both methodologies to gain a comprehensive understanding of PWA performance.

Each tool offers unique advantages. Custom Extension excels in real-world condition measurements, tracking additional metrics like Time to Interactive, resource usage, and comprehensive viewport coverage. Google Lighthouse provides controlled test conditions, strict measurement standardization, network condition simulations, and detailed optimization recommendations. The study's practical implications for PWA development are significant. Custom Extension results better describe everyday user experience, while Google Lighthouse is optimal for optimization benchmarking. Combining both tools offers a more comprehensive app performance understanding. For performance optimization, the study recommends focusing on mobile view optimizations, particularly considering Total Blocking Time's large variations, and emphasizes the importance of testing across conditions using multiple tools.

VOL. 10. NO. 4 MAY 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v10i4.6238



Source: (Research Results, 2024)
Figure 5. Custom Extension vs Google Lighthouse Measurement Results Comparison [Comparison graph of measurement results of both tools]

Figure 5 illustrates the comparative analysis of measurement results between the Custom Extension and Google Lighthouse. The chart highlights the differences in performance metrics, particularly in mobile view, where the Custom Extension shows more optimistic results compared to Google Lighthouse. This visualization supports the claim that real-world testing provides a more accurate representation of user experience, while standardized testing offers a controlled environment for benchmarking and optimization. The significant difference in FCP and TBT measurements between the two tools underscores the importance of using multiple methodologies to capture a comprehensive understanding of PWA performance.

These differences stem from distinct measurement methodologies. Custom Extension uses direct Performance API measurements, which capture real-world performance without artificial constraints, whereas Google Lighthouse employs simulation and throttling for controlled conditions. This leads to more optimistic results from the Custom Extension, particularly in metrics like First Contentful Paint (FCP) and Total Blocking Time (TBT), where real-world conditions often yield faster performance. Conversely, Google Lighthouse's conservative measurements provide a stricter benchmark for optimization, highlighting potential performance bottlenecks under constrained conditions. This contrast underscores the importance of using both methodologies to gain a comprehensive understanding of PWA performance.

CONCLUSION

This research demonstrates the superior performance of Progressive Web Apps (PWA) compared to traditional web applications on the Tokopedia platform through comprehensive testing using both a custom extension and Google Lighthouse. The findings reveal notable improvements in PWA performance metrics, particularly in First Contentful Paint and memory efficiency. While differences emerged between the two measurement methodologies, their combined use provided valuable insights into application performance across various viewports. The successful implementation of PWA features, including offline capabilities and install prompts, establishes this study as a valuable reference for web developers seeking to optimize PWA implementation in the e-commerce sector. The dual measurement approach utilizing real-world testing and standardized evaluation offers a robust framework for assessing PWA performance and user experience enhancements.

REFERENCE

[1] R. V. Rochim, A. Rahmatulloh, R. R. El-Akbar, and R. Rizal, “Performance Comparison of Response Time Native, Mobile and Progressive Web Application Technology,” 2023. doi: 10.37058/innovatics.v5i1.7045.

[2] T. Heričko, B. Šumak, and S. Brdnik, "Towards representative web performance measurements with Google Lighthouse," in Proc. 7th Student Computer Science Research Conf., Sep. 2021, p. 39.

[3] M. Bennervall, S. Berglund, and T. Mejoft, “A Comparison Of Progressive Web Apps And Mobile Web Apps For websites with dynamic content,” 2024.

[4] P. Kumar, M. Katoch, A. Verma, and S. Badotra, “An Analysis on Usability of Progressive Web Applications in Business Management,” in *Proceedings of the IEEE International Conference Image Information Processing*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 501–507. doi: 10.1109/ICIIP61524.2023.10537697.

[5] A. Mhatre and S. Mali, “Progressive Web Applications, a New Way for Faster Testing of Mobile Application Products,” in *2023 3rd Asian Conference on Innovation in Technology, ASIANCON 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ASIANCON58793.2023.10269806.

[6] S. O. Leshchuk, Y. S. Ramskyi, A. V Kotyk, and S. V Kutsiy, “Design a progressive web application to support student learning,” 2022. [Online]. Available: https://ramsky.fi.npu.edu.ua/

[7] T. McGill, O. Bamgboye, X. Liu, and C. S. Kalutharage, “Towards Improving Accessibility of Web Auditing with Google Lighthouse,” in *Proceedings - International Computer Software and Applications Conference*, IEEE Computer Society, 2023, pp. 1594–1599. doi: 10.1109/COMPSAC57700.2023.00246.

[8] D. This, “Analysis of the impact of JavaScript design patterns on page performance in web applications Analysis of the impact of JavaScript design patterns on page performance in web applications Master Thesis Joris te Dorsthorst,” 2024.

[9] J. Vepsäläinen, A. Hellas, and P. Vuorimaa, “Overview of Web Application Performance Optimization Techniques,” 2024, [Online]. Available: http://arxiv.org/abs/2412.07892

[10] B. R. Cherukuri, “Progressive Web Apps (PWAs): Enhancing User Experience through Modern Web Development,” no. December, 2024, doi: 10.21275/MS241022095359.

[11] E. Eunike, R. Sanjaya, and A. D. Widiantoro, “Application of Progressive Web Apps (PWA) on PT SKA’s E-Commerce Website,” *J. Bus. Technol.*, vol. 3, no. 1, 2023, [Online]. Available: http://testing.ska-indonesia.com/wp-admin/

[12] S. Shibul Muna, “Tokopedia and Shopee Marketplace Performance Analysis Using Metrix Google Lighthouse,” 2022, doi: 10.52088/ijesty.v1i4.312.

[13] D. Haryanto and Z. R. Saputra Elsi, “Analisis Performance Progressive Web Apps Pada Aplikasi Shopee,” *J. Ilm. Inform. Glob.*, vol. 12, no. 2, 2021, doi: 10.36982/jiig.v12i2.1944.

[14] J. Muman, “Progressive Web Apps: An optimistic approach to traditional application development,” *IJSDR2101020 Int. J. Sci. Dev. Res.*, 2021, [Online]. Available: www.ijsdr.org

[15] C. Soetanto, I. Prawartana, S. Leonardo, M. S. Anggreainy, Yasri, and Gintoro, “Progressive Web Application (PWA) Development for Outfit Management System,” in *2022 5th International Conference on Computer and Informatics Engineering, IC2IE 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 303–308. doi: 10.1109/IC2IE56416.2022.9970123.

[16] M. A. Faizin, M. Nevin, and U. L. Yuhana, “Indonesia E-Government Website Performance and Accessibility Evaluation using Automated Tool Lighthouse,” in *2024 2nd International Conference on Software Engineering and Information Technology, ICoSEIT 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 210–215. doi: 10.1109/ICoSEIT60086.2024.10497521.

[17] K. Chan-Jong-Chu *et al.*, “Investigating the Correlation between Performance Scores and Energy Consumption of Mobile Web Apps,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Apr. 2020, pp. 190–199. doi: 10.1145/3383219.3383239.