JCSI 36 (2025) 271–277
Received: 13 March 2025
Accepted:29 April 2025

Study of factors affecting the performance of web applications on mobile devices

Badanie czynników wpływających na wydajność aplikacji webowych na urządzeniach mobilnych

Jarosław Królikowski*, Marek Miłosz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The aim of the article is to investigate the factors affecting the performance of Progressive Web Applications (PWAs) on mobile devices. For research purposes, a custom PWA website was created, which served as the primary object of analysis. Various optimization techniques were then applied to the site, including resource minimization, lazy loading, and a distributed content delivery network. The performance analysis of the PWA site was conducted using assessment tools such as PageSpeed Insights, DebugBear, Uptrends, and WebPageTest. The research results were grouped according to several criteria, such as page load time, interactivity, and overall application performance. The obtained results were analyzed after each optimization was implemented. Based on the conducted analysis, conclusions were drawn regarding the impact of the applied optimization techniques on the performance of the PWA. The article emphasizes the importance of continuous optimization and monitoring of applications to ensure high-quality user experiences, as well as presenting an effective alternative method for developing applications for mobile devices.

Keywords: Progressive Web Apps (PWA); Performance; Optimization

Streszczenie

Celem artykułu jest zbadanie czynników wpływających na wydajność progresywnych aplikacji internetowych (PWA) na urządzeniach mobilnych. W celach badawczych stworzono własną stronę internetową PWA, która posłużyła jako podstawowy obiekt analizy. Następnie na stronie zastosowano różnorodne techniki optymalizacji, w tym minimalizację zasobów, leniwe ładowanie oraz rozproszoną sieć zasobów. Analizę wydajności strony PWA przeprowadzono przy użyciu narzędzi oceny, takich jak PageSpeed Insights, DebugBear, Uptrends oraz WebPageTest. Wyniki badań zostały pogrupowane według kilku kryteriów, takich jak czas ładowania strony, interaktywność oraz ogólna wydajność aplikacji. Otrzymane wyniki przeanalizowano po każdej wprowadzonej optymalizacji. Na podstawie przeprowadzonej analizy wyciągnięto wnioski dotyczące wpływu zastosowanych technik optymalizacji na wydajność aplikacji PWA. Artykuł podkreśla znaczenie ciągłej optymalizacji oraz monitorowania aplikacji w celu zapewnienia wysokiej jakości doświadczeń użytkowników, a także prezentuje skuteczną alternatywną metodę tworzenia aplikacji na urządzenia mobilne.

Słowa kluczowe: Progresywne aplikacje internetowe (PWA); wydajność; optymalizacja

*Corresponding author

Email address: kontakt@jaroslawkrolikowski.pl (J. Królikowski)

1. Introduction

For several decades, we have been observing the dynamic development of mobile technologies, which have revolutionized the way we use the Internet. Mobile phones are owned by 70.3% of the population [1], and according to the "Digital 2024" report [2], over 5.35 billion people worldwide have access to the Internet, and 5.16 billion use mobile phones. These numbers highlight the importance of mobile devices in everyday life.

There is a constant increase in the popularity and number of mobile phone users year by year. This is indicated by studies on the number of phone subscription subscribers and data for individual countries regarding the increase in the number of mobile phone users [3, 4, 5, 6, 7]. Screen time increased by 30% from 2013 to 2023 [8] [9], highlighting the growing role of technology in our lives. Already in the 20th century, the need for portable computing devices was recognized, leading to the creation of the first calculators, palmtops, and personal digital assistants (PDA). Further research gradually increased the capabilities of phones, and by 1996, access to the first website from mobile phones via the GSM network was enabled [10].

The launch of the first Apple phone in 2007 [11] and the first Android phone in 2008 [12] significantly impacted the development of mobile technologies. The introduction of touchscreens blurred the lines between phones and computers, and smartphones began offering functionalities comparable to personal computers. However, the development of mobile technologies comes with new challenges, such as limited battery capacity and the variety of operating systems, which requires adapting software to the specifics of each. Mobile apps must be designed considering touch navigation and the varied conditions of Internet usage.

Therefore, one approach to software development is to create a Progressive Web Apps (PWA), which allows for reducing the number of tests and focusing on a single platform.

This article focuses on studying the factors affecting the performance of PWAs on mobile devices. PWAs combine the advantages of websites and mobile apps, making them an increasingly popular solution. The work includes optimizing web application code and techniques for improving loading time and smooth operation, which are crucial for positive user experiences. The research focuses on optimization techniques such as resource minimization, lazy loading, and the use of content delivery networks, which can be a valuable source of knowledge for developers and designers.

2. Literature Review

In recent years, websites have evolved from simple static text pages to interactive web applications. With the growing interest in mobile devices and the increasing number of users, websites optimized for mobile devices have become essential. Website performance optimization is crucial for smooth operation on mobile devices, as long waiting times impact user experience and business metrics like conversion rates [13]. Performance affects not only user experience but also the success of websites, particularly in e-Commerce [14].

Web application performance is broadly defined and typically measured by load time, response time, and resource usage stability [15, 16, 17]. In the context of mobile devices, minimizing delays and reducing resource consumption is essential [18, 19].

Slow application loading is often caused by resources like images and scripts. Therefore, one frequently discussed performance optimization element is reducing image and video sizes [16, 20]. Media optimization studies show that lowering image resolution or using formats like WebP can reduce their size by 30-50% while maintaining sufficient quality for most mobile devices [21]. Similar effects are achieved by minifying HTML, CSS, and JavaScript, reducing their size by approximately 20% [22].

Network bandwidth and latency significantly impact web application performance, especially for users on 3G or LTE networks with poor coverage [23]. Studies show that low-bandwidth networks can double resource loading times compared to high-speed Wi-Fi [24]. Network optimization includes using Content Delivery Networks (CDNs) that distribute resources across multiple geographic locations, resulting in faster loading for users worldwide [25, 26].

PWA gains popularity as an alternative to native apps due to better responsiveness and offline capabilities. Comparative studies show that well-optimized PWAs load 15-20% faster than traditional mobile sites, thanks to asynchronous resource loading and browser caching [17, 27]. Performance gains, however, depend on implementation and optimization of individual components [28]. Given the increasing importance of mobile technology and the growing popularity of PWAs, further research on their optimization is needed. Better understanding of ways to enhance PWA performance on mobile devices will help create applications that offer smooth, responsive, and satisfying experiences for mobile users.

3. Purpose of the Study

The purpose of this study is to examine the performance and optimization of a PWA. The website was created and then modified to investigate the potential impact of optimization techniques on its performance.
The study is divided into several parts:

- Selection of tools for analyzing website performance.
- Creation of a website and optimization using selected techniques.
- Examination of the website using the presented tools.
- Analysis of the research results.
- Conclusions.

4. Research Methodology

4.1. Research Tools

To create a PWA and measure its performance, the following tools were used:

- **Visual Studio** – an integrated development environment that allows for creating web applications in various programming languages, such as HTML, CSS, and JavaScript. With rich features like IntelliSense and debugging, Visual Studio simplifies the process of writing and testing code.
- **PageSpeed Insights** – a tool provided by Google that evaluates the performance of a website on both mobile and desktop devices. It analyzes various aspects such as load time, image optimization, and cache usage, providing detailed recommendations for improving performance.
- **Uptrends** – a platform for monitoring website performance that allows tracking load times and site availability from various global locations. Uptrends also offers real-time performance analysis, enabling quick identification of issues.
- **DebugBear** – a website performance analysis tool that enables monitoring and comparing performance results over time. DebugBear provides detailed reports on load times, interactivity, and visual stability, allowing for the optimization of user experiences.
- **WebPageTest** – an advanced tool for testing website performance that enables testing from various locations and browsers. WebPageTest provides detailed data on load times, resource analysis, and charts that help identify performance bottlenecks.

4.2. Creation of the Website

A simple website was created using HTML, CSS, and JavaScript technologies, which allowed for the development of an interactive and responsive user interface. The Bootstrap framework was used for rapid and aesthetically pleasing design, facilitating the creation of layouts and styles. Additionally, the API of the National Bank of Poland (NBP) was utilized to obtain real-time exchange rate data, enabling dynamic content.

4.3. Research Scenarios

To analyze the impact of performance optimization on the website, five different scenarios were conducted, focusing on optimizing specific elements of the site. The website's performance was verified using key metrics:

- **First Contentful Paint (FCP)**: Measures the time it takes for any content element to be rendered. It signals the beginning of the visual loading progress [29] and helps monitor loading progress and detect early delays.
- **Largest Contentful Paint (LCP)**: Measures the time it takes for the largest content element to become visible. Studies confirm that faster result correlates with lower bounce rates and higher conversion rates [30].
- **Speed Index (SI)**: Measures the speed of the visual loading progress [31].
- **Time to First Byte (TTFB)**: Measures server response time. A high result negatively impacts the overall page load time [32].
- **Download Size**: total amount of data, measured in bytes, that must be transferred over the network to load a web page in the browser.
- **Score**: A synthetic performance score aggregating multiple metrics, with an emphasis on Core Web Vitals [33]. A higher score indicates better overall performance.

The research scenarios were as follows:

- **S1:** *Minification of scripts and styles, WebP format instead of PNG and JPG* – JavaScript and CSS files were minified by removing comments, special characters, and spaces, which reduced their size and accelerated loading times. Additionally, using the WebP format for images instead of traditional PNG and JPG formats further reduced the size of graphic resources without compromising the visual quality of the images.
- **S2:** *JSON Data Format Instead of XML* – In this scenario, the format of transmitted data was changed from XML to JSON.
- **S3:** *Asynchronous Script Loading* – Implementing asynchronous loading of JavaScript scripts allowed for background resource loading.
- **S4:** *Lazy Loading* – Applying the lazy loading technique for images and other resources ensured that only elements visible on the user's screen were loaded.
- **S5:** *Use of CDN* –The use of a CDN enabled the distribution of resources across multiple geographic locations and optimized resource loading by leveraging caching functionality.

Each of these scenarios aimed to improve the website's performance, and their effects were thoroughly analyzed using the previously mentioned metrics, providing valuable insights into website optimization.

5. Results

In this chapter, the results of the analysis of the implemented optimization techniques are presented. Tables 1 - 6 illustrate the results of the analysis for each case and the average of all analyzers. The use of multiple analyzers allowed for more accurate results, which increases the reliability of the conducted research.

Table 1: Performance Analysis of the Baseline Version of the Website

Analyzers / Metrics	PageSpeed	UpTrends	DebugBear	WebPage Test
FCP [s]	1.21	1.02	3.05	3.85
LCP [s]	15.32	1.02	3.05	3.86
SI [s]	1.24	X	3.05	3.88
TTFB [s]	X	0.34	0.80	1.27
Download size [MB]	X	3.01	2.93	2.99
Score [points]	74.83	X	87.13	X

Table 2: Performance Analysis After Data Compression

Analyzers / Metrics	PageSpeed	UpTrends	DebugBear	WebPage Test
FCP [s]	1.23	0.57	2.74	2.81
LCP [s]	3.96	0.57	2.74	2.85
SI [s]	1.23	X	2.73	2.87
TTFB [s]	X	0.22	0.89	1.04
Download size [MB]	X	0.58	0.62	0.62
Score [points]	87.6	X	90.73	X

Table 3: Performance Analysis After JSON Format Implementation

Analyzers / Metrics	PageSpeed	UpTrends	DebugBear	WebPage Test
FCP [s]	1.21	1.07	3.32	3.35
LCP [s]	14.22	1.07	3.32	3.38
SI [s]	1.20	X	3.31	3.41
TTFB [s]	X	0.36	0.88	1.02
Download size [MB]	X	3.01	2.93	2.99
Score [points]	75.00	X	84.77	X

Table 4: Performance Analysis After Implementing Asynchronous Script Loading

Analyzers Metrics	PageSpeed	UpTrends	DebugBear	WebPage Test
FCP [s]	1.25	1.41	2.23	2.57
LCP [s]	1.71	1.41	2.24	2.67
SI [s]	1.30	X	2.23	2.69
TTFB [s]	X	0.45	0.91	1.02
Download size [MB]	X	3.01	2.93	2.93
Score [points]	100.00	X	96.00	X

Table 5: Performance Analysis After Implementing Lazy Loading

Analyzers Metrics	PageSpeed	UpTrends	DebugBear	WebPage Test
FCP [s]	1.15	0.51	2.14	3.26
LCP [s]	14.68	0.53	2.14	3.26
SI [s]	2.58	X	2.27	3.30
TTFB [s]	X	0.25	0.89	1.03
Download size [MB]	X	0.77	2.93	2.99
Score [points]	74.23	X	93.97	X

Table 6: Performance Analysis After Implementing CDN.

Analyzers Metrics	PageSpeed	UpTrends	DebugBear	WebPage Test
FCP [s]	2.05	0.99	2.61	3,43
LCP [s]	4.16	0.99	2.61	3,46
SI [s]	2.05		2.61	3,47
TTFB [s]	X	0.30	0.90	1,01
Download size [MB]	X	0.59	0.57	0.62
Score [points]	85.13	X	92.70	X

Based on the data presented in the tables, an average was calculated for each metric. Then, charts (Figures 1-5) were created to show the data for the baseline website and after applying the performance optimization techniques. These visualizations allow for easy comparison of the effects of the introduced optimization and its impact on key performance metrics. For the metrics on the left side, a lower value indicates a better result. For the metric on the right side, score, a higher value indicates a better result. The analysis of the results will enable drawing conclusions regarding the effectiveness of the applied optimization methods.

Figure 1 presents the results obtained after implementing the data compression technique. The introduction of minification for JS and CSS files, along with image compression to the WebP format, significantly improved the website's performance

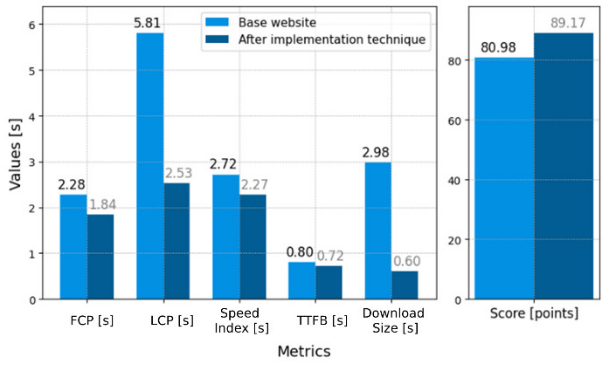

Figure 1: Chart showing the differences in analyzer results for the baseline website and after implementing the data compression technique.

FCP time decreased from 2.28s to 1.84s, accelerating the initial loading of the page. LCP time improved from 5.81s to 2.53s, reducing the perceived loading time for users. Speed Index dropped from 2.72s to 2.27s, indicating a smoother visual loading experience. TTFB slightly improved from 0.8s to 0.72s, contributing to the overall faster loading time. Download Size was reduced from 2.98MB to 0.6MB, greatly speeding up the loading process, especially for users with slower internet connections. The overall performance score increased from 80.98 to 89.17 points, confirming the effectiveness of the optimization techniques applied.

Figure 2 presents the results obtained after implementing the JSON format. After applying this optimization technique, some performance metrics improved, while others changed only slightly.

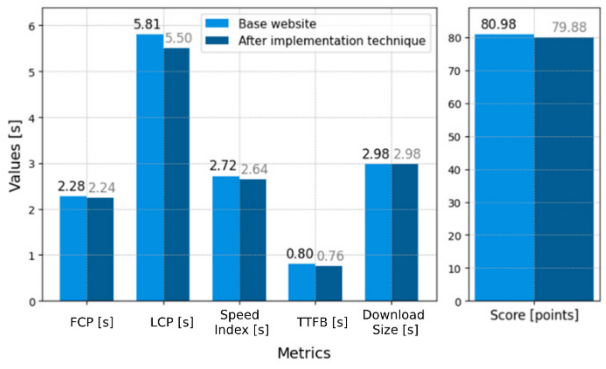

Figure 2: Chart showing the differences in analyzer results for the baseline website and after implementing the JSON format.

FCP slightly decreased from 2.28s to 2.24s, indicating a minor acceleration in displaying the first content element on the page. LCP improved from 5.81s to 5.50s, meaning the largest content element loaded faster. Speed Index improved from 2.72s to 2.64s, suggesting a smoother visual loading of the page. TTFB slightly improved from 0.8s to 0.77s, indicating a slightly faster server response.

Download Size remained unchanged at 2.98MB, suggesting that the implementation of the JSON format did not affect the reduction in the size of the downloaded data. The overall performance score slightly decreased from 80.98 to 79.88 points, which may indicate a small negative impact of this optimization technique on other aspects of the website's performance.

Figure 3 presents the results obtained after implementing the asynchronous script loading technique. The introduction of this technique significantly improved the website's performance.

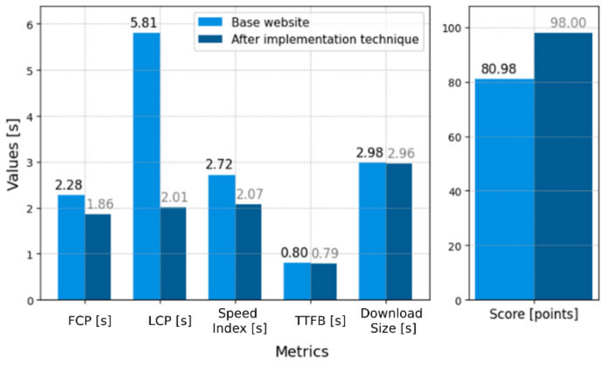

Figure 3: Chart showing the differences in analyzer results for the baseline website and after implementing asynchronous script loading.

FCP decreased from 2.28s to 1.86s, meaning that the first content element on the page is displayed faster. LCP reduced from 5.81s to 2.01s, indicating faster loading of the largest content element.Speed Index improved from 2.72s to 2.07s, suggesting a smoother visual loading experience. TTFB slightly improved from 0.8s to 0.79s, indicating a marginally faster server response.

Download Size decreased slightly from 2.98MB to 2.96MB, indicating a minor reduction in data size. Most importantly, the overall performance score increased significantly from 80.98 to 98.00 points, confirming the effectiveness of the asynchronous script loading technique in improving website performance.

Figure 4 presents the results obtained after implementing the lazy loading technique. This technique led to several improvements in the website's performance metrics.

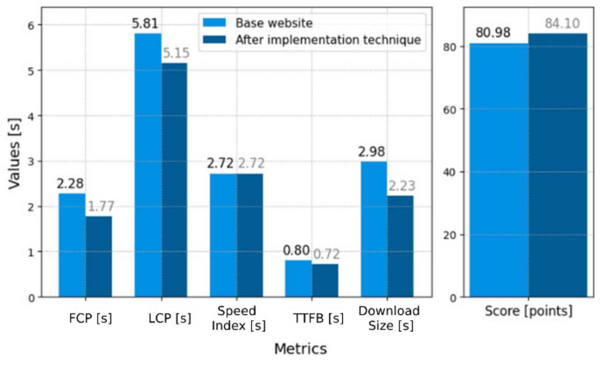

Figure 4: Chart showing the differences in analyzer results for the baseline website and after implementing Lazy Loading.

FCP decreased from 2.28s to 1.77s, indicating faster initial content rendering. LCP also improved, reducing from 5.81s to 5.15s, which means that the most significant content element loaded more quickly.

Speed Index remained the same at 2.72s, suggesting that the overall visual loading experience did not change significantly. TTFB improved slightly from 0.8s to 0.72s, indicating a marginally faster server response. Download Size decreased significantly from 2.98MB to 2.23MB, indicating that lazy loading effectively reduced the amount of data loaded initially. This reduction in download size can be particularly beneficial for users with slower internet connections, as it speeds up the overall page loading time.

The overall performance score increased from 80.98 to 84.10 points, confirming the effectiveness of the lazy loading technique in enhancing the website's performance.

Figure 5 presents the results obtained after implementing the CDN technique. This technique resulted in notable improvements in various performance metrics of the website.

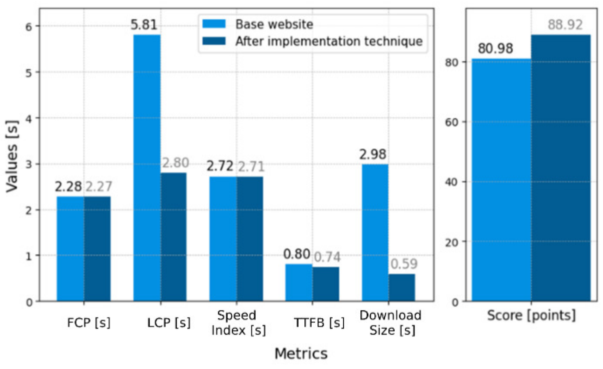

Figure 5: Chart showing the differences in analyzer results for the baseline website and after implementing the content delivery network.

FCP slightly decreased from 2.28s to 2.27s, indicating a marginally faster initial content rendering. LCP significantly improved from 5.81s to 2.8s, meaning that the most substantial content element loaded much more quickly. Speed Index showed a minor improvement from 2.72s to 2.71s, suggesting a smoother visual loading experience. TTFB improved slightly from 0.8s to 0.74s, indicating a faster server response.

Download Size saw a dramatic reduction from 2.98MB to 0.59MB, showcasing the effectiveness of the CDN in minimizing the size of data transferred. This reduction is particularly beneficial for users with slower internet connections, as it considerably speeds up the overall page loading time. The overall performance score increased significantly from 80.98 to 88.92 points, confirming the effectiveness of the CDN technique in enhancing the website's performance.

6. Conclusions

The aim of this article was to investigate and optimize the performance of PWA on mobile devices. A PWA website was created, and various optimization techniques were applied, such as resource compression, lazy loading, data format changes, CDN, and asynchronous script loading. Performance analysis was conducted using tools like PageSpeed Insights, DebugBear, Uptrends, and WebPageTest, measuring key performance metrics such as loading time, responsiveness, and visual stability.

The research results showed that each optimization technique had both positive and negative impacts on the website's performance. For instance, using a CDN significantly reduced the LCP and the size of downloaded resources, which is crucial for mobile users with limited data. However, some cases observed a decline in the FCP metric, meaning initial visual elements loaded slower.

Similarly, the Lazy Loading technique, which delays the loading of non-critical resources, contributed to reducing the loading time and data usage.

The analysis revealed that web application performance optimization is a process requiring consideration of many factors and trade-offs. A technique improving one performance aspect might not be beneficial for others. For example, minifying JavaScript and CSS code and compressing resources reduce file sizes and speed up downloads but may increase CPU load during parsing, especially on older mobile devices.

The conducted research opens avenues for further analysis and experiments. Future studies could investigate the combined effects of various optimization techniques on PWA performance and analyze performance based on network conditions and mobile device specifications. Additionally, it would be essential to study the impact of optimizations on user experiences, such as engagement and conversion metrics.

References

[1] Digital 2024 October Global Statshot Report, DataReportal, https://datareportal.com/reports/digital-2024-october-global-statshot, [05.03.2025].

[2] Internet use in 2024, DataReportal, https://datareportal.com/reports/digital-2024-deep-dive-the-state-of-internet-adoption, [05.03.2025].

[3] K. Chan-Jong-Chu, T. Islam, M. M. Exposito, S. Sheombar, C. Valladares, O. Philippot, E. M. Grua, I. Malavolta, Investigating the correlation between performance scores and energy consumption of mobile web apps, *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering* (2020) 190–199, https://doi.org/10.1145/3383219.3383239.

[4] Fixed and mobile-telephone subscriptions, Global and Regional ICT Data, https://stats.areppim.com/stats/stats_mobilexpenetr.htm, [05.03.2025].

[5] N. T. Krell, S. A. Giroux, Z. Guido, C. Hannah, S. E. Lopus, K. K. Caylor, T. P. Evans, Smallholder farmers' use of mobile phone services in central Kenya, Climate and Development 13(3) (2020) 215–227, https://doi.org/10.1080/17565529.2020.1748847.

[6] Mobile Fact Sheet, Pew Research Center, https://www.pewresearch.org/internet/fact-sheet/mobile, [05.03.2025].

[7] Percentage of households with mobile phones in the United Kingdom (UK) from 1996 to 2022, Statista, https://www.statista.com/statistics/289167/mobile-phone-penetration-in-the-uk, [05.03.2025].

[8] B. Pourghassemi, A. A. Sani, A. Chandramowlishwaran, What-if analysis of page load time in web browsers using causal profiling, Proceedings of the ACM on Measurement and Analysis of Computing Systems 3(2) (2019) 1-23, https://doi.org/10.1145/3341617.3326142.

[9] Time Spent Using Smartphones (2024 Statistic), Exploding Topics, https://explodingtopics.com/blog/smartphone-usage-stats, [05.03.2025].

[10] Mobile Web, Wikipedia, https://en.wikipedia.org/wiki/Mobile_web, [05.03.2025].

[11] iPhone (1st generation), Wikipedia, https://en.wikipedia.org/wiki/IPhone_(1st_generation), [05.03.2025].

[12] Android version history, Wikipedia, https://en.wikipedia.org/wiki/Android_version_history, [05.03.2025].

[13] Conversion – definition, Google Ads, https://support.google.com/google-ads/answer/6365, [05.03.2025].

[14] M. Mnyakin, Investigating the impacts of AR, AI, and website optimization on ecommerce sales growth, ResearchBerg Rev. Sci. Technol. 3(1) (2020) 116–130.

[15] I. Malavolta, K. Chinnappan, L. Jasmontas, S. Gupta, K. A. K. Soltany, Evaluating the impact of caching on the energy consumption and performance of progressive web apps, Proceedings of the 7th IEEE/ACM on International Conference on Mobile Software Engineering and Systems (MOBILESoft), Seoul, Republic of Korea (2020) 109–119, https://doi.org/10.1145/3387905.3388593.

[16] F. S. Tahirshah, Comparison between Progressive Web App and Regular Web App, Bachelor's thesis, Blekinge tekniska högskola, Karlskrona, 2019.

[17] P. Setialana, M. Ardiansyah, N. Suparmanto, Development and performance analysis of the Gunungkidul cultural potential application based on progressive web apps, J. Eng. Appl. Technol. 2(1) (2021), https://doi.org/10.21831/jeatech.v2i1.39525.

[18] R. Fransson, A. Driaguine, Comparing progressive web applications with native Android applications: An evaluation of performance when it comes to response time, Bachelor's thesis, Linnaeus University, Växjö, 2017.

[19] A. Nilsson, Performance and feature support of Progressive Web Applications: A performance and available feature comparison between Progressive Web Applications, React Native applications and native iOS applications, (2022).

[20] B. V. Rajeev, K. Bakula, A developer's insights into performance optimizations for mobile web apps, IEEE International Advance Computing Conference (IACC) (2015) 671-675, https://doi.org/10.1109/iadcc.2015.7154791.

[21] L. Butler, M. Rege, Building an analytical model to predict workforce outcomes in medical education, Issues Inf. Syst. 21(2) (2020) 229–237, https://doi.org/10.48009/2_iis_2020_229-237.

[22] K. K. Singh, P. Kumar, Optimizing the performance of mobile web application, Int. J. Eng. Res. Gen. Sci. 3(1) (2015) 849.

[23] D.-J. Rensema, The current state of Progressive Web Apps: A study on the performance, compatibility, consistency, security and privacy, and user and business impact of progressive web apps, Master's thesis, Karlstad University, Karlstad, 2020.

[24] M. Rajiullah, A. Lutu, A. S. Khatouni, M.-R. Fida, M. Mellia, A. Brunstrom, V. Mancuso, Web experience in mobile networks: Lessons from two million page visits, The World Wide Web Conf. (WWW '19) (2019) 1532–1543, https://doi.org/10.1145/3308558.3313606.

[25] R. Rochim, A. Rahmatulloh, R. El-Akbar, R. Rizal, Performance comparison of response time native, mobile and progressive web application technology, Innovation in Research of Informatics (INNOVATICS) (2023), https://doi.org/10.37058/innovatics.v5i1.7045.

[26] T. Everts, Rules for mobile performance optimization, Commun. ACM 56(8) (2013) 52–59, https://doi.org/10.1145/2492007.2492024.

[27] P. Ling, Based on web application front-end performance optimization, Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology (2011) 234-237, https://doi.org/10.1109/emeit.2011.6022862.

[28] P. Panduwika, S. Solehatin, Performance measurement implementation on the smart fisheries village website using PageSpeed Insights, J. Soft Comput. Explor. 5(2) (2024) 161–172, https://doi.org/10.52465/joscex.v5i2.363.

[29] First Contentful Paint (FCP), Google Web Developer documentation, https://web.dev/fcp, [05.03.2025].

[30] How QuintoAndar increased conversion rates and pages per session by improving page performance, Google Web Developer documentation, https://web.dev/case-studies/quintoandar, [05.03.2025].

[31] Speed Index, WebPageTest documentation, https://docs.webpagetest.org/metrics/speed-index, [05.03.2025].

[32] Time to First Byte (TTFB), Google Web Developer documentation, https://web.dev/ttfb, [05.03.2025].

[33] How to interpret your PageSpeed Insights report, Google Web Developer documentation, https://web.dev/articles/speed-insights-basics, [05.03.2025].