Published *online* on the journal's web page : http://innovatics.unsil.ac.id

Innovation in Research of Informatics (INNOVATICS)

| ISSN (Online) 2656-8993 |

Performance Comparison of Response Time Native, Mobile and Progressive Web Application Technology

Rachma Verina Rochim[1], Alam Rahmatulloh[2], R Reza El Akbar[3], Randi Rizal[4]

[1,2,3,4]*Department of Informatics, Siliwangi University, Jl. Siliwangi No. 24 Kota Tasikmalaya 46115, Indonesia*

[1] *rachmaverina@gmail.com,* [2]*alam@unsil.ac.id,* [3]*reza@unsil.ac.id,* [4]*randirizal@unsil.ac.id*

ARTICLE INFORMATION

Article History:
Received: May 15, 2023
Last Revision: May 21, 2023
Published *Online*: May 23, 2023

KEYWORDS

Chrome DevTool,
GTMetrix,
Lighthouse,
Mobile Web,
Progressive Web App

CORRESPONDENCE

Phone: 085321122010
E-mail: rachma@gmail.com

ABSTRACT

The development of technology in web-based applications is growing, this creates new problems. The web technology that is currently being discussed is Progressive Web Application (PWA) but is the PWA's performance better than the previous technology. This research is about measuring the performance of the Native Web, Mobile Web and PWA using three testing tools, namely GTMetrix, Lighthouse, and Chrome DevTool. The results of this study show how to measure the performance of a Progressive Web Application (PWA), where PWA can beat the performance of Native Web and Mobile Web if a web page is tested more than once. Test results on the Progressive Web Application (PWA), the minimum number of page files (home) is 217 kB with page loading time of 638 ms, while the medium page (about) is 431 kB with page loading time of 646 ms, and when accessing heavy pages (news) with a size of 41700 kB the page load time is 532 ms.

1. INTRODUCTION

A website is a page that is static or dynamic which contains information in the form of a collection of text, images, videos and even animations. In accessing the website, an internet network is needed which is opened through an explorer or commonly called a web browser [1]. Both static and dynamic are characteristics of the Native Web. Native Web is an ordinary web where files are written as plain text files, which are arranged and combined in such a way with HTML or XHTML-based instructions, which are sometimes also included with various scripting languages. This web file will later be translated by the browser engine and displayed as a page / website that we normally see [2].

Website creation technology develops continuously not only static or dynamic, but websites that are responsive with a better web appearance that is adjusted to the screen size of a device. In 2016 almost 60% of searches were carried out via mobile devices [3]. The mobile web is one of the best choices because the mobile web is a web with performance and features that don't make it difficult to access a web, it's just that the mobile web must be accessed via an internet network and cannot be accessed if there is no internet network available.

The need for technology that can overcome the lack of internet network availability. The Minister of Communication and Informatics stated that Indonesia is still lacking in terms of internet network connection, the 3T (Forefront, Outermost, and Disadvantaged) areas where most of them are not connected to the internet network are in the eastern part of Indonesia [4].

Progressive Web App (PWA) is an idea first endorsed by Google engineer Alex Russell in June 2015 [5]. Progressive Web App (PWA) is a term for web-based applications that use the latest web technologies, PWAs are actually just regular web-based applications but take advantage of modern browsing features to appear as if they were native applications [6]. PWA not only allows cross-platform development across websites, but also provides features such as background synchronization, offline support, home screen installation [7] for various devices. The technological concept brought by PWA and allows it to be together, namely Service Worker, AppShell, and Web

App Manifest [8]. The Service Worker itself is utilized in managing cache on the website.

This statement is the motivation to investigate how to compare and measure the performance of Progressive Web App, Native Web as well as Mobile Web in terms of the speed parameters of accessing web pages, the number of page files used and other optimizations using open-source tools GTMetrix, Lighthouse and Chrome DevTools.

2. RELATED WORK

In the news portal application, PhoneGap technology is implemented but it is often left behind when there are new features, the need for modern technology that can optimize the use of news portal applications [9].

Modern Progressive Web App technology is compared to Native Web performance and the research results show that PWA is superior in terms of performance compared to Native Web using GTMetrix tools [10]. However, the comparison of the web used is a different web and even an existing web [2].

The modern technology of the Progressive Web App is also compared to the Mobile Web to provide an assumption that the larger the file size of the web page, the better the performance of the progressive web app compared to the mobile web. However, it is not known to what extent a web page file affects the performance of progressive web apps [4]. In the application of the Progressive Web App that is inserted the service worker can cache the data, but this research caching only 500 JSON data [6].

Progressive Web App testing using response time parameters has been carried out only on hardware devices, namely cameras and geolocation [11].

The research conducted [12] tested PWA performance twice using Lighthouse Tools, but only tested on performance, there were no recommendations to improve performance.

3. METHODOLOGY

The research method consists of six stages which can be seen in Figure 1.

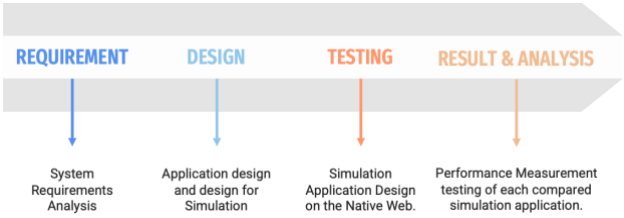

Figure 1. Methodology

3.1 System Requirement

Analysis of system requirements is a stage in preparing all needs in research, obtaining data is the most needed thing. Data collection is carried out by utilizing the API from newsapi.org, based on the data obtained, there is a needs analysis process in accordance with the data. Analyzing software and hardware requirements is also needed in research, as well as identifying problems that can describe the core problems of research.

Hardware and software requirements in making this simulation application can be seen in Table 1 and Table 2.

Table 1. Hardware Requirement

No.	Component	Spesification
1.	*Processor*	A8-6410 4 Cores 2.0 GHz
2.	*Memory*	RAM 4 GB
3.	VGA	VGA 2 GB

Table 2. Software Requirement

No.	Component	Spesification
1.	Sistem Operasi	Windows 10 64 bit
2.	*Web Server*	Apache
3.	*Database*	MariaDB
4.	Programming	***PHP 7.3.0*** and Javascript
5.	*Web Browser*	Google Chrome
6.	*Text Editor*	Visual Studio Code
7.	*Tools* Testing	GTMetrix., Lighthouse dan Chrome DevTools.

3.2 Design Apps Simulation

Simulation Application Design on the Native Web, namely the selection of software modeling, application interface design, and implementation. The stages of designing a simulation application on the Mobile Web are the same as on the Native Web, namely the selection of software modeling, application interface design, and implementation.

While the design of the simulation application on the Progressive Web App includes how to form the PWA page framework interface based on the AppShell guidelines, create a manifest.json file to allow users to install simulation applications, and register service workers so that applications can run offline.

3.3 Testing

Figure numbers and titles are written in the center alignment of the column. Figure numbers are written according to the sequence using Arabic numerals. The title of the image is written at the bottom of the image (effect font: small caps), except for conjunctions and prepositions. The title of the image uses a font size of 8 (eight). Images may not exceed the margin limit of each column, unless the size of a large image is not sufficient in 1 column, it can traverse 2 columns.

Simulation application testing uses 3 tools, namely GTMetrix, Lighthouse Tools, and Chrome DevTools. Stages of testing using GTMetrix are carried out to measure the performance of each simulation application compared to the parameters Page Speed, YSlow, Fully Loaded Time, Total Page Site and number of Requests. Testing using Lighthouse aims to audit performance comparisons in each simulation application with parameters Performance, Accessibility, Best Practices, SEO, PWA and recommendations for improving performance in PWA simulation applications.

Stages of testing are carried out using Chrome DevTool, namely, to test the file size of a page which can affect the performance of the Progressive Web App with the number of page file sizes in stages or periodically, by utilizing the Network Panel which will be tested from the parameters request, transferred, resource, finish, DOMContentLoaded, Load.

3.4 Result and Analysis

At this stage it shows the results of testing the performance measurements of each compared simulation application.

4. RESULT AND DISCUSSION

Data collection was obtained from news data on newsapi.org as many as 539 data in the form of JSON, the newsapi.org API Key is needed in this data collection. Then the data is decoded or converted into an array and entered a database that has previously tidied up its attributes. The 539-news data came from 5 news categories, namely "corona", "covid", "wfh", "Indonesia", and "informatics". The news data is applied to each simulation application which will later be used to measure performance or see the performance of each application being tested.

4.1 Design Apps Simulation

Figure 2 is the FlowChart of the News Portal simulation application, there are 2 terminal symbols indicating "Start" and "Finish". The first process is that the user accesses the simulation application, the first page displayed is the homepage. The homepage display uses the offline connector symbol because there are two processes that can be selected, namely the News menu and the About menu.

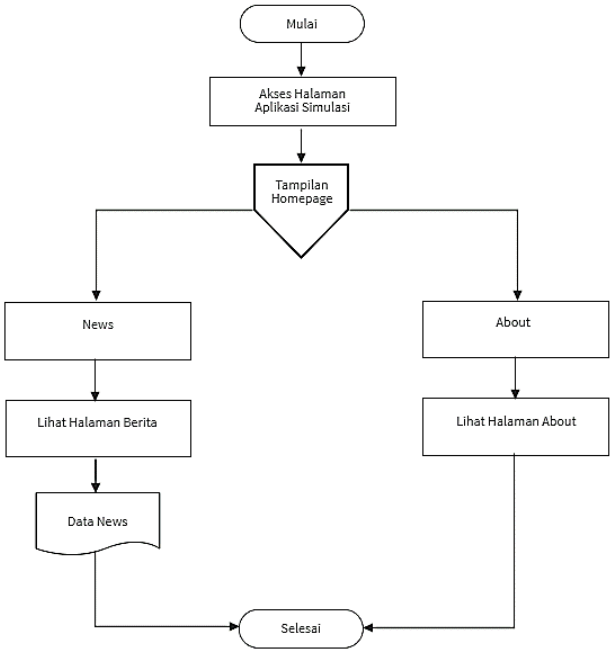

Figure 2. Flowchart Apps Simulation

In the News process, load the news content and display the process for View News Page. This news content is called from the Data News database. While the About process displays an about page which contains information about the simulation application maker.

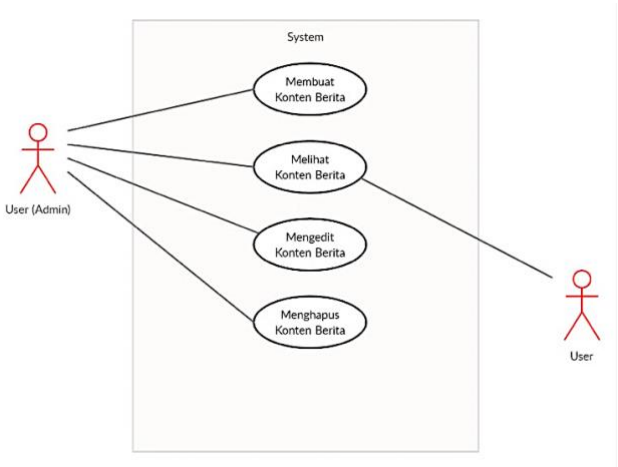

Figure 3. Use Case Apps Simulation

Figure 3 is a simulation application use case where the actor is marked with a user (admin) and a non-admin user. The user (admin) performs the process of creating news content, viewing news content, editing news content, and deleting news content. Meanwhile, ordinary users or users can only view news content.

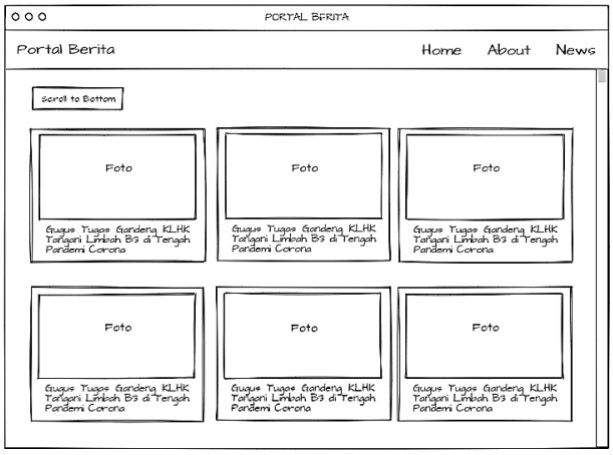

Figure 4. Desktop Interface Menu News on Native Web

Figure 4 is the display interface of the News menu on the Native Web which contains news content in the form of images and text. The design of a Native Web simulation application describes the design of a simple Native Web simulation application that is displayed when opened on a desktop or via a mobile device. However, when a native web application is accessed via a mobile device, the display will not be neat or will not fit the screen size of the mobile device because the native web design is not responsive.

The simulation application interface on the mobile web is not much different from the Native Web application, it's just that the layout or appearance of each page will be in the middle when accessing via a desktop because the display design is specifically for mobile devices and is designed responsively.

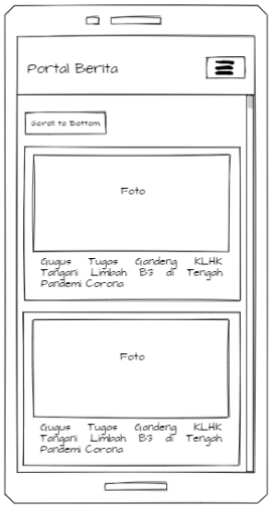

Figure 5. Desktop Interface Menu News on Mobile Web

Figure 5 is the display of the mobile interface from the Home, About, and News menus on the Native Web which contains news content in the form of images and text.

A. Application Model based on App Shell Architecture

The design of the PWA simulation application is built based on the AppShell architecture, where the AppShell architecture is an important component in the main display of the PWA application, namely the user interface display which contains HTML, CSS, and JavaScript scripts that aim to make this PWA application reliable. AppShell is divided into two parts, namely page outline and page content.

The page framework section of this PWA simulation application is divided into a minimal page (homepage), medium page (about), and heavy page (news page).

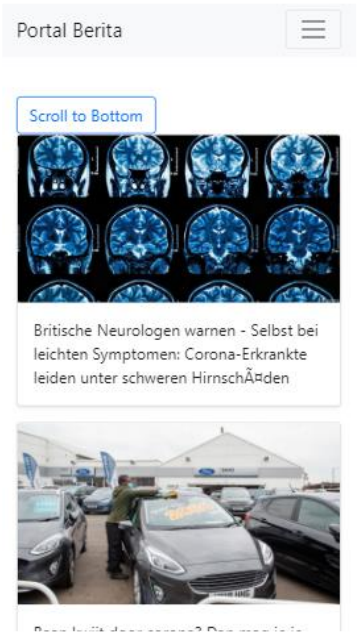

Figure 6. News Page

Figure 6 shows the display of the heavy page, namely the News page which contains 539 news data. Meanwhile, the page content consists of text blocks and image blocks.

B. Create manifest.Json

Before creating the manifest.json file, an image for the simulation application icon has been prepared, which is a minimum size of 512 x 512 pixels. Manifest.json contains brief information or data such as icon, application name, appearance, application display orientation, etc. to set the appearance, add applications to A2HS, set the color of the splash screen and the color on the taskbar. The properties in the manifest.json used in this PWA simulation application are “name”, “short_name”, “theme_color”, “background_color”, “display”, “standalone”, “orientation”, “scope”, “start_url”, "icons", more details can be seen in Table 3.

Tabel 3. Fungsi dari properti manifest.json

No.	The Property Use	Function
1.	name	Naming when requesting to install the application.
2.	short_name	Naming on the home screen.
3.	theme_color	Color settings on the application tool bar.
4.	background_color	For splash screen use.
5.	display	To select the display type of the application.
6.	orientation	Defines the application view position.
7.	scope	Defines the navigation scope of the application.
8.	start_ur	To tell the browser where the location of the application starts running when run by the user.

C. Service Worker

Service worker is a file that contains javascript code that runs in the background or behind the scenes. The service worker is an important component of the PWA simulation application that allows the application to run offline, the service worker also manages network requests and plays with caching because the service worker acts as an intermediary between the user and the website server that will be accessed. To allow the application to be accessed offline, the service worker must be registered first so that it can fetch or news content data from the newsapi.org website whose data has been entered into the application database.

The response mechanism of the PWA application service worker is cache first, network first, then generic fallback, meaning that if a website page is accessed for the first time, the website will be cached first and if it fails, it will return to the network. If it fails to present data or a request from an application that is not in the cache and network, it will be handled by displaying a generic fallback informing that the page failed to load or is not available as shown in Figure 7. The generic fallback mechanism can be seen in Figure 8.

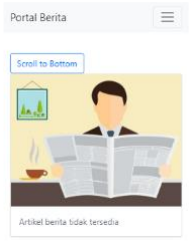

Figure 7. Generic Fallback

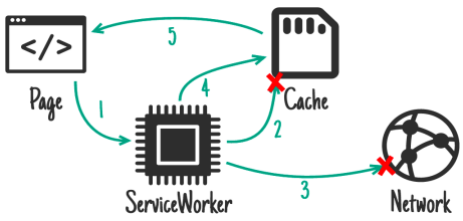

Figure 8. Generic Fallback Mechanism

4.2 Testing

Response time measurement testing was carried out using 3 tools, namely GTMetrix, Lighthouse and Chrome DevTools, the device used was a laptop with specifications that met software and hardware requirements. The network speed when testing is 14.77 Mbps.

a. GTMetrix

Testing using GTMetrix aims to measure the performance of each simulation application based on the parameters available on the GTMetrix site, namely Page Speed, YSlow, Fully Loaded Time, Total Page Site and Number of Requests.

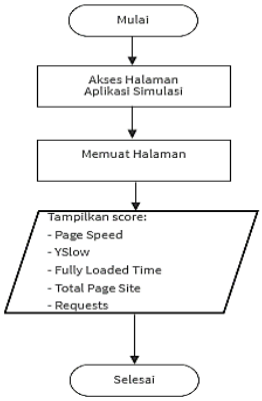

Figure 9. Flowchart GTMetrix Testing

Figure 9 is the stages of testing using GTMetrix. GTMetrix provides assessment results in grades and numerical scores, grades in grades are symbolized by the letters A, B, C, D, E, and F while the scores are marked with numbers. Testing is carried out by entering the url address of each simulation application, Figure 9 shows a comparison of testing a simulated application using GTMetrix.

Figure 10. Comparison Testing Using GTMetrix

Table 4. Score Testing Using GTMetrix

Simulation Apps	Page Speed	YSlow	Fully Loaded Time	Total Page Size	Requests
Native Web	A (99%)	A (98%)	1.4 s	35.1 kb	4
Mobile Web	A (99%)	A (98%)	1.6 s	35.2 kb	4
PWA	A (99%)	A (97%)	2.1 s	40.5 kb	6

Table 4 shows that the performance of each page of the three simulation applications is in grade A where the Page Speed score for each application is 99%, the YSlow score on Mobile Web and Native Web is 98% while PWA gets a score of 97%. Likewise with the YSlow score, on the parameters Fully Loaded Time, Total Page Size and number of Requests, the PWA simulation application still loses to Mobile Web and Native Web, which takes 2.3s with a total page size of 40.5 kb and a total of 6 requests.

This happens because in PWA there are several additional processes that are carried out and not carried out on the Mobile Web as well as the Native Web, such as PWA having to register a Service Worker, then storing the Application Shell in cache, there are additional page requests in accessing manifest files and icon files, as well other processes. Even so, there is no significant difference between PWA and Mobile Web and Native Web, it's just that the network access speed is sometimes unstable during the testing process.

b. Lighthouse Tool

The Lighthouse Tool is used to audit the performance of a web including PWA, the lighthouse tool provides parameters that have been set on the official Google Developer website, namely Performance, Accessibility, Best Practices, SEO, and the implementation of the PWA itself with a score rating from 0-100. Table 5 shows the scoring category for the lighthouse, if a web after being tested gets a score of 0-49, then the web is said to be slow, and the score is red. If you get a score of 50-89, then the web is standard or average like the web in general and the score is orange. If the web gets a score of 90-100, then the status of the web is fast or very good and the score will be green. The stages in the Lighthouse test can be seen in Figure 11.

Tabel 5. Category Assessment Score on Lighthouse

Skor	Status	Warna Skor
0 - 49	Lambat	Merah
50 - 89	Rata-rata	Jingga
90 - 100	Cepat	Hijau

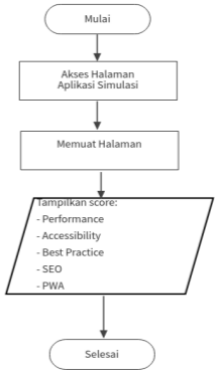

Figure 11. Flowchart Lighthouse Testing

Testing the simulation application using Lighthouse can be seen in Figure 12-14. Figure 11 is a test on the Native Web simulation application, Figure 12 tests on the Mobile Web and Figure 13 tests on PWA.

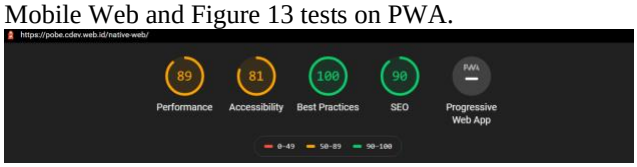

Figure 12. Score Lighthouse Aplication Native Web

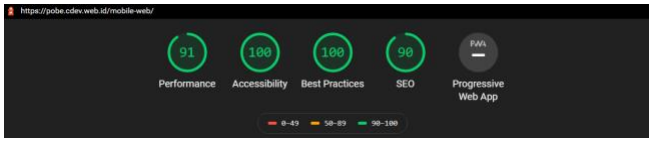

Figure 13. Score Lighthouse Aplication Mobile Web

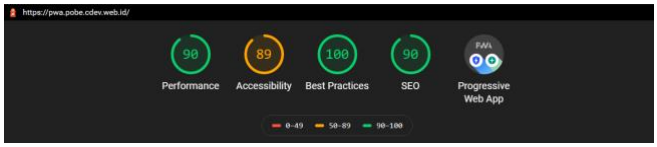

Figure 14. Score Lighthouse Aplication PWA

Table 6. Score Testing Using Lighthouse

Simulation Apps	Performance	Accessibility	Best Practice	SEO	PWA
Native Web	89	81	100	90	Tidak ada
Mobile Web	91	100	100	90	Tidak ada
PWA	90	89	100	90	Ada

Table 6 shows testing using Lighthouse on the three simulation applications, the Mobile Web is superior to PWA and Native Web. Because the PWA simulation application detects PWA in the Lighthouse tool, there is a separate assessment along with recommendations. PWA assessment criteria are divided into 3, namely fast and reliable as shown in Table 7, installable in Table 8, and PWA Optimized in Table 9.

Table 7. Criteria Fast and Reliable

No.	Condition	Information
1	The web should load pages fast enough on mobile networks	Fulfilled
2	The final web page should respond with code 200 when offline	Fulfilled
3	start_url responds with code 200 when offline	Fulfilled

Table 8. Criteria Installable

No.	Condition	Information
1	Must use HTTPS	Fulfilled
2	Register the service worker that controls the page and start_url	Fulfilled
3	The created web app manifest meets PWA requirements	Fulfilled

Tabel 9. Criteria PWA Optimized

No.	Condition	Information
1	Redirect http traffic to https	Fulfilled
2	Configure for a custom splash screen	Fulfilled
3	Sets the theme color for the address bar	Not Fulfilled
4	Creates the right content size for the viewport	Fulfilled
5	Has a tag with width or initial-scale	Fulfilled
6	Load some content when Javascript is not available	Fulfilled
7	Provides a valid apple-touch-icon	Not Fulfilled
8	Manifest implements a maskable icon	Not Fulfilled

The recommendations suggested by the Lighthouse Tool for criteria that are not met in points 3,7 and 8 of the PWA Optimized criteria are:

- Added browser address bar theme color to match the website.
- For ideal view on iOS when user add PWA app to home screen, specify valid apple-touch-icon. The important point is that the icon size is at least 192px and not transparent.
- Having a maskable icon ensures that the image/icon can fill the available space when installing the application on the device.

c. Chrome DevTools

The Chrome DevTools test aims to test the number of file pages that can affect the performance of simulated applications, especially in PWA. This test scenario is carried out on three multilevel home pages, about, and news which will be tested on one of the Chrome DevTools features, namely the Network Panel feature. The test scenario can be seen in Figure 15.

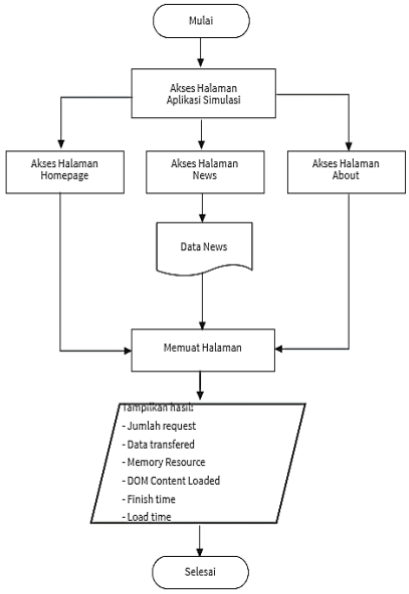

Figure 15. Flowchart Chrome DevTools

The Network Panel is available when you right-click on a website then select inspect and select Network. This feature displays details on the file name, status received by http, file format, file size, time to access the page, and the waterfall web page file that has been displayed. However, this test compares the parameters contained in the Network taskbar which is the result of loading a page.

Table 10. Native Web Testing (P1) and (P2)

Hal.	*Requests* (hal)		*Transferred* (kB)		*Resource* (kB)		*Finish* (ms)		DOM *Content Loaded* (ms)		*Load* (ms)	
	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2
Home	3	3	0,153	0,153	210	210	354	2520	740	2820	772	2830
About	4	4	0,153	0,153	423	423	322	381	721	720	844	744
News	401	399	374	351	42800	42800	26540	25500	914	693	943	712

Table 11. Mobile Web Testing (P1) and (P2)

Hal.	*Requests* (hal)		*Transferred* (kB)		*Resource* (kB)		*Finish* (ms)		DOM *Content Loaded* (ms)		*Load (ms)*	
	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2
Home	3	3	0,153	0,153	211	211	602	572	733	707	757	739
About	4	4	0,153	0,153	424	424	537	2370	680	2495	743	2520
News	460	399	351	351	42800	42800	32500	24350	857	1100	904	1120

Table 10 and Table 11 show the results of testing the Mobile Web and Native Web simulation applications, the difference between each page when tested 2 times is not very significant and the difference is only small. For the resource parameter there is no change at all for the mobile web or native web because when loading the data page it will always be called over the network.

Table 12. PWA Testing (P1) and (P2)

Hal.	*Requests* (hal)		*Transferred* (kB)		*Resource* (kB)		*Finish* (ms)		DOM *Content Loaded* (ms)		*Load* (ms)	
	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2
Home	14	7	0,168	0	645	217	880	739	669	608	691	638
About	9	9	0	0	430	431	819	730	630	593	741	646
News	785	460	730	348	770000	41700	33570	24180	640	512	704	532

Table 12 shows the results of testing the PWA simulation application, the test was carried out 2 times to see the difference when accessing the page the second time. A significant difference can be seen from the data resource, which requires 77,000 kB of data the first time it is accessed, and when the page is accessed again, only 41,700 kB of data is needed. This happens because the first time the page is loaded the cache data will be stored in the service worker, therefore when accessing the page for the second time the data will be called from the service worker and not through the network. That's when PWAs can be accessed while the network is offline.

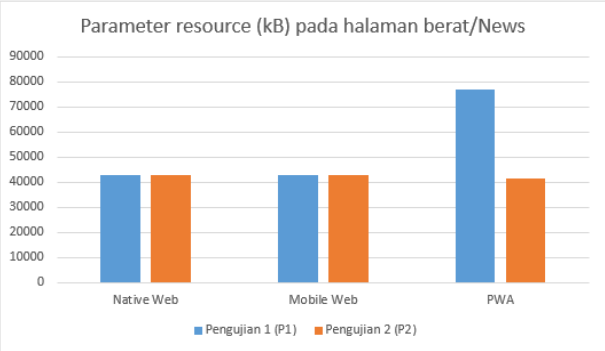

Figure 16. Diagram Parameter Resource on News

Figure 16 shows that on heavy pages with 2 tests, PWA resources (kB) will decrease and the data is stored in the service worker.

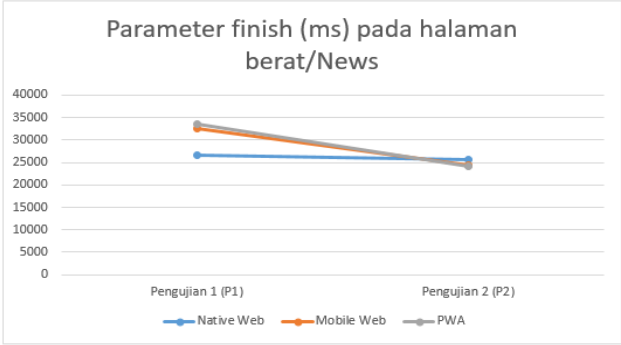

Figure 17. Grafik Parameter Finish Resource on News

Figure 17 shows that on a heavy page with 2 tests, the PWA finish time parameter (ms) is reduced and it is faster than the two applications because at the time of testing the two data were taken from the service worker.

4.3 Results and Analysis

GTMetrix testing the performance of the PWA simulation application is still inferior to the performance of the other two simulation applications, even though the difference is small. In the Lighthouse Tools test, the PWA simulation application has implemented the requirements of the PWA criteria. Among them are using https, there is an add to home screen (A2HS) and can be accessed when offline. As shown in Figure 18, Figure 19, and Figure 20.

Figure 18. Add A2HS Portal News

Figure 19. View A2HS Portal News

Figure 20. View Network Offline / Airplane Mode

The advantages and disadvantages of the simulation application are:

a. Native Web: Pros of Native Web: Low difficulty level when using the application. Disadvantages of Native Web: The network must always be online, the display is not responsive.
b. Mobile Web: Strengths of Mobile Web: Responsive Design, can operate on cross platforms. Disadvantages of Mobile Web: Network connection must always be online.
c. PWA Simulstion Application: Can be used offline, responsive design, special icon when Add to Home Screen (A2HS), data caching management with service workers, fast performance when accessed more than once.
d. Disadvantages of PWA: Slow performance when accessing the page for the first time.

The application of modern technology using the Progressive Web App is very suitable for applications that are frequently accessed by users, for example social media applications, transactional buying and selling applications / e-commerce, etc.

5. CONCLUSIONS

Based on the research that has been done, measurement of response time performance on Progressive Web Applications compared to Mobile Web and Native Web, seen from the number of times the page is accessed and adjusts to the amount of cache data as well as network availability used during testing. Web page file size affects the performance of Progressive Web Applications, as evidenced by testing the number of nested pages. However, it proves to have an effect here when accessing the page more than once.

In the second test, the minimum number of page files (home) is 217 kB with a page load time of 638 ms, while the average page (about) is 431 kB with a page load time of 646 ms, and when accessing the page weight (news) with size 41700 kB page load time is 532 ms. Even though the average page (about) experiences an increase in time when heavy pages are accessed the page loads faster than the minimum page (home) and medium page (about). The development of technology applications in making a website, especially PWA, can continue to be carried out by adding other features to be able to compete and be compared with other technologies. In testing the News page, additional data can be added to see the accuracy of the effect of PWA performance, and the existence of tests based on page / pagination.

REFERENCES

[1] J. P. Dias and H. S. Ferreira, “Automating the Extraction of Static Content and Dynamic Behaviour from e-Commerce Websites,” *Procedia Comput Sci*, vol. 109, pp. 297–304, 2017, doi: 10.1016/j.procs.2017.05.355.

[2] K. Syaifudin, W. Nafisah, and R. Dian, “Analisis Usability pada Perbandingan Web-Native dengan Web Berbasis Progressive Web App.”

[3] S. Sophonhiranrak, “Features, barriers, and influencing factors of mobile learning in higher education: A systematic review,” *Heliyon*, vol. 7, no. 4, p. e06696, Apr. 2021, doi: 10.1016/j.heliyon.2021.e06696.

[4] M. R. Ridho, A. Pinandito, and R. K. Dewi, “Perbandingan Performa Progressive Web Apps dan Mobile Web Terkait Waktu Respon, Penggunaan Memori dan Penggunaan Media Penyimpanan,” 2018. [Online]. Available: http://j-ptiik.ub.ac.id

[5] V. Karpagam, “Performance Enhancement Of Webpage Using Progressive Web App Features,” 2017. [Online]. Available: www.ijirae.com

[6] A. Kurniawan, I. S. Areni, and A. Achmad, “Implementasi Progressive Web Application pada Sistem Monitoring Keluhan Sampah Kota Makassar,” *Jurnal Penelitian Enjiniring*, vol. 21, no. 2, pp. 34–38, Jan. 2018, doi: 10.25042/jpe.112017.05.

[7] V. Sharma, R. Verma, V. Pathak, M. Paliwal, and P. Jain, “Progressive Web App (PWA) - One Stop Solution for All Application Development Across All Platforms,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 1120–1122, Apr. 2019, doi: 10.32628/CSEIT1952290.

[8] N. Nurwanto, “Penerapan Progressive Web Application (PWA) pada E-Commerce,” *Techno.Com*, vol. 18, no. 3, pp. 227–235, Aug. 2019, doi: 10.33633/tc.v18i3.2400.

[9] O. Adetunji, C. Ajaegbu, N. Otuneme, and O. J. Omotosho, “Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development,” *Technology, and Sciences (ASRJETS) American Scientific Research Journal for Engineering*, vol. 68, no. 1, pp. 85–99, 2020, [Online]. Available: http://asrjetsjournal.org/

[10] D. Haryanto, Z. Reno, and S. Elsi, “Analisis Performance Progressive Web Apps Pada Aplikasi Shopee”.

[11] A. D. Rebecca Fransson, “Comparing Progressive Web Applications with Native Android Applications,” Linnaeus University, Faculty of Technology, Sweden, 2017.

[12] Farid Said Tahirshah, “Comparison between Progressive Web App and Regular Web App,” Blekinge Institute of Technology, Sweden, 2019.