Proceeding of International Conference on Digital Advance Tourism, Management and Technology 2023
Volume. 1 No. 2 2023 e-ISSN : 3031-5999, Hal 586-596
DOI: https://doi.org/10.56910/ictmt.v1i2.126

Collaboration of Progressive Web App (PWA) And Firebase Cloud Messaging (FCM) for Optimal Performance Mailing Software

Dwi Purnomo Putro [1], Adhika Pramita Widyassari[2], Dea Salsabilla[3]
[1-3] Sekolah Tinggi Teknologi Ronggolawe, Blora, Indonesia

Corresponding email: dwipp@gmail.com

Abstract *The problem of correspondence cannot be separated from the ease, accuracy and speed of the processing process. In research from Riswandi Ishak and H. Trizaka, they proposed report management software as well as notification of disposition and monitoring of correspondence as a solution to correspondence problems. However, there are shortcomings in the notification process for the disposition of correspondence, which requires the software to be actively open. In research, P. Dwi proposed a notification solution with Firebase Cloud Messaging (FCM), so that it can send notifications as long as the browser is active and connected to the internet, even without opening the software. There is a problem currently when the software becomes unstable when the internet connection is bad or offline. Progressive Web Apps (PWA) offers the concept of web-based application development by implementing browser technology such as service workers and app manifests. PWA is capable of displaying pages offline but cannot save, change, or delete data in the database. The test results of this research used Lighthouse and showed an average score of 100 on the PWA criteria, 85 on the performance criteria, 97 on the accessibility criteria, and 100 on the best practices criteria. Additional results obtained by implementing PWA mean page loading times are 26.6% faster with cache and service workers. The PWA and FCM concepts provide the best experience in using Mailing Software even with minimal internet connection or offline. This strategy was chosen to still get a fast response when running the mail processing software.*

Keywords *Mail; push notification; firebase cloud messaging; progressive web apps;*

INTRODUCTION

Correspondence is a written communication medium that continues to exist and is used by many institutions/agencies/organizations and seems difficult to replace by other communication media (Trizaka et al., 2019) . Correspondence is very important for the administration of institutions/agencies/organizations, especially in the process of managing its creation (Muthia Farida & Dian Agustini, 2017) , (Rahman et al., 2019) . Therefore, the issue of correspondence needs to receive special attention, so that it can have a positive impact and not cause losses to the parties concerned.

There are problems in correspondence such as human error regarding errors/repetitions in forming letter numbers (Priyadi & Lestari, 2018) , (Riswandi Ishak et al., 2020) . Then there is the issue of whether or not the process of searching for letters, monitoring the status of letters and archiving letters is easy (Putra et al., 2019) , (Wijaya et al., 2018) , (Ardhia et al., 2019) . Furthermore, there is the problem of whether or not the completion of letters quickly, distribution of letters, disposition of letters and presentation of letter reports (Eka Purnama Rijaludin et al., 2018) , (Nur et al., 2019) , (Ikhsan & Ramadhani, 2020) (Febi, 2018) . Some of these problems have been offered solutions by researchers leading to the design, creation and development of software/applications.

Received November 19, 2023; Revised Desember 01 , 2023; Accepted Desember 31, 2023
* Dwi Purnomo Putro , dwipp@gmail.com

There is research by Riswandi et all (Riswandi Ishak et al., 2020) and Ikhsan et al. (Ikhsan & Ramadhani, 2020) offers a solution to solving correspondence problems through creating website-based software. The advantages of their research are seen in letter management, letter search and structured letter storage media, simpler/easier processes and automatic letter reporting. However, the system does not involve a letter verifier, so there is no integration concept to increase the speed of business processes and there is no monitoring of letter disposition status to minimize the risk of human error. Then their research also did not continue from the previous year's research which had answered system integration.

The 2019 research offers solutions to correspondence problems related to system integration between users. For example, research by H. Trizaka et al., G. Putra et al. and A. Ardhia et al. has created website-based correspondence software with improved letter management and disposition processes. Where their research implemented notifications to increase response to completion of letters and there was a monitoring feature to monitor information on the condition of letters, whether they had been received, read and followed up. Because their research created notifications by requiring users to open the software, the user's response to finding out about mail updates was not optimal.

Then research from (Putro et al., 2022). creating website-based correspondence software and adding the concept of notifications using Firebase Cloud Messaging (FCM). The advantage of implementing this concept is that notification messages can run smoothly on Android, iOS and websites without expensive costs. However, FCM requires the user to be in an internet network condition (online), so that active notification messages run. Then, to support the concept of running website-based software displays in offline conditions, Progressive Web Apps (PWA) need to be developed.

Progressive Web Apps (PWA) offers the concept of web-based application development by implementing browser technology such as service workers and app manifests (Huber et al., 2021). PWA is capable of displaying pages offline but cannot save, change, or delete data in the database. In this research, correspondence software will be created that can run even offline using a service worker implemented through the PWA concept, and still utilizes FCM for the disposition of correspondence. This strategy was chosen to provide the best experience in using website-based mailing software and still get a fast response when running it.

METHOD

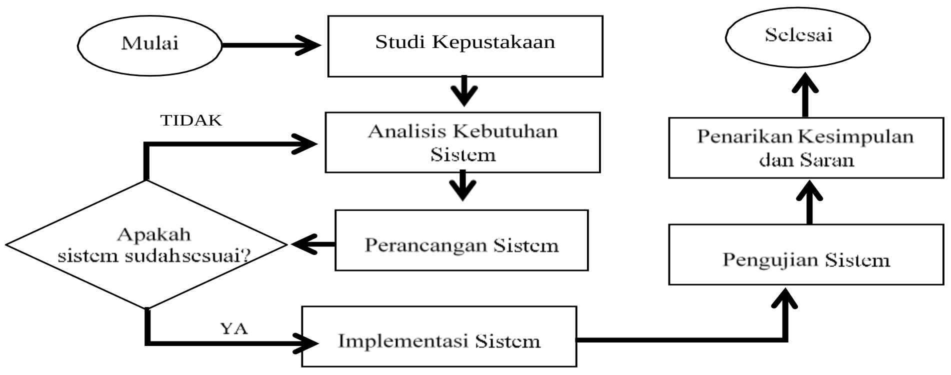


Figure 1. Research method flow diagram

The steps in preparing this research begin with finding literature studies and needs analysis, then designing, implementing and testing the software to be created. When there is a mismatch between the development results and requirements, it needs to be reviewed back to the system requirements analysis stage. The research method flow diagram can be seen in figure 1.

1. **Literature Study**
   The literature study explains the theoretical basis and concepts needed to build a system based on the literature references found.
2. **Requirements Analysis**
   System requirements analysis provides a general description of the system, which explains the main features of the system in outline. Requirements implemented in the system are based on the best combination sourced from references that have been collected. System requirements include identifying actors who interact with the system, specifying in more detail system access rights requirements and modeling requirements in the form of use case diagrams and use case scenarios.
3. **System planning**
   From the results of the analysis, an architectural design is created which explains the architecture from client to server, database design, component design and interface design. In designing this system, tools are used to design the system to be more structured, such as: UML (Unified Modeling Language) and ERD (Entity Relationship Diagram). The results of the design evaluation carried out determine whether the system is appropriate and accurate as desired or not? If it is not appropriate, then the work process returns to the needs analysis stage, and if the system is appropriate and accurate as needed then it will enter the next stage.
4. **System Implementation**
   System implementation is carried out to start changing the results of the requirements design towards the program coding process, creating appropriate interfaces and databases. At this stage a series of programs or program units related to database storage use MySQL. And the entire program unit code in this software uses the HTML, CSS, JQUERY, PHP, AJAX and JSON programming languages. The local website testing process uses XAMPP which has integrated MySQL and Apache.
5. **System Testing**
   The testing process searches for system errors and fixes the bugs found. Further testing and evaluation uses a Progressive Web Apps testing tool called Lighthouse, to see what percentage

of applications meet the PWA results criteria. Testing was also carried out by applying online and offline conditions and then looking at the response time in Chrome DevTools.

6. **Drawing Conclusions and Suggestions**
   From the problems described in the problem formulation, conclusions will be drawn to get answers to these problems. And suggestions are given for reference for further research.

RESULTS AND DISCUSSION

1. Software Design

A use case is a diagram that describes all activities by the system from the user's perspective. The use case of the system to be developed can be seen in Figure 2.

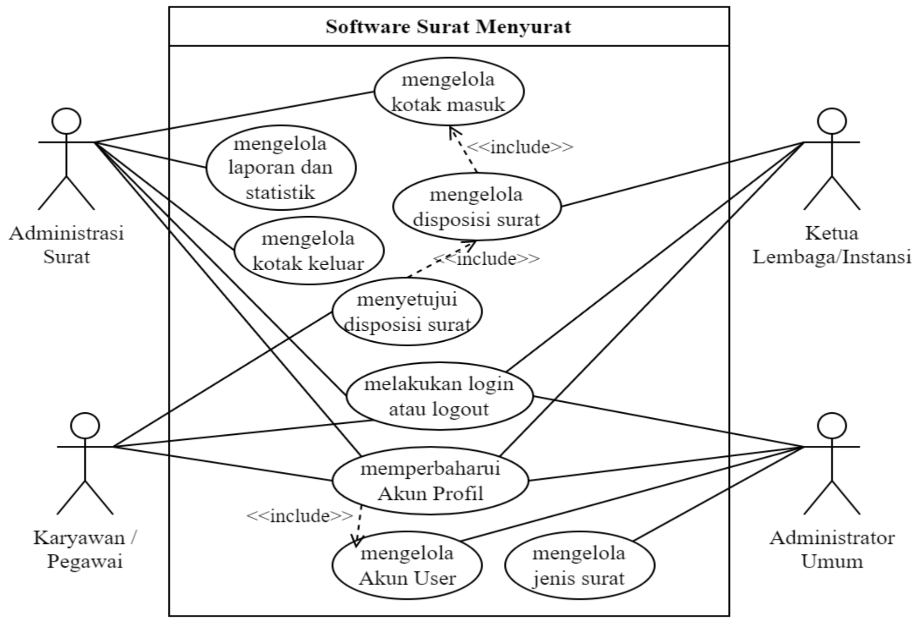


Figure 2. Use case for correspondence software

The definition of actor in the diagram can be seen in Table 1.

Table 1: *Definition of correspondence software actors*

Actor Name	Definition
Mail Administrator	Mail admins have access to login/logout, *update* profile accounts, manage inboxes (including adding initial mail disposition), manage outboxes, manage report dates and statistics.
General Administrator	General admins have access to log in/log out, *update* profile accounts, manage user accounts, and manage mail types.
Employee	Employees have access to log in/log out, *update* profile accounts, and approve letter disposition.
Head of Institution	Heads of Institutions/Agencies have access to log in/log out, *update* profile accounts, and manage letter disposition (including approving letter disposition)

The definition of user case in the diagram can be seen in Table 2.

Table 2: Definition of use case for correspondence software

Use case name	Definition
Login/logout	All actors can log in before accessing the software menu and can log out after completing accessing the software menu.

Update Account Profile	In this use case, all actors can update profile data information along with the password used when logging into the software.
Managing User Accounts	This use case allows actors to add, edit, display and delete data from all users of this correspondence software.
Manage Mail Types	This use case allows actors to add, edit, display and delete data from a type of letter. Examples of letter types: Ordinary, Confidential, Immediate/Important
Manage your inbox	This use case allows actors to add, edit, display and delete data from the inbox (including managing mail disposition and managing mail sender data)
Manage mail disposition	This use case facilitates actors being able to add, edit, display, delete and approve employee disposition assignments.
Approve the disposition of the letter	This use case facilitates the actor being able to edit changes or approve the disposition of the letter
Manage Outbox	This use case allows actors to add, edit, display and delete data from the outbox (including determining the destination of the letter).
Manage Reports and Statistics	This use case facilitates actors to display report data filters and statistics based on date or year. After displaying, the data can be downloaded in Excel or PDF format.

Below is a picture of the main database design in the correspondence software according to the design requirements can be seen in figure 3.

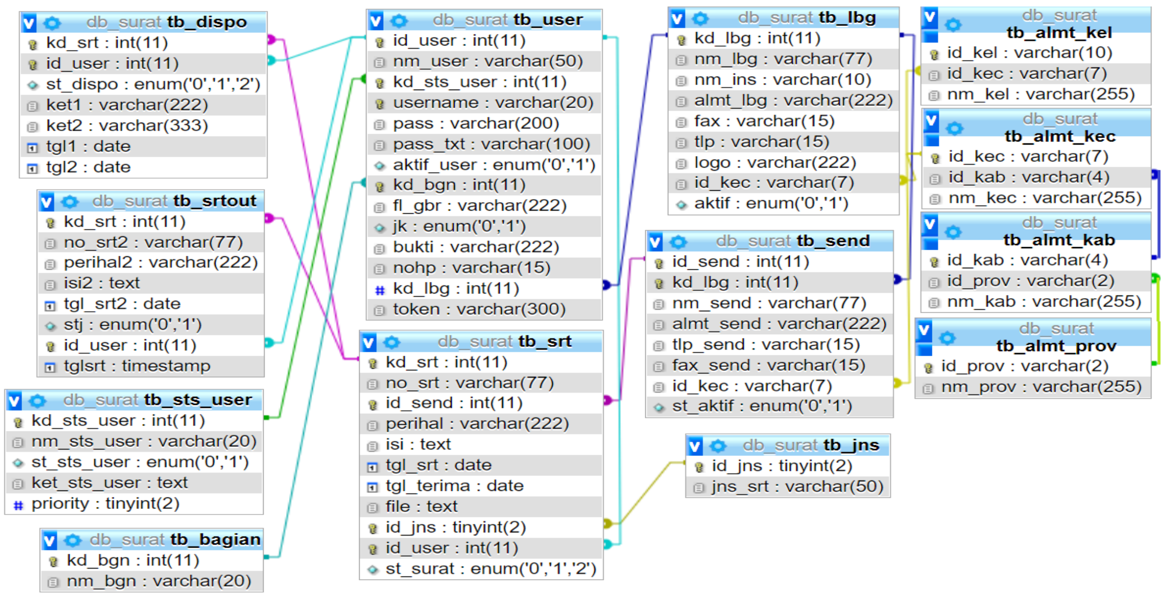


Figure 3. Mailing software database design

2. Software Implementation

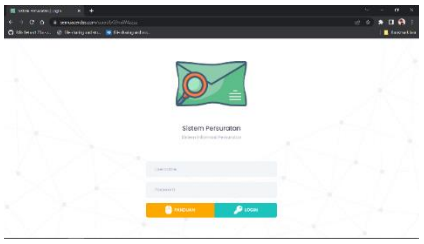
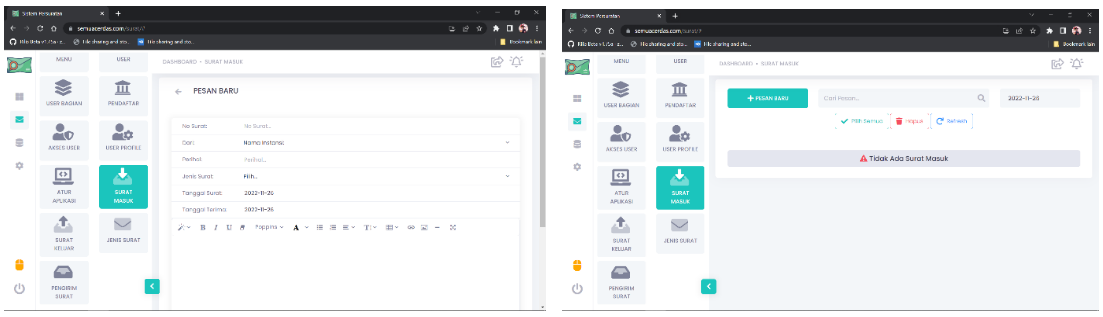


Figure 4. Mailing Software Login Page

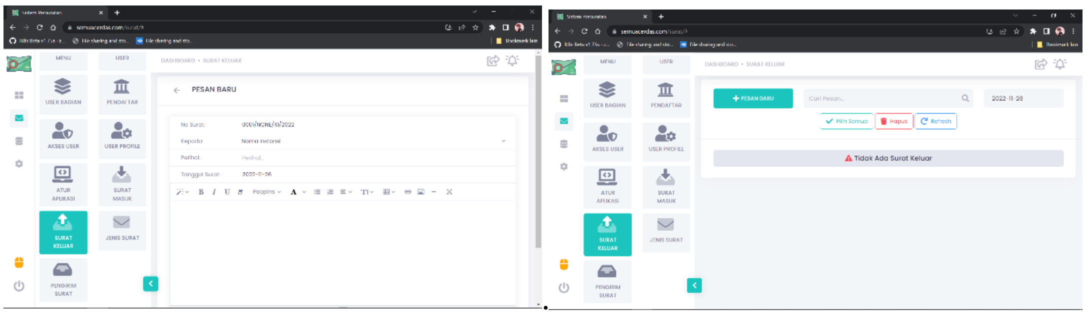
Before you can access the existing menu, the user must log in as shown above can be seen in figure 4. If part When the administration runs the incoming letter menu, it must enter the data for the letter to in system through data input form like shown in the following image.



Tidak Ada Surat Masuk

Figure 5. Input Form and Data Display Page Letter inbox

If there is outgoing mail to be sent, it can be input via the outgoing mail data input page as shown below:



Tidak Ada Surat Keluar

Figure 6. Input and Display Form Page Data Letter outbox

Every letter inbox by mail administration and management of its disposition, then system will be automatic send notification i to the intended user (disposition). A notification will appear in the right corner bottom on the computer or in the notification section on smartphones. Example appearance notification can be seen in the following.

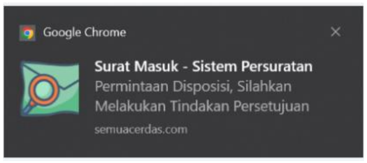

Google Chrome
Surat Masuk - Sistem Persuratan
Permintaan Disposisi, Silahkan Melakukan Tindakan Persetujuan
semuacerdas.com

Figure 7. Appearance Notifications on computers

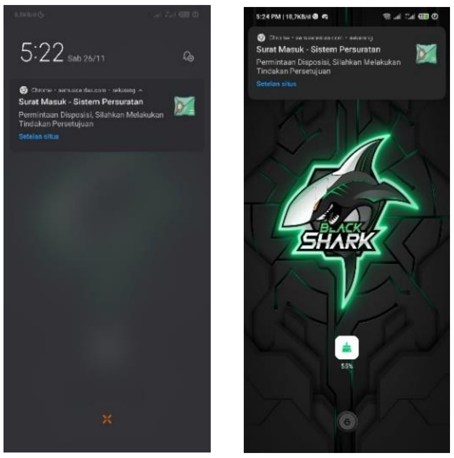

Surat Masuk - Sistem Persuratan
Permintaan Disposisi, Silahkan Melakukan Tindakan Persetujuan
Setelan situs

Figure 8. Appearance Notifications on Smartphones

To be able to carry out the notification process, there are important things that we must condition. First you have to get the sender ID and server key at the URL https://firebase.google.com

3. Test Execution Environment

environment that will be used to carry out the implementation is as follows:

- **Hardware**
  a. Intel(R) Pentium(R) CPU 4425Y @ 1.70GHz 1.70 GHz
  b. 4.00 GB RAM memory
- **Software**
  a. Operating System: Windows 11 Home 64-bit
  b. Google Chrome 58.0.3029.110 (64-bit)
  c. Lighthouse 2.1.0

4. Evaluation of Testing Using Lighthouse

Testing was carried out 30 times on 4 assessment criteria, namely PWA, Performance, Accessibility and Best Practices. The test results can be seen in Table 3 below.

Table 3: Lighthouse Test Results Using Caliber

No	PWA	Performance	Accessibility	Best Practices
1	100	95	97	100
2	100	85	97	100
3	100	92	97	100
4	100	96	97	100
5	100	95	97	100
6	100	53	97	100
7	100	83	97	100
8	100	96	97	100
9	100	94	97	100
10	100	93	97	100
11	100	63	97	100
12	100	62	97	100
13	100	95	97	100
14	100	94	97	100
15	100	96	97	100
16	100	95	94	100
17	100	92	97	100
18	100	93	97	100
19	100	92	97	100
20	100	94	97	100
21	100	95	97	100
22	100	80	97	100
23	100	96	97	100
24	100	68	97	100
25	100	69	97	100
26	100	96	97	100
27	100	94	97	100
28	100	96	97	100
29	100	72	97	100
30	100	62	97	100

The test results using Lighthouse show that the software built can be said to meet the criteria for a Progressive Web Apps with a score of 100 out of 100. Apart from that, it also got good scores on three other criteria, namely performance criteria with an average of 85 out of 100, accessibility criteria with an average -an average of 97 out of 100, and best practices criteria with an average of 100 out of 100.

In the performance criteria, the values obtained have a fairly large range, namely from 53 to 96. This large range of values is because the performance value is very dependent on the condition of the internet network. When the internet network is slow, low values will be generated.

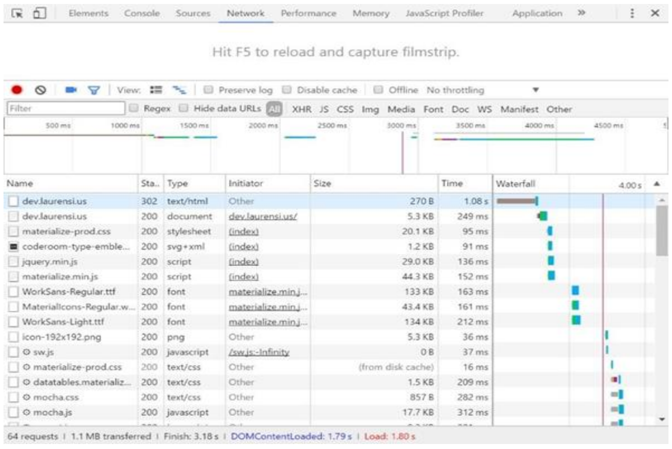
5. Evaluation of Network Response Testing Using Chrome DevTools

Discussion of network response testing results using Chrome DevTools is divided based on the following scenarios:

a. In the first scenario the page will be tested with the following conditions:

• Page Views	: first time
• Service Worker	: not yet available
• Page cache	: not yet available
• Internet	: online

The test results of the first scenario are shown in Figure 9, it is known that there were 64 requests made from the page, 1.1 MB of data was received, the page load time was 1.8 seconds, and all requests were completed in 3.18 seconds. This large number of requests come from service workers where all static files are downloaded and stored in the cache. This process occurs behind the scenes and after the page has finished loading, so it does not interfere with the user using the application.



Name	Sta..	Type	Initiator	Size	Time	Waterfall 4.00 s
dev.laurensi.us	302	text/html	Other	270 B	1.08 s	
dev.laurensi.us	200	document	dev.laurensi.us/	5.3 KB	249 ms	
materialize-prod.css	200	stylesheet	(index)	20.1 KB	95 ms	
coderoom-type-emble...	200	svg+xml	(index)	1.2 KB	91 ms	
jquery.min.js	200	script	(index)	29.0 KB	136 ms	
materialize.min.js	200	script	(index)	44.3 KB	152 ms	
WorkSans-Regular.ttf	200	font	materialize.min.j...	133 KB	163 ms	
MaterialIcons-Regular.w...	200	font	materialize.min.j...	43.4 KB	161 ms	
WorkSans-Light.ttf	200	font	materialize.min.j...	134 KB	212 ms	
icon-192x192.png	200	png	Other	5.3 KB	36 ms	
⚙ sw.js	200	javascript	/sw.js:-Infinity	0 B	37 ms	
⚙ materialize-prod.css	200	text/css	Other	(from disk cache)	16 ms	
⚙ datatables.materializ...	200	text/css	Other	1.5 KB	209 ms	
⚙ mocha.css	200	text/css	Other	857 B	282 ms	
⚙ mocha.js	200	javascript	Other	17.7 KB	312 ms	

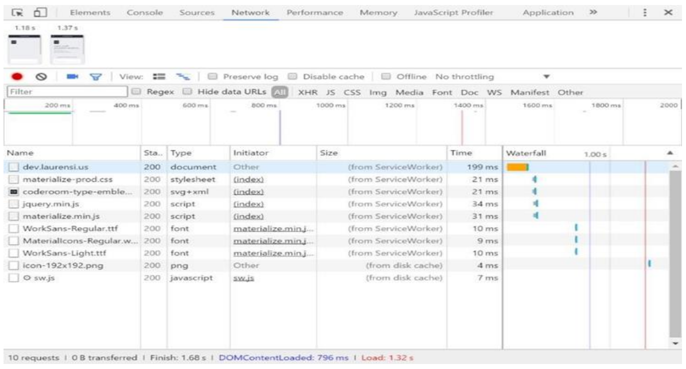
64 requests | 1.1 MB transferred | Finish: 3.18 s | DOMContentLoaded: 1.79 s | Load: 1.80 s

Figure 9. First Visit Response Network

b. In the second scenario the page will be tested with the following conditions:

• Page Views	: after first
• Service Worker	: is active
• Page cache	: already exists
• Internet	: online

The results of the second scenario test are shown in Figure 10, where it is known that there were 10 requests made from the page, 0 B data were received, the page loading time was 1.32 seconds, and all requests were completed in 1.68 seconds. The number of requests made by the same page after the service worker is active is much less than in the first scenario because the service worker does not download any new data for static files. In this scenario, no data is received by the page because all the data is already in the cache and returned to the page by the service worker so that the page load time is 0.48 seconds or 26.6% faster.



Name	Sta..	Type	Initiator	Size	Time
dev.laurensi.us	200	document	Other	(from ServiceWorker)	199 ms
materialize-prod.css	200	stylesheet	(index)	(from ServiceWorker)	21 ms
coderoom-type-emble...	200	svg+xml	(index)	(from ServiceWorker)	21 ms
jquery.min.js	200	script	(index)	(from ServiceWorker)	34 ms
materialize.min.js	200	script	(index)	(from ServiceWorker)	31 ms
WorkSans-Regular.ttf	200	font	materialize.min.j...	(from ServiceWorker)	10 ms
MaterialIcons-Regular.w...	200	font	materialize.min.j...	(from ServiceWorker)	9 ms
WorkSans-Light.ttf	200	font	materialize.min.j...	(from ServiceWorker)	10 ms
icon-192x192.png	200	png	Other	(from disk cache)	4 ms
⚙ sw.js	200	javascript	sw.js	(from disk cache)	7 ms

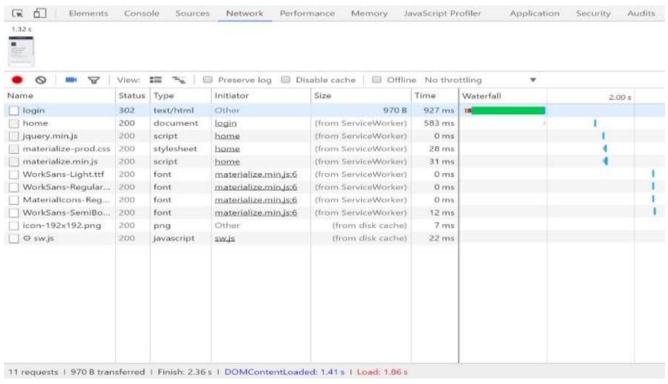
10 requests | 0 B transferred | Finish: 1.68 s | DOMContentLoaded: 796 ms | Load: 1.32 s

Figure 10. Network Response Repeat Visit

c. In the third scenario the page will be tested with the following conditions:

- Page Views : after first
- Service Worker : is active
- Page cache : not yet available
- Internet : online

The results of the third scenario test are shown in Figure 11, where it is known that there were 11 requests made from the page, 970 B of data were received, the page load time was 1.86 seconds, and all requests were completed in 2.36 seconds. The number of requests made increases by one because they go through the login process first. The data received increases because the visited page is not available in the cache so the request is forwarded to the server and the data received is only the HTML data of the new page, while the static file uses data from the cache.



Name	Status	Type	Initiator	Size	Time
login	302	text/html	Other	970 B	927 ms
home	200	document	login	(from ServiceWorker)	583 ms
jquery.min.js	200	script	home	(from ServiceWorker)	0 ms
materialize-prod.css	200	stylesheet	home	(from ServiceWorker)	28 ms
materialize.min.js	200	script	home	(from ServiceWorker)	31 ms
WorkSans-Light.ttf	200	font	materialize.min.js:6	(from ServiceWorker)	0 ms
WorkSans-Regular...	200	font	materialize.min.js:6	(from ServiceWorker)	0 ms
MaterialIcons-Reg...	200	font	materialize.min.js:6	(from ServiceWorker)	0 ms
WorkSans-SemiBo...	200	font	materialize.min.js:6	(from ServiceWorker)	12 ms
icon-192x192.png	200	png	Other	(from disk cache)	7 ms
⚙ sw.js	200	javascript	sw.js	(from disk cache)	22 ms

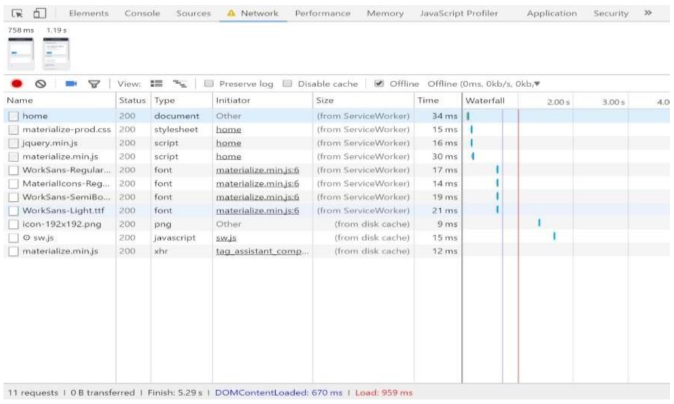
11 requests | 970 B transferred | Finish: 2.36 s | DOMContentLoaded: 1.41 s | Load: 1.86 s

Figure 11. Network Response No Cache Page

d. In the fourth scenario the page will be tested with the following conditions:

• Page Views	: after first
• Service Worker	: is active
• Page cache	: not yet available
• Internet	: online

The fourth scenario test results are shown in Figure 12, where it is known that there were 11 requests made from the page, 0 B data were received, the page loading time was 0.959 seconds, and all requests were completed in 5.29 seconds.



Name	Status	Type	Initiator	Size	Time
home	200	document	Other	(from ServiceWorker)	34 ms
materialize-prod.css	200	stylesheet	home	(from ServiceWorker)	15 ms
jquery.min.js	200	script	home	(from ServiceWorker)	16 ms
materialize.min.js	200	script	home	(from ServiceWorker)	30 ms
WorkSans-Regular...	200	font	materialize.min.js:6	(from ServiceWorker)	17 ms
MaterialIcons-Reg...	200	font	materialize.min.js:6	(from ServiceWorker)	14 ms
WorkSans-SemiBo...	200	font	materialize.min.js:6	(from ServiceWorker)	19 ms
WorkSans-Light.ttf	200	font	materialize.min.js:6	(from ServiceWorker)	21 ms
icon-192x192.png	200	png	Other	(from disk cache)	9 ms
sw.js	200	javascript	sw.js	(from disk cache)	15 ms
materialize.min.js	200	xhr	tag_assistant_comp...	(from disk cache)	12 ms

11 requests | 0 B transferred | Finish: 5.29 s | DOMContentLoaded: 670 ms | Load: 959 ms

Figure 12. Network Response Offline

This test shows the application can run offline. In this test, the application loads content to the page much faster than before because it does not wait for a response from the server at all, where all responses are returned from the cache.

CONCLUSION

The FCM concept rules for sending notifications can only run smoothly and more optimally when using the HTTPS network protocol with valid SSL. Then the PWA test results on this software using Lighthouse show that the average score is 100 on the Progressive Web Apps criteria, the average score is 85 on the performance criteria, the average score is 97 on the accessibility criteria, and the average score is 100 on the criteria best practices. Apart from that, the use of the PWA concept also shows page loading times that are 26.6% faster with cache and service workers. The PWA and FCM concepts provide the best experience in using website-based correspondence software even with minimal internet connection or offline. This strategy was chosen to still get a fast response when running the mail processing software.

REFERENCES

Ardhia, A., Kusuma, A., Rusdianto, D. S., & Brata, A. H. (2019). Pengembangan Sistem Persuratan PT PGAS Solution. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(5), 4971–4978.

Eka Purnama Rijaludin, M., Witanti, W., & Id Hadiana, A. (2018). Sistem Informasi Administrasi Persuratan Terintegrasi Jurusan Informatika Dan Fakultas Mipa Universitas Jenderal Achmad Yani. *Jurnal Mnemonic*, *1*(1), 25–31. https://doi.org/10.36040/mnemonic.v1i1.16

Febi, A. (2018). Realtime Notification pada Aplikasi Berbasis Web Menggunakan Firebase Cloud Messanging (FCM). *Mnemonic*, *1*(2), 14–17.

Huber, S., Demetz, L., & Felderer, M. (2021). PWA vs the Others: A Comparative Study on the UI Energy-Efficiency of Progressive Web Apps. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12706 LNCS*(May), 464–479. https://doi.org/10.1007/978-3-030-74296-6_35

Ikhsan, N., & Ramadhani, S. (2020). Sistem Informasi Administrasi Surat Menyurat Kantor Wilayah Kementerian Agama Provinsi Riau. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *2*(2), 141–151.

Muthia Farida, & Dian Agustini. (2017). Aplikasi Pengarsipan Surat Menyurat Pada Program Pascasarjana Universitas Islam Kalimantan Muhammad Arsyad Albanjari Banjarmasin. *Jurnal Teknik Mesin UNISKA*, *02*(02), 75–80.

Nur, P., Faridiani, R., Brata, A. H., & Brata, K. C. (2019). Pembangunan Website Manajemen Persuratan Pada Balai Pemasyarakatan Klas 1 Malang Menggunakan Metode Prototyping. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(10), 10145–10150.

Priyadi, D. A., & Lestari, E. W. (2018). Perancangan Sistem Informasi Pelayanan Surat Menyurat Pada Kantor Desa Tanjungsari Kutowinangun Kebumen Berbasis Desktop. *Jurnal Teknik Komputer*, *IV*(2), 84–91. https://doi.org/10.31294/jtk.v4i2.3444

Putra, G. P., Santoso, N., Muhammad, E., & Junemaro, A. (2019). Rancang Bangun Sistem Informasi Manajemen Persuratan Dinas Pendidikan Banyuwangi. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(5), 4276–4282.

Putro, D. P., Gunawan, I., & Suryani, P. E. (2022). Software Push Notification Disposisi Persuratan Berbasis Website Menggunakan Firebase Cloud Messaging. *Journal of Information Technology Ampera*, *3*(3), 370–381. https://journal-computing.org/index.php/journal-ita/article/view/330

Rahman, B., Susetyo, B., & Primasari, D. (2019). Analisis Kinerja Pelayanan Surat-Menyurat Berbasis Web Di Pgri Kabupaten Bogor. *IKRA-ITH INFORMATIKA : Jurnal Komputer Dan Informatika*, *3*(1), 1–11.

Riswandi Ishak, Setiaji, Fajar Akbar, & Mahmud Safudin. (2020). Rancang Bangun Sistem Informasi Surat Masuk Dan Surat Keluar Berbasis WEB Menggunakan Metode Waterfall. *Jurnal Indonesia Sosial Teknologi*, *1*(3), 198–209. https://doi.org/10.36418/jist.v1i3.33

Trizaka, H., Rusdianto, D. S., & Brata, A. H. (2019). Pengembangan sistem aplikasi persuratan elektronik berbasis web di fakultas ilmu komputer (FILKOM) Universitas Brawijaya. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(5), 5115–5121. http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/5424/2545

Wijaya, M. R. R., Nurwarsito, H., & Sagita, D. (2018). Pengembangan Perangkat Lunak Dokumentasi Persuratan Menggunakan Codeigniter dengan Semantic Web Pada Unit Kerja Inspektorat Jenderal Kemnaker. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer E-ISSN*, *2*(6), 2343–2352