International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

Analysis of Offline-First App Technology in Raspberry Pi Edge Computing for Post-Disaster Hospital Situation

Zulkifli Tahir[1], Muhammad Niswar[2] Wardi[3], Andi A.P. Alimuddin[4], Tyanita Puti Marindah Wardhani[5], & Laura N. Nainggolan[6]

1,2,4,5,6 Department of Informatics, Faculty of Engineering, Universitas Hasanuddin, Indonesia

3Department of Electrical Engineering, Faculty of Engineering, Universitas Hasanuddin, Indonesia

(zulkifli, niswar, wardi, aisprayogi, tyanitaputi)@unhas.ac.id, nainggolan15d@student.unhas.ac.id]
**Corresponding author: Zulkifli Tahir*

Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

Abstract: The occurrence of disasters has a significant influence on hospitals. The accessibility of the hospital emergency information system may be compromised due to an inconsistent network connection. In order to address this requirement, it is necessary to develop a website system that is capable of functioning in situations where the network connection is limited or unavailable, sometimes referred to as offline functionality. Progressive Web Apps (PWA) is the theory of creating web-based applications that are user friendly, even in situations where the internet connection is limited or there is no internet connection at all by utilizing Service Workers. This offline functionality solution is achieved by combining Service Workers technology with React JavaScript for frontend development and cloud Firestore for backend data storage. This research conducts a comparative analysis between online programs versus programs that utilize Service Workers. Additionally, this device incorporates a Raspberry Pi server to support the edge computing architecture. The findings indicate that web Service Workers exhibit a notably quicker average access time and average browser processing time in comparison to their counterparts. The throughput test indicate that the inclusion of the integrated web Service Workers leads to a higher volume of requests being generated within a certain time period compared to the scenario where the Service Workers is not utilized. The hospital emergency information system has demonstrated its capacity to operate consistently and efficiently by leveraging various technologies.

Keywords: Disaster, Hospital Emergency Information System, Progressive Web Apps, Edge Computing, Offline web system

1. Introduction

According to the global Risk Report 2023, Indonesia is in second place among the countries most at risk of disasters among 193 countries. This ranking is based on the global risk index, which measures the vulnerability of a country to various types of disasters. In the case of Indonesia, its world risk index is recorded at 41.46 [1]. In a comprehensive manner, the National Agency for Disaster Management in Indonesia, classifies ten distinct types of disaster risks prevalent in the country. These include earthquakes, tsunamis, volcanic eruptions, floods, flash floods, landslides, droughts, forest and land fires, extreme weather events, and extreme waves [2]. Indonesia has garnered significant international attention in the past two decades due to the severe catastrophes that have caused extensive destruction to its geographical area [3].

Disaster occurrences give rise to diverse forms of damage, such as harm to electricity infrastructure, the destruction of the Base Transceiver Station (BTS) network, and the occurrence of blackouts or disconnections in the communications network. The occurrence of the disaster had a detrimental impact on the Hospital. Hospital systems will experience limited accessibility after a disaster due to factors such as disrupted network connectivity, or even loss of internet connectivity, and problematic electrical infrastructure. This is the importance of addressing public health problems efficiently through the use of an effective health information system that is able to manage data efficiently after a disaster occurs [4]. Therefore, technological advances are needed that

E-mail:author's email

can facilitate development systems that are able to effectively overcome the challenges that arise after disasters.

Previous studies have demonstrated successful implementation of a dependable offline web system suitable for utilization by small and medium-sized industries [5][6] and by e-commerce system [7][8]. Offline solutions for adaptive distance learning have been utilized by other researchers as well [9]. In the present study, the implemented solution utilizes offline-first app technology, employing the PWAs with Service Workers, React JavaScript, Node JavaScript frameworks, and both online-offline Firebase API databases. Our proposed system has been built using Edge Computing architecture. These technologies are combined and utilized to increase the resilience of hospital emergency information systems after a disaster occurs. By utilizing this technology, website speed can be increased while accessibility can be maintained in conditions of unstable or even interrupted internet connectivity.

PWAs represent a contemporary phenomenon within the notable progression of web applications [10]. It refers to web apps that leverage contemporary web capabilities in order to offer users an experience similar to that of a native app [11]. PWAs is equipped with a web application manifest, which enables them to be easily integrated and accessed immediately from the home screen of a user's device [12]. PWAs is also utilized for the purpose of developing cross-platform software that is compatible with various platforms, including personal computers, tablets, smartphones, and other similar devices for agricultural and health systems [13][14]. Several widely used frontend frameworks, such as React, Preact, Vue, Angular, and Ionic, include built-in support for PWAs [15]. Furthermore, PWAs exhibit the advantageous characteristic of swift loading times, irrespective of the prevailing network conditions. The ability to cache material for offline access and provide push notifications is made possible by the assistance of Service Workers [16].

Service Workers is a category of Web Workers, which are scripts that operate in the background of a user's web browser [17]. Service Workers are JavaScript files that operate on a separate thread from the main thread running the browser. It is responsible for managing network requests, caching data, retrieving resources from the cache, and facilitating the transmission of push notifications [18]. Web assets have the capability to be kept as a local cache, so enabling users to access a satisfactory experience even in situations where the internet network is insufficient. Applications have the capability to persistently store and utilize previously accessed online pages, as well as offer information regarding the condition of the internet connection, even in the absence of an active browser session, hence avoiding the occurrence of error messages [19]. Service Workers ensure that they operate on the HTTPS protocol for secure connections. This system has the ability to handle network connection requests originating from the browser and provide responses using cache data [20].

React JavaScript is a frontend library that was created by Facebook with the purpose of aiding web developers in the creation of interactive, stateful, and user-friendly user interface (UI) components. React JS is widely regarded as the most optimal library for efficiently creating intricate UI, ensuring exceptional performance. React JS uses basic principles with a component-based architecture [21]. This library also supports the use of PWA technology [15].

Node.js is a software framework developed for creating web-based applications using Java Script programming syntax. Node.js is specifically designed to enhance the functionality of JavaScript by allowing it to function as a programming language that runs on the server side. Node.js includes its own HTTP library server, allowing it to act as a web server [22]. So, with Node.js which functions as the backend it can be run with React which functions as the frontend [23].

IndexedDB is a type of NoSQL transactional database system that facilitates efficient retrieval of persistent data by utilizing JavaScript Object Notation (JSON) objects. The functionality of this system can be executed by utilizing JavaScript code, hence enhancing its usability for web browsers. IndexedDB offers a significant benefit in terms of its substantial storage capacity, which commences at 50 MB of data allocation for every origin [24]. IndexedDB is a method for gathering and organizing data through a web application that possesses offline capabilities [25].

The Firebase Realtime Database is a cloud-hosted NoSQL database. The real-time database idea facilitates the synchronization of interconnected devices and maintains accessibility even in the absence of network connectivity through the utilization of a local cache. The Firebase Realtime Database was designed within the framework of Google's infrastructure to enable developers to prioritize the development of high-quality applications that facilitate seamless movement of both offline and online data [26].

Edge computing refers to a distributed computing framework in which data, computing resources, storage, and applications are situated in a location at the edge of network. This system is situated either in proximity to the user or in proximity to the data. This paradigm places computing resources near areas where they are needed, minimizing the distance data is sent over the network and can result in increased overall network efficiency and performance [27]. An edge computing prototype with Raspberry Pi has been used as an edge server in this research. In certain cases, this strategy of using a Raspberry Pi shows superior performance when compared to using a cloud computing system [28].

2. METHODOLOGY

The web-based system that has been developed is known as the hospital emergency information system. The primary objective of this system is designed to provide real-time information and facilitate effective communication among healthcare professionals inside hospital settings. The developed system would possess the capability to gather patient data in challenging network conditions or in offline scenarios through the utilization of PWAs technology.

The illustration presented in Figure. 1 provides a visual representation of the overall structure and layout of the hospital system that has been developed. The system has been implemented on a Raspberry Pi 4 device, serving as a web server. The system has been developed utilizing React JS, node.js, firebase, and PWAs. Node.js is a widely employed platform for developing projects, while Nginx is commonly utilized as a web server. The initial step is utilizing the create-react-app command to generate a React JS project. To construct a Service Workers, it is necessary to install the JavaScript library and the workbox-webpack node module. Additionally, the Firebase database is installed. The solution, which has been constructed and afterwards deployed in nginx as a web server, facilitates web access on the client side. The user has the capability to access the hospital emergency information system web interface by utilizing a web browser, such as Google Chrome or Firefox, on a network that is shared with the

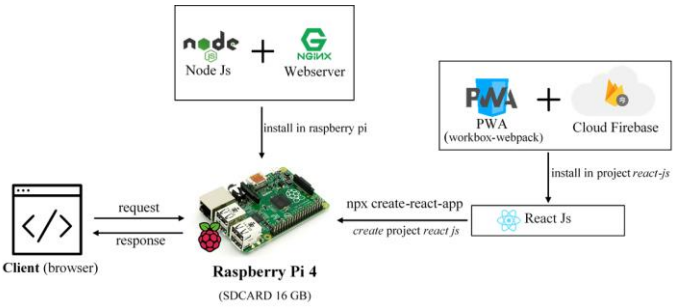

Figure 1. An Overview of the Hospital Emergency System Design

Raspberry Pi web server.

Hospitals employing web technology commonly have a shared challenge, namely, their inability to function in the absence of server or internet network connectivity. The utilization of Service Workers enables the implementation of an offline first app model, hence employing web technologies. The developed system would possess the capability to operate both in online and offline modes.
The utilization of local databases with NoSQL database, such as Local Storage and IndexedDB, for offline database management, along with the integration of Firebase API as a backend service for online database management. The two databases will undergo periodic synchronization, contingent upon the presence of an internet connection.

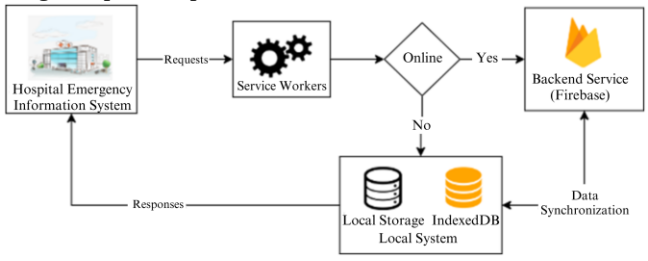

Figure 2. Data Flow System

The system was developed in accordance with the flow depicted in Figure. 2. The web application will initiate data requests that will be received by the Service Workers operating on the client side. Subsequently, the Service Workers will verify the operational status of the website, determining whether it is now accessible or inaccessible. In the context of a system that runs online, user requests will be sent to the database server, in this case using Firebase, which is located on the original server. Synchronization will occur between data stored in the local database and data stored in the server database, depending on whether there is an internet connection or not. Subsequently, the retrieved request outcomes will be transmitted back to the web-based application for presentation within the browser interface. In the event of the internet being inaccessible, Service Workers assume a

crucial function. The service worker will be operational and deliver a return value in the form of data that had been stored in the cache during the initial execution of the system. Service Workers retrieve data from the local database and subsequently provide a response to the website, which is then displayed in the web browser.

3. IMPLEMENTATION AND TESTING

This section details the execution and creation of the previously planned system utilizing React Js, Node.js, Service Workers, and the Firebase API Server database technology. This system is then run on Raspberry Pi hardware which functions as a web server. The first test includes performance testing, which specifically focuses on response time, throughput, and testing web browser parameters. Next, the second test includes evaluating the feasibility of the PWA concept using Lighthouse.

A. Hospital Emergency Information System Implementation

The process of analyzing system requirements was conducted by interviewing the officers from the Management Information Systems department at Hasanuddin University Teaching Hospital in Makassar, Indonesia. Based on the findings derived from the conducted interviews, it was determined that the implementation of a hospital emergency information system is important in order to ensure post-disaster accessibility. The interviews yielded the necessary data for the development of a hospital emergency information system, specifically encompassing patient forms and medical records.

The author developed online design for a hospital emergency information system utilizing the React Js library to generate code scripts. Additionally, Service Workers technology was employed to enhance web accessibility in situations of unreliable or unavailable network connectivity.

The database structure employed by the hospital emergency information system website. The research utilizes a NoSQL database, specifically Cloud Firestore from Firebase. Cloud Firestore is a data storage system that organizes data in the form of documents, which are further sorted into collections. Every document is a form of data that is characterized by its lightweight nature and consists of columns that correspond to specific values. Two distinct sets are utilized in this study, specifically patients and medical records.

The system depicted in Figure. 3 is constructed utilizing the React Js library and seamlessly incorporated with a service worker. The system is capable of functioning both online and offline, enabling it to perform tasks such as data entry, data modification, data display, data deletion, and payment processing even in the absence of an internet connection.

B. Testing Scenarios

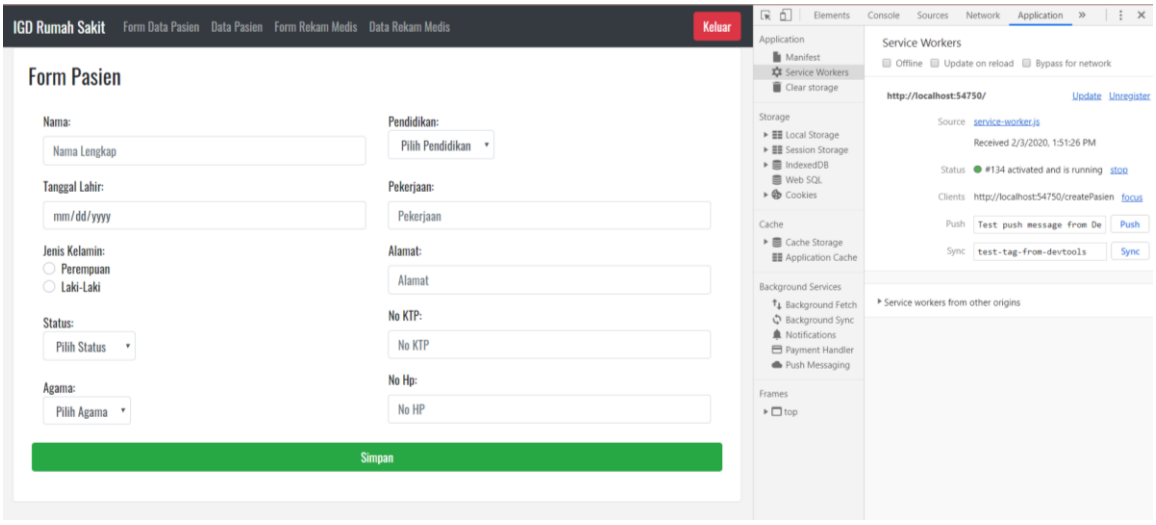

Figure 3. Service Workers in web application

Test scenarios are used to assess and obtain performance results from hospital emergency information system applications that use PWA technology. The test was carried out on a Raspberry Pi device with the following specifications:

- SoC : Broadcom BCM2711 Cortex-A72 64Bit SoC @1.5 GHz
- GPU : Videocore VI @500 MHz
- RAM : 4 GB
- SdCard : 16 GB

The performance testing parameters used include response time, throughput, and execution of browser operations on the client side. The evaluation of response time and throughput is conducted by employing the

Apache JMeter application and the Google Chrome extension. Additionally, the examination of browser processes is carried by utilizing the DevTools feature provided by Google Chrome. The present study aims to evaluate the feasibility of PWAs by utilizing the Lighthouse addon within the Google Chrome browser.

C. Performance Testing Scenarios

The performance test in this study involves two hospital emergency information system websites that do not employ Service Workers, as well as another website that does utilize Service Workers. This system operates on a Raspberry Pi device that serves as a web server. Client-side performance experiments are conducted. The testing encompassed the assessment of average response times using extensions in Google Chrome and Apache JMeter software, along with the evaluation of throughput using Apache JMeter software.

1) Average Response Time Testing

The average response time refers to the calculation of the sum of the time intervals between client-server requests, divided by the total number of requests as expressed by (1).

$$Average = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{1}{n}(a_1 + a_2 + \cdots + a_n) \quad (1)$$

n : Number of requests from client to server.
i : Counter number (i = 1,2,…,n) server
a : Time interval for each request by the client

A comparative analysis was conducted to determine the average response time of systems employing service workers vs those without Service Workers. This experiment employs Google Chrome as the web browser, utilizing the Google Chrome extension called Open Multiple URLs. This addon allows for the simultaneous opening of several tabs with the identical URL. The Reload All Tab extension enables the simultaneous reloading of all currently open URL tabs, while the page load time extension offers a method to revert response timings. Conduct an experiment by sequentially opening URL tabs ranging from 1 to 65. The process of testing involves the utilization of two distinct websites, with the first website employing a Service Workers, while the second website without the presence of a Service Workers. Testing is done on the client side. Response time evaluation also uses Apache JMeter software, which combines a ramp up duration of 1 and a number of loops of 1. The threads used come from one user and are in multiples of 100, ranging from 100 to 1100 threads. The response time value for each test thread is determined by performing 10 repetitions, and then calculating the average of the recorded response times.

D. Throughput Testing

Throughput is a parameter used to analyze the performance of a web system. Throughput refers to the quantity of accesses that can be executed within a certain time period, usually measured in Transactions Per Second (TPS). Throughput calculation using equation (2).

$$Throughput = \frac{Number\ of\ requests}{Average\ time\ x\ number\ of\ requests} \quad (2)$$

Evaluation of web system throughput is carried out both for systems that employ Service Workers and those that do not. Utilization of Apache JMeter software is used in the testing process, where a ramp up period of 1 and number of loops is implemented 1. Threads are started by one user and increase in multiples of 100, starting from 1 thread to a maximum of 1100 threads. The throughput value for each test thread is determined by performing 10 repetitions, and then calculating the average throughput. The purpose of this assessment is to determine the quantity of transactions that can be executed within a certain time period on the Service Workers website according to the specified number of threads.

E. Browser Testing

The load time of page resources can be measured by utilizing Google Chrome dev-tools. The Chrome browser is capable of generating a comprehensive inventory of server-based resources, including JavaScript, HTML, and CSS. Additionally, it offers a visual representation of the access time for each resource through the use of a timeline graphic model. The duration required for data retrieval from the server to the browser is represented by a numerical value known as DomContentLoaded. Similarly, the duration for the complete retrieval of data from the server to the browser, indicating the successful loading of the web page, is denoted by another numerical value called Load.

The browser testing procedure involves the evaluation of website load times in the presence and absence of service workers. This is accomplished by enabling and disabling the cache feature in Google Chrome dev-tools. The experiment was conducted by accessing each webpage a total of 10 times. Subsequently, a computation is performed to determine the mean value for each conducted experiment. The dev-tools display for evaluating DomContentLoaded and Load is depicted in Figure. 4. The evaluation of cache functionality can be conducted by using the "disable cache" option to prevent cache activation, and disabling the "disable cache" option to enable cache functionality.

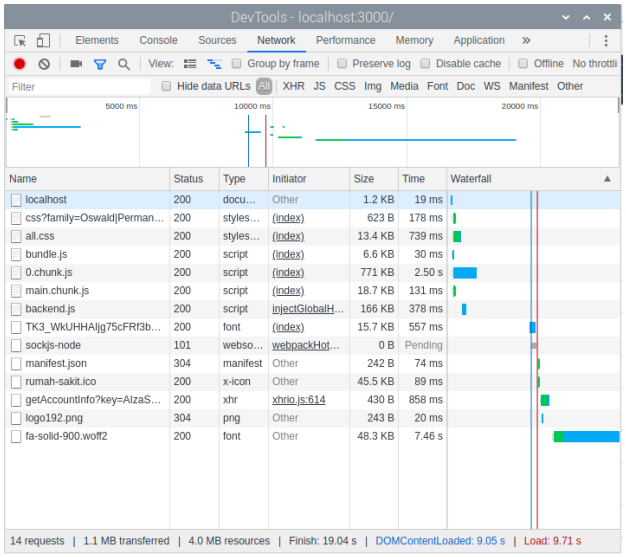

Figure 4. Chrome's Dev-tools Display

F. PWAs Concept Testing

In order to assess non-functional needs, the testing process involved the utilization of the Google Chrome extension known as Lighthouse. An experimental investigation was conducted using a Raspberry Pi device. Lighthouse is an open-source program developed by Google that serves as a tool for evaluating the PWAs idea. It encompasses various dimensions like performance, accessibility, adherence to best practices, search engine optimization (SEO), and the implementation of PWAs features. The Lighthouse tool has the capability to operate within the user's web browser by means of a Google Chrome extension. Experiments conducted with the utilization of Lighthouse are executed for a singular trial on an individual client browser.

4. RESULTS AND DISCUSSION

A. Performance Results

TABLE I. Response time data with Google Chrome Extension

Number of Requests	With Service Workers	Without Service Workers
1	1.16	1.58
5	1.51	1.81
10	1.82	2.40
15	2.87	3.07
20	4.08	4.16
25	5.11	5.26
30	6.03	6.43
35	6.38	6.57
40	6.47	6.71
45	7.08	7.29
50	7.28	7.75

Response time testing when accessing a website homepage with and without Service Workers on a Raspberry Pi device was obtained using the Google Chrome extension. The average response time results obtained for each experiment are presented in Table I and depicted in Figure. 5. The findings from this research show that the average access time for websites equipped with Service Workers is faster than websites without Service Workers. This is because Service Workers are able to manage website asset caches efficiently, enabling faster performance and incorporating image optimization capabilities that can reduce bandwidth usage.

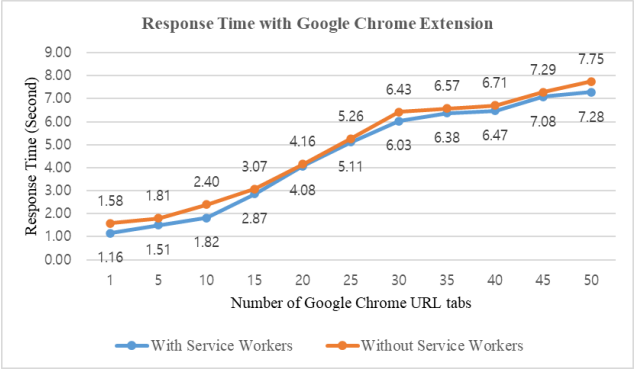

Figure 5. Response time graph with Google Chrome Extension

The outcomes of the response time experiment conducted on a Raspberry Pi device, involving the access of web pages with and without Service Workers using Apache Jmeter software, will yield results for each trial. These results will include the average response time, which will be presented in Table II and Figure. 6. Average access time testing is carried out on each page from 1 to 1100 threads. The findings show that the average access time for websites equipped with Service Workers is faster than websites that are not equipped with Service Workers. The presence of Service Workers in a web application leads to a reduction in file size, resulting in better response times when compared to web applications that do not have Service Workers.

TABLE II. Response time data with Apache JMeter

Number of Threads	With Service Workers	Without Service Workers
1	7.3	11.1
100	8.4	13.1
200	13.9	19.0
300	23.0	54.9

400	25.3	57.1
500	50.1	117.9
600	52.1	119.5
700	371.0	401.3
800	404.1	454.0
900	442.2	472.2
1000	511.3	567.7
1100	514.0	606.7

A significant increase in response time occurs when the number of threads is between 600 and 700. This phenomenon is caused by the Raspberry Pi CPU reaching its maximum utilization capacity. Thus, it can be concluded that the Raspberry Pi device is reliable in accepting up to approximately 600 threads.

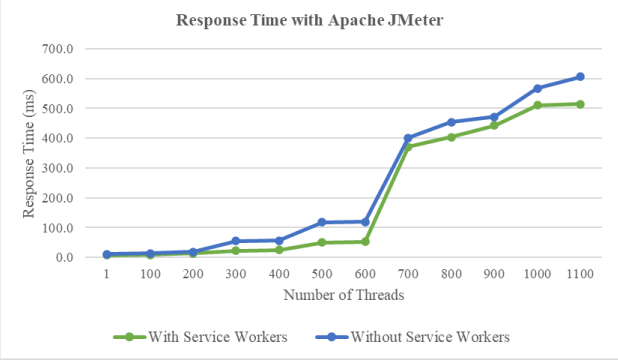

Figure 6. Response time graph with Apache JMeter

Throughput testing for visiting web pages with and without Service Workers was carried out on a Raspberry Pi device using Apache JMeter software. The results of each experiment are presented in Table III and Figure. 7. The testing process involves evaluating each page in a range of 1 to 1100 threads.

TABLE III. Throughput data

Number of Threads	With Service Workers	Without Service Workers
1	139	96.01
100	100.4	93.72
200	199	194.16
300	327.6	316.43
400	328.7	323.88
500	480	434.12
600	483.5	464.57
700	559.6	481.94
800	576.1	509.85
900	545.6	525.19
1000	574.6	540.68
1100	588.8	523.45

The experimental graph indicates that the throughput of a web application without Service Workers is comparatively lower than that of a web application utilizing Service Workers. However, the disparity in throughput may not be substantial. Web applications that utilize Service Workers demonstrate enhanced throughput, enabling them to process a greater number of requests within a specific timeframe.

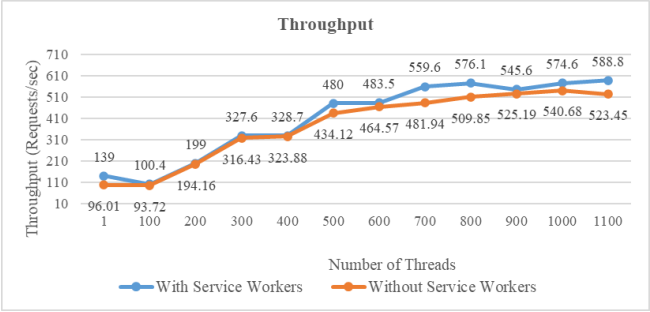

Figure 7. Throughput graph with Apache JMeter

The figures presented in Figure. 8 and Figure. 9 illustrate the test outcomes of browser processing on a Raspberry Pi web server, comparing a web without a Service Workers and a web with a Service Workers. The two parameters examined in this research are DomContentLoaded and Load. The experiment was carried out with a series of tests by visiting the internet without using cache. Each test was repeated ten times to obtain a total of ten trials. The average time required for the browser is calculated to access the online system. After making ten attempts to access the web without using the cache, continue visiting the web by storing the cache in the browser for ten additional attempts as well.

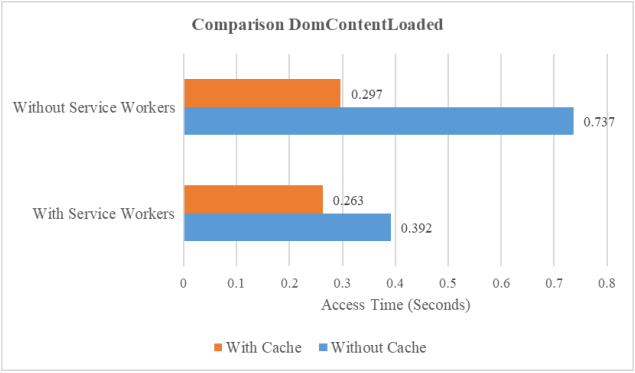

Figure 8. DomContentLoaded Access Time Comparison Chart

The findings indicate that the average DomContentLoaded and Load time for each website were significantly longer when accessed without enabling the cache. The browser will retrieve data directly from the server when accessing a website without using cache and the time required for this process depends on the speed of the internet connection used and the amount of data on the server.

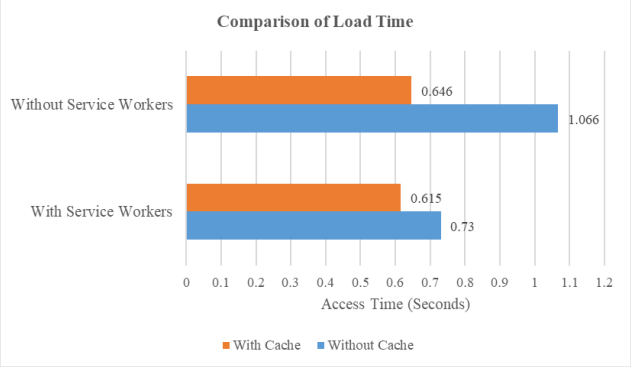

Figure 9. Load Access Time Comparison Chart

Activating the cache can significantly reduce access times for DomContentLoaded and load events on the website. The data retrieval process can not only be done on the server, but can also utilize the browser cache to store and access data more efficiently.

Websites that utilize Service Workers demonstrate improved access times for DomContentLoaded and Load events in comparison to websites that do not employ Service Workers. The Service Workers script executes in the background, which is different from the usual behavior of JavaScript. Nevertheless, the disparity in access time between the two is negligible, as both were developed using React JS. React JS is a JavaScript package that employs a Single Page Application (SPA) architecture. This means that after the initial page load from the server, subsequent page transitions do not necessitate another refresh to display the new page. The system exhibits enhanced performance by eliminating the need for constant page loading.

Data collection for offline access time commenced from the second experiment. The initial procedure is conducted through online means for the purpose of asset storage. The offline data retrieval process is executed through the activation of the cache, which is repeated ten times. The acquired results for offline web access with a Service Workers are the Load and DomContentLoaded events, as seen in Figure. 10.

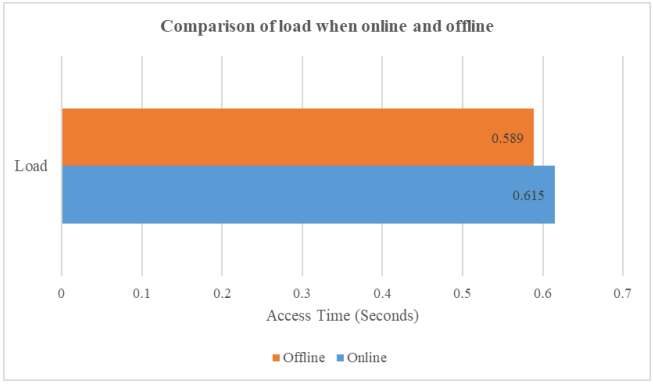

Figure 10. Load access time offline and online with Service Workers

In the context of an online Service Workers web condition, the initial loading process results in increased loading times. During the initial load process, all asset files are processed and stored in the storage cache, resulting in significant additional latency. The subsequent procedure will facilitate the system's continued accessibility in the absence of a network connection.

In offline conditions, the loading time for data during web access is observed to be faster compared to online conditions. This phenomenon occurs due to the Service Workers' ability to get assets from the cache and database stored in local storage when browsing the web in an offline environment. In the context of online access to the web, the Service Workers functions by retrieving data from the server, caching it, and then storing it in both local storage and the server. The duration of loading time experienced while online activities is comparatively extended.

B. PWAs Concept Feasibility

The evaluation of PWAs through concept testing is classified as a non-functional need. The examination employs the Lighthouse addon, specifically designed for Google Chrome. Lighthouse is an open-source application developed by Google that serves as a tool for evaluating several elements of PWAs, including their performance, accessibility, adherence to best practices, and SEO. Lighthouse is a Google Chrome extension that operates within the user's web browser.

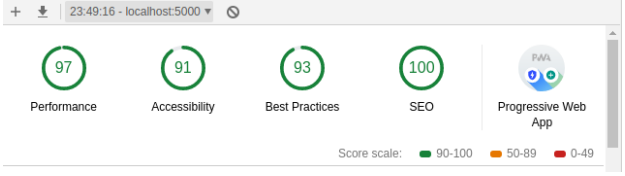

Figure 11. Test Results with Lighthouse

Resulting in the test shown in Figure. 11 on a web-based hospital emergency information system. The Lighthouse test gave positive results for the built web

application, as it achieved commendable scores across four different criteria. Specifically, the app demonstrated strong performance with an average score of 97 out of 100, demonstrated good accessibility with an average score of 91 out of 100, adhered to best practices with an average score of 93 out of 100, and excelled in terms of SEO. with a perfect score of 100 out of 100.

5. Conclusion

The performance investigation was conducted to compare websites that utilize Service Workers with those that do not, employing the Apache Jmeter tool and Google Chrome developer tools. Additionally, the notion of PWAs was examined using the Lighthouse tool on Google Chrome. The obtained results indicate the following:

The utilization of PWAs enables the hospital emergency information system website to function offline, allowing for the storage, modification, deletion, and retrieval of data from the database even when not connected to the internet.

The present study examines the disparity in response time and throughput between web testing conducted with Service Workers and web testing conducted without Service Workers on the Raspberry Pi web server. The utilization of Service Workers leads to enhanced response times and increased throughput compared to websites that do not employ Service Workers. In order to enable efficient processing of substantial requests within a specified timeframe, it is imperative to incorporate Service Workers into the web architecture.

According to the conducted experimentation on the Raspberry Pi web server, it has been observed that websites incorporating Service Workers exhibit enhanced load access speed in comparison to websites lacking Service Workers. The Service Workers process occurs in the background, thereby improving web performance and facilitating cache management.

The results of the test findings carried out using Lighthouse on Raspberry Pi devices obtained Performance criteria with an average score of 97, Accessibility criteria with an average score of 91, Best Practice criteria with an average score of 93, and SEO criteria with an average score of 93. and SEO with a full score of 100.

The findings of this study demonstrate that the utilization of these technologies offers numerous benefits and has the potential to be integrated into hospital emergency information systems. In our forthcoming study, we intend to incorporate various cutting-edge technologies, including the Web Speech API, WebRTC, WebBluetooth, and others [29]. In the future, our research will also include exploration of system functional improvements aimed at increasing the resolution and performance speed of edge computing devices in disaster situations, including by utilizing cluster systems, implementing solar panel electrical connections, and integrating satellite internet connections.

6. Acknowledgement

The present study received financial support from the Basic Research for Higher Education Excellence or Penelitian Dasar Unggulan Perguruan Tinggi (PDUPT) Grant.

7. References

[1] I. A. Frege *et al.*, *WorldRiskReport 2023*. 2023.

[2] Badan Nasional Penanggulangan Bencana (BNPB), "Peraturan Kepala Badan Nasional Penanggulangan Bencana Nomor 9 Tahun 2008 tentang Prosedur Tetap Tim Reaksi Cepat Badan Nasional Penanggulangan Bencana," 2008.

[3] J. Widagdo, D. Putra, B. Syihabuddin, T. Juhana, E. Mulyana, and A. Munir, *Android-based Disaster Management Application for After-Disaster Rapid Mobile Assessment*. 2021. doi: 10.1109/IoTaIS50849.2021.9359695.

[4] E. Aung and M. Whittaker, "Preparing routine health information systems for immediate health responses to disasters," *Health Policy Plan.*, vol. 28, no. 5, pp. 495–507, 2013, doi: 10.1093/heapol/czs081.

[5] Z. Tahir, A.-R. Dasmito, Adnan, M. Niswar, and Wardi, "A Reliable Offline Web System for Small and Medium Industries," *MATEC Web Conf.*, vol. 331, p. 06007, 2020, doi: 10.1051/matecconf/202033106007.

[6] Z. Tahir, A. A. Ilham, A. P. Alimuddin, M. Z. A. Suyuti, and Charina, "Reliable and Low-Cost Digital Transformation Technology Using Progressive Web Apps in Fog Computing Architecture for Small and Medium Industries in Indonesia BT - Advances in Internet, Data & Web Technologies," L. Barolli, E. Kulla, and M. Ikeda, Eds., Cham: Springer International Publishing, 2022, pp. 163–174.

[7] Z. Tahir, A. A. Ilham, M. Niswar, A. Adnan, and A. Fauzy, *Progressive Web Apps Development and Analysis with Angular Framework and Service Worker for E-Commerce System*. 2021. doi: 10.1109/ICOCO53166.2021.9673557.

[8] E. Eunike, R. Sanjaya, and A. D. Widiantoro, "Application of Progressive Web Apps (PWA) on PT SKA's E-Commerce Website," *J. Bus. Technol.*, vol. 3, no. 1, pp. 14–20, 2023, doi: 10.24167/jbt.v3i1.5263.

[9] M. Pikuliak, I. Lazarovych, and M. Usyk, "Progressive web technology-based improvement

of the distance learning adaptive system," *Sci. J. Ternopil Natl. Tech. Univ.*, vol. 105, no. 1, pp. 118–127, 2022, doi: 10.33108/visnyk_tntu2022.01.118.

[10] M. Squarcina, S. Calzavara, and M. Maffei, "The Remote on the Local: Exacerbating Web Attacks Via Service Workers Caches," *Proc. - 2021 IEEE Symp. Secur. Priv. Work. SPW 2021*, pp. 432–443, 2021, doi: 10.1109/SPW53761.2021.00062.

[11] R. Wijaya, P. Crisgar, M. Pakpahan, E. Syamsuddin, and M. Hasanuddin, *Implementation of Motor Vehicle Tracking Software-as-a-Service (SaaS) Application Based on Progressive Web App*. 2021. doi: 10.1109/ISESD53023.2021.9501600.

[12] D. M. Case, C. Steeve, and M. Woolery, "Progressive Web Apps are a Game-Changer! Use Active Learning to Engage Students and Convert Any Website into a Mobile-Installable, Offline-Capable, Interactive App." pp. 1396–1396, 2020. doi: 10.1145/3328778.3367007.

[13] I. Bekiaris *et al.*, *GRETA: Pervasive and AR Interfaces for Controlling Intelligent Greenhouses*. 2021. doi: 10.1109/IE51775.2021.9486584.

[14] A. Cesario *et al.*, "Development of a Digital Research Assistant for the Management of Patients' Enrollment in Oncology Clinical Trials within a Research Hospital.," *J. Pers. Med.*, vol. 11, no. 4, Mar. 2021, doi: 10.3390/jpm11040244.

[15] V. K. Kotaru, *Service Workers," in Building Offline Applications with Angular*. 2022.

[16] T. A. Majchrzak, A. Biørn-Hansen, and T. M. Grønli, "Progressive web apps: The definite approach to cross-platform development?," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2018-Janua, pp. 5735–5744, 2018, doi: 10.24251/hicss.2018.718.

[17] P. Chinprutthiwong, R. Vardhan, G. Yang, Y. Zhang, and G. Gu, "The service worker hiding in your browser: The next web attack target?," *ACM Int. Conf. Proceeding Ser.*, pp. 312–323, 2021, doi: 10.1145/3471621.3471845.

[18] K. Subramani, J. Jueckstock, A. Kapravelos, and R. Perdisci, "SoK: Workerounds - Categorizing Service Worker Attacks and Mitigations," *Proc. - 7th IEEE Eur. Symp. Secur. Privacy, Euro S P 2022*, pp. 555–571, 2022, doi: 10.1109/EuroSP53844.2022.00041.

[19] S. Ali, C. Grover, and R. Chaudhary, "Progressive Web Apps (PWAs)---Alternate to Mobile and Web," in *Emerging Technologies in Data Mining and Information Security*, P. Dutta, S. Chakrabarti, A. Bhattacharya, S. Dutta, and V. Piuri, Eds., Singapore: Springer Nature Singapore, 2023, pp. 565–576.

[20] L. Karavashkin, S. Molodyakov, and B. Medvedev, "Caching Data in a Web Audio Service Using Progressive Web Apps Technologies," in *Cyber-Physical Systems and Control II*, D. G. Arseniev and N. Aouf, Eds., Cham: Springer International Publishing, 2023, pp. 372–380.

[21] P. S. Maratkar and P. Adkar, "React JS – An Emerging Frontend Javascript Library Virtual DOM React One-Way Data Flow JSX Syntax," vol. 4, no. 12, pp. 99–102, 2021.

[22] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *Internet Comput. IEEE*, vol. 14, pp. 80–83, Jan. 2011, doi: 10.1109/MIC.2010.145.

[23] A. M. Vukicevic, M. Djapan, M. Stefanovic, and I. Macuzic, "SafE-Tag mobile: A novel javascript framework for real-time management of unsafe conditions and unsafe acts in SMEs," *Safety Science*, vol. 120. pp. 507–516, 2019. doi: 10.1016/j.ssci.2019.07.024.

[24] F. Paligu and C. Varol, "Browser forensic investigations of whatsapp web utilizing indexeddb persistent storage," *Futur. Internet*, vol. 12, no. 11, pp. 1–17, 2020, doi: 10.3390/fi12110184.

[25] U. S. Nagarakshitha, B. R; Lohith, K. S ; Aarthy, K. P; Gopkumar, Arjun; Ranjan, "Application of NoSQL Technology to Facilitate Storing and Retrieval of Clinical Data Using IndexedDb in Offline Conditions," *J. Comput. Theor. Nanosci.*, vol. 17, no. 9–10, pp. 4012–4015, 2020.

[26] M. Ohyver, J. V. Moniaga, I. Sungkawa, B. E. Subagyo, and I. A. Chandra, "The comparison firebase realtime database and MySQL database performance using wilcoxon signed-rank test," *Procedia Comput. Sci.*, vol. 157, pp. 396–405, 2019, doi: 10.1016/j.procs.2019.08.231.

[27] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An Overview on Edge Computing Research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020, doi: 10.1109/ACCESS.2020.2991734.

[28] T. Gizinski and X. Cao, "Design, Implementation and Performance of an Edge Computing Prototype Using Raspberry Pis," *2022 IEEE 12th Annual Computing and Communication Workshop and Conference, CCWC 2022*. pp. 592–601, 2022. doi: 10.1109/CCWC54503.2022.9720848.

[29] A. Talukder and R. Haas, *AIoT: AI meets IoT and Web in Smart Healthcare*. 2021. doi: 10.1145/3462741.3466650.

Dr. Zulkifli Tahir is an Assistant Professor in the Department of Informatics, Faculty of Engineering, Universitas Hasanuddin, Indonesia. His research interests lie in the areas of Web Technologies including progressive web apps, web server load balancing, and digital transformation for small and medium industries; Distributed System, Fog Computing and Internet of Things (IoT), with a focus on smart homes, solar panel monitoring, and resource allocation; and Machine Learning and Artificial Neural Networks for applications in maintenance decision support systems, industrial machine fault detection, and image processing.

Dr. Muhammad Niswar is an associate professor at the Department of Informatics, Faculty of Engineering, Hasanuddin University, Indonesia. He graduated Bachelor of Engineering in Electrical Engineering (1997) from Hasanuddin University, Indonesia. He then received a Master of Information Technology (2001) and Ph.D. in Information Science (2010) from the University of Newcastle, Australia, and Nara Institute of Science and Technology, Japan, respectively. His research interests includes computer networking, internet of things, cloud computing, cybersecurity, and artificial intelligence.

Dr. Wardi is Dr. Wardi is Associate Professor in the Department of Electrical Engineering, Faculty of Engineering, Universitas Hasanuddin and is renowned for his groundbreaking contributions in the field of telecommunications technology. The primary focus of his study is on many facets of wireless communications, namely the capacity and coverage planning of LTE-A (4.5G) at a frequency of 2300 MHz. His investigations on the performance of routing protocols such as OLSR and BATMAN in Multi-hop Ad Hoc Networks and Mesh Ad Hoc Networks, using Raspberry Pi as a platform, have resulted in noteworthy contributions to the field. Dr. Wardi has created a portable IP-based communication system that utilizes Raspberry Pi as an exchange. This demonstrates his inventive approach to practical telecoms solutions.

Andi Ais Prayogi Alimuddin is a Lecturer in the Department of Informatics, Faculty of Engineering, Universitas Hasanuddin, Indonesia. His research interests lie in the areas of information system security, and educational game development.

Tyanita Puti Marindah Wardhani is junior lecturer in Department of Informatics, Faculty of Engineering, Universitas Hasanuddin. She has varied academic experience in Social Informatics. Her research interest includes disaster risk management, social network and data analysis, big data, and artificial intelligence implemented in multidisciplinary subjects.

Laura Natalia Nainggolan is a Bachelor of Informatics Engineering graduate from Universitas Hasanuddin, Indonesia. Her is qualified in data science and web development, with licenses and certifications in areas such as Data Analysis with Python, Python for Data Science and AI, Tools for Data Science, and HTML Essentials Training. She also holds an EFSET English Language Certificate (C1 Advanced), which indicates advanced proficiency in the language. His other skills include industry knowledge, data analysis, web development, front-end development, and data visualization.