A Reliable Offline Web System for Small and Medium Industries

Zulkifli Tahir[1*], *Al-Riefqy* Dasmito[1], Adnan[1], *Muhammad* Niswar[1], and Wardi[2]

[1]Department of Informatics Engineering, Hasanuddin University, Indonesia
[2]Department of Electrical Engineering, Hasanuddin University, Indonesia

Abstract. The information technology such as web-based systems has been widely implemented in Small and Medium Industries (SMIs) to improve the management and effectiveness. However, the web-based systems still discover the problem when those are applied to the SMIs that located in the rural areas which generally have low network quality. The problem can also occur due to various conditions such as unstable electricity or during a disaster. Based on the problem, the web-based systems should meet the reliability standards so it can continue to be work in every condition. This study proposes a reliable offline web system of React JavaScript with the service worker concept as an effort to overcome the problem. The results not only can improve the performance of web-based systems in various conditions but also encourage other SMIs to use the web-based system for their industrial process.

1 Introduction

The Small and Medium Industries (SMIs) role for the nation's economic growth becomes increasingly important, especially in its role of reducing poverty and unemployment [1]. Therefore, the efforts to develop an appropriate and effective SMIs will have a significant impact on the growth of the nation's economy. The common problems that often found in SMIs are the small capital and the ineffectiveness of the production process. Although the government has tried to provide the development grants, the production process of SMIs are always seen as below expectations [2]. To assist the performance improvement of each unit and the effectiveness of the production process in SMIs, the web-based system has been applied to our previous works and research's [3].

Today the online web-based system is an important factor for SMIs as a condition to be able to survive the latest economic business [4]. The online system must have reliability standards that can work well in every condition found in the SMIs. However, after the online web-based system is spread in SMIs that located in rural areas which generally have low network quality, some access problems still often occur and must be handled.

The utilizations of web-based systems in industries have been increased rapidly. The web-based system is technology that is spread everywhere and has the potential to connect thousands of devices to exchange information anywhere with the Internet connection [5].

* Corresponding author: zulkifli@unhas.ac.id

Today, many researchers have conducted research on web-based systems in many types of industries. Some of them focus on designing and developing web applications [6], [7]. Other researches develop and implement web-based applications on information technology services [8], [9]. Analysis, learning, interpretation, investigation or evaluation of impacts, effectiveness, and potential of web-based systems have been performed by several researchers ([10], [11], [12], [13]).

Based on the study, most researchers in web-based systems concentrate on system analysis and implementation. However, rarely researchers focus on developing technology from the web system itself. Therefore, this study is important because not only develops the previous research that has analysed and implemented the web-based system in SMIs but also focuses on the development of the web-based system technology.

The offline web system with service worker concept is proposed in this study. The system is developed by currently popular front-end concepts, that is React JavaScript library. The concepts are also integrated by the current web technologies such as RxDB, offline web database, and Service Worker API itself. These technologies provide reliability for web-based systems that experiencing slow or unstable Internet connectivity.

The technology is then applied as a new web-based system development to rural SMIs. To achieve the goal, the study is performed on the web-based systems in SMIs. Then conduct the preparations of web-based systems with offline web system and service worker concept. The latest web technologies are studied and analysed and then applied to the current web-based system. The web-based systems development also includes the modelling of logic and programming concepts.

2 The Offline Web Systems

Traditional Web Applications assume that the network is reachable. This assumption covers the platform. HTML documents are loaded over HTTP and traditionally fetch all of their sub-resources via subsequent HTTP requests. This places web content at a disadvantage versus other technology stacks. Because internet connections can be flakey or non-existent, the offline web system is needed. Once the web system works offline, it can be work also whatever network functionality performed, and it will be doing more when the network online.

In our previous research [14], the HTML5 offline system with web worker concept has been applied. The web worker offline system workflow shown in Figure 1. It is programmed by using HTML5 script, JavaScript and multimedia functions. The web-based systems are stored offline with offline application technology, offline database, and web worker. When the web-based system is connected to the web for the first time, the web browser will list the cache manifest file, and download the required resources, and be stored locally. With no network connection, the web system will replace all data to the local system so that it can be processed offline.

The service worker is designed first to redress the balance by providing a web worker context, which can be started by a runtime when navigations are about to occur. The service worker is a new browser feature that provides event-driven scripts that run independently of the web pages. Unlike other workers, service worker can be shut down at the end of the events, note the lack of retained references from documents, and it has access to domain-wide events such as network fetches. Service worker also has scriptable caches. Along with the ability to respond to network requests from certain web pages via script, this provides a way for applications to go offline. Service worker is meant to replace the HTML5 Application Cache. Unlike AppCache, service worker is comprised of scriptable primitives that make it possible for application developers to build URL-friendly, always-available

applications in a sane and layered way. In addition, the service worker is referred to as performance boosters because it saves networks and provides a better user experience [15].

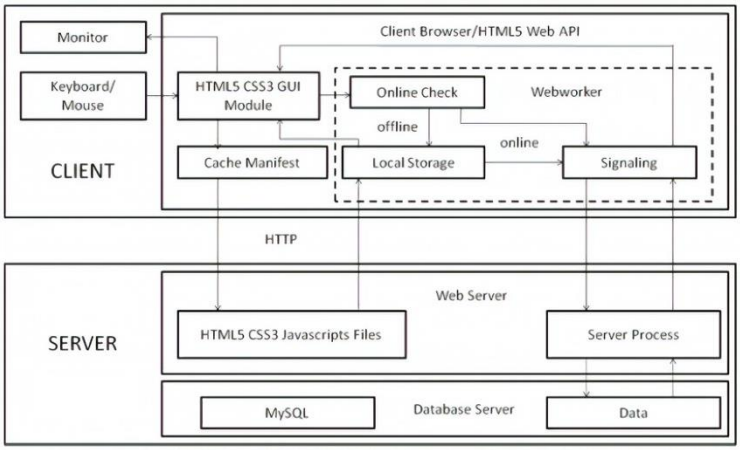

Fig. 1. The web worker offline system workflow

The current study implements a service worker concept workflow shown in Figure 2. The service worker acts as a programmable network proxy, allowing developers to handle how network requests from the web pages. So the developer can take appropriate action based on the availability of the network. The current designed web application with the service worker is depended on two network state. The first state is when the network offline, the web will the files that need to be stored locally, and use local storage and cache to process the web functionality. The storage must be continuously synchronized with a remote data source. The second state's online, the resources and data will be processed directly to the online web server.

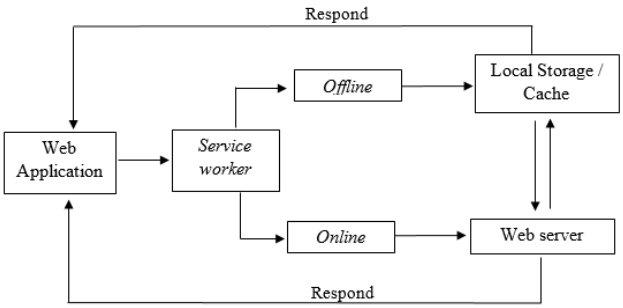

Fig. 2. The service worker workflow

3 The Service Worker with React JavaScript

JavaScript frameworks are developing at an extremely fast pace, meaning that currently there are some front ends have frequently updated versions, such as Angular JS, React JS, and Vue JS. This study implements React JS to operate service worker. React JS is a JavaScript library for building user interfaces [16]. It is maintained by Facebook and a community of individual developers and companies. It was open-sourced at JSConf US in May 2013. When a new project is started by React JS, service worker is invoked by default. The Create React App

can be used to generate PWA with React and RxDB with reactive, client-side, and offline first database that will sync with the server-side CouchDB database. There some dependencies need to be installed after the React application has been created, such as:

1. concurrently. It is performed to run two npm scripts at the same time.
2. moment. It is performed to format the creation date of the message.
3. pouchdb-adapter-http. PouchDB adapter for communicating with an external CouchDB (or CouchDB-like) database.
4. pouchdb-adapter-idb. Pouch DB adapter to use IndexedDB in a browser. The PouchDB (and RxDB) can be used in different environments by just switching the adapter.
5. pouchdb-server. This will work as the server-side database.
6. react-toastify. It is performed to show notifications in the app for the database events.
7. rxdb. This will work as our client-side database.
8. rxjs. RxDB handles data in a reactive way, so it depends on rxjs.
9. serve. When testing the offline functionality, we'll need an HTTP server to serve our app.

In addition, offline support is more advanced at the level of storage abstraction. The data have to be saved into a database which has to figure out how to synchronize it to the backend. Several tools can be utilized in this situation. The Pouch DB synchronizes to a server-side CouchDB. The Realm platform, an end-to-end synchronization solution released by the mobile database product company that does an excellent process for the data synchronization for native apps. Diatomic for Clojure is another stunning engineering technology that can be chosen.

4 Implementation and Analysis

The system is built using the React JS library as the client side and uses RxDB to transfer data to the local database or server. The Figure 3 shows the system main page that has been created using the React JS library and has been integrated with the service worker. The existing features work online and offline. Whether or not there is an internet connection, the application will still work. So that entering data, updating data, and deleting data can be done without requiring an internet connection. The database used for local is the indexedDB database by implementing document objects and couch db for database servers, both of which will synchronize periodically as long as the internet network is available.

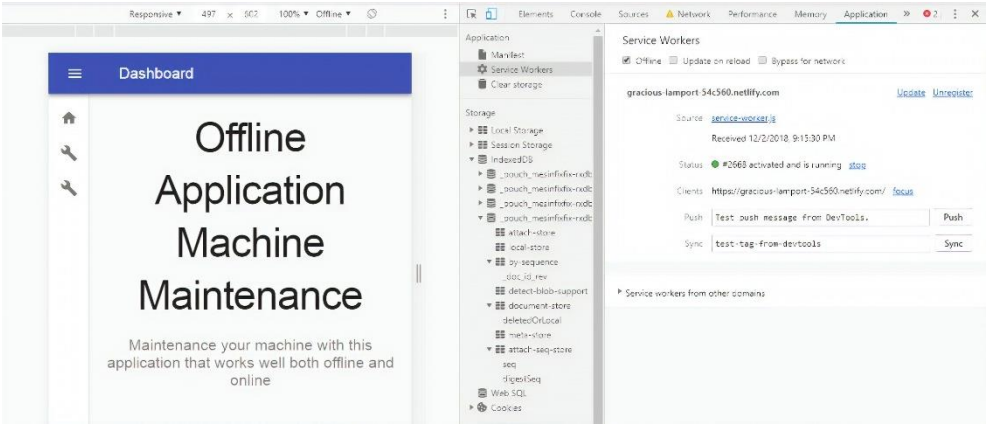

Fig. 3. Service worker on a web application

The analysis was carried out using the response time parameter. The response time from the web using React JS and conventional web (with PHP and MySQL) are calculated. The Apache JMeter application is used to test response time. The JMeter gives the value of the time interval of the request is made until the response is received. Response time analysis is done by giving many requests at once from the client to the server of each thread ten times on each web. After the results are obtained, the average values are calculated. The average time data are shown in Table 1.

Table 1. Average time data

No.	The number of Threads	React JS with service worker (ms)	Conventional Web (ms)
1	1	39	55
2	50	56	96
3	100	102	134
4	150	180	182
5	200	219	279
6	250	230	295
7	300	329	330
8	350	385	405
9	400	392	821
10	450	468	861
11	600	587	1049
12	700	885	1169
13	800	1091	1281
14	900	1278	1768
15	1000	1314	2176

Based on graphs obtained from each web and each network for average time graph in Figure 4, it shows that the average access time on a conventional web for all threads produces a greater number than React JS. The average access time difference between threads 1 to 350 is not too far apart. The average access time difference is more clearly seen if the number of threads is getting bigger. By using 1000 threads, the average time required by React JS is calculated at 1314 ms, whereas with conventional web it takes much longer time with 2176 ms.

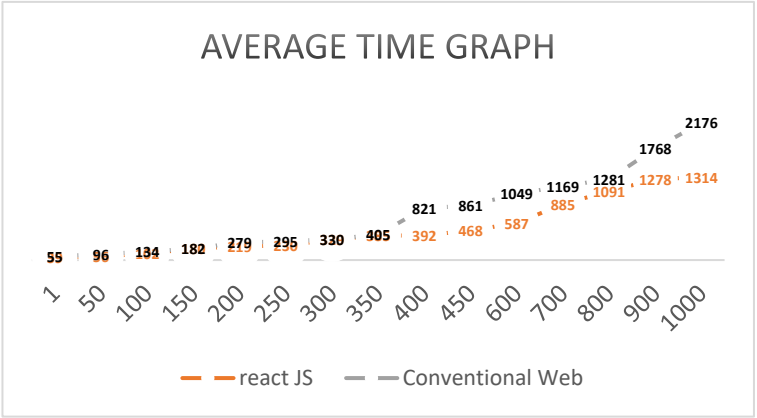

Fig. 4. Average Time Analysis

The time difference is due to the React JS has a virtual DOM feature and performs Node JS as a server-side programming language that has non-blocking input-output that easily serves multiple events simultaneously so developers can easily create more scalable servers that maximize the use of one CPU. Conventional web uses multi-threaded execution in the process of requesting to a server where one host is for one process thread, so this method will take a long time and will load the server compared to web React JS.

5 The Reliable Offline Web System for SMIs

Due to the lack of the capital and the information about the effectiveness of the industrial process, the productions in the SMIs fall under the expectations. In a recent study, the effectiveness of SMIs in the production process is raised by a reliable web-based application. The results produce an important impact on the economic development of the country.

The systems with web-based technologies indicated many advantages to develop the SMIs. However, the literature reviews and practices in SMIs indicated a lot of complications in implementation of web-based technologies. Commonly, the web-based applications can't work properly with narrow Internet bandwidth and low reliability of network connectivity. These conditions are frequently found in rural areas in developing countries. Furthermore, the online web-based applications have the potential for poor performance, i.e. due to a high number of requests to the same server simultaneously, slow network bandwidth, etc. Moreover, without denying the rising tendency in the web adoption, many SMIs chooses not to have a web-based system due to several barriers in the risks and the knowledge, ability, experience, and innovation of the SMIs owner and workers.

The reliable offline web system with the React JS and service worker concept is proposed in the current study. It might be applied to the web-based applications in SMIs. As an example of application, the prototype of a web-based system for SMIs is developed with the concept.

6 Conclusion and Future Works

Based on the results of the analysis, the web with React JS provide lower average access times compared to conventional web. With the total number of threads given 1000 on the conventional web, it shows a higher access time than the web of React JS. It can be concluded that the web using React JS can serve multiple requests simultaneously quickly.

The web system with React JS and service worker can be working offline on a client. By this capability, it can be said that this web-based system is more reliable. It may operate by a better performance in the condition of narrow Internet bandwidth and low reliability of network connectivity. Therefore, it can raise the effectiveness of labors, machines, and other units and support industrial processes. It will also encourage more and more SMIs in the rural areas of developing countries to utilize web-based applications.

It is expected to analyze the implementation of a completely web-based application with the countermeasure of the offline web restrictions, i.e. the analyses of the availability, the fault-tolerance, the reliability, etc. It is expected that much more studies are performed, i.e. the current technologies are implemented and analyzed, and the program function on the server is upgraded by applying the popular web-based technology, e.g. cloud computing, Web of Things, the impacts of the offline web-based technology in *SMIs* production floor, etc.

Acknowledgement

The authors would like to thank the Ministry of Research, Technology and Higher Education of Indonesia and Hasanuddin University for providing the facilities and financial support.

References

1. V. Kotelnikov, H. Kim, Soc. *Polity,(APCICT)*, (2007).
2. E. and enterprise census Ministry of Internal Affairs and communication, 10 a. Japan Policy on SME and Micro Enterprises, (2013).
3. A. W. Labib, J. Qual. Maint. Eng., (1998).
4. R. Rahayu, J. Day, Procedia-Social Behav. Sci., **195** (2015).
5. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Futur. *Gener*. *Comput*. *Syst*., **29**, 7 (2013).
6. M. A. Hussin, S. Shahbudin, N. M. Tahir, *IEEE Confer*. *Syst*. *Proce*. *Contr*. *(ICSPC)*, (2016).
7. S. A. A. Rizvi, S. Sunder, F. Haroon, A. Mirza, 10th Intern. *Confe*. *Frontiers Inform*. *Techn*., (2012).
8. N. V Gavrilov, IEEE Confer. Russian Young Researc. Elect. Electron. Eng.(EIConRus), (2017).
9. A. Scarpellini, L. Fasanotti, A. Piccinini, S. Ierace, F. Floreani, *IEEE 2nd Int*. *Forum Resear*. *Technol*. *Socie*. *Indust*. *Leveraging a better tomorrow (RTSI)*, (2016).
10. H. Fleischmann, J. Kohl, J. Franke, A. Reidt, M. Duchon, H. Krcmar, I*EEE 14th Int*. *Confer*. *Industr*. *Informat*.*(INDIN)*, (2016).
11. J. Yan Xin, T. Ramayah, P. Soto-Acosta, S. Popa, T. Ai Ping, Inf. Syst. Manag., **31**, 2 (2014).
12. P. Vyas, Int. J. Bus. Manag., **9,** 11 (2014).
13. J. Hu et al., 3rd Int. Confer. *Inform*. *Sci Contr*. *Eng*.(ICISCE), (2016).
14. Z. Tahir, *The Study on Automated HTML5 Offline Services to Overcome Low Reliability of Network Connectivity for Web-based Decision Support System Applications,* (2016).
15. I. Malavolta, G. Procaccianti, P. Noorland, P. Vukmirovic, IEEE/ACM 4th Int. *Confe*. *Mobile Software Eng*. *Systems (MOBILESoft*), (2017).
16. A. Fedosejev, *React*. *js essentials*. Packt Publishing Ltd, (2015).