Journal of Engineering and Applied Technology
Vol. 2, No. 1, March 2021, pp. 1-12

E-ISSN: 2716-2265
P-ISSN: 2716-2257

Development and performance analysis of the Gunungkidul cultural potential application based on progressive web apps

Pradana Setialana[1, *], Muhammad Nurwidya Ardiansyah[2], Nova Suparmanto[3]

[1, 2] Department of Electronics and Informatics Engineering Education, Faculty of Engineering, Universitas Negeri Yogyakarta, Kampus Karangmalang, Yogyakarta 55281, Indonesia
[3] Department of Industrial Engineering, Faculty of Engineering, Universitas Gadjah Mada, Jl. Grafika No. 2 Kampus UGM, Yogyakarta 55281, Yogyakarta, Indonesia
E-mail: pradana.setialana@uny.ac.id *
* Corresponding Author

ABSTRACT

Gunungkidul has various cultural potentials that make it a tourist destination. To make it easier for tourists to visit and get tourist destination information, several researchers developed a mobile application-based information system. However, mobile applications have several drawbacks as such as the user must install the application and can only be used on specific operating systems. The purpose of this research is to develop and analyze applications regarding the potential of Gunungkidul culture based on progressive web apps and which can be used without having to install applications and can run in all operating systems. The application development method uses Scrum and Ionic Framework as a framework for the application. The performance analysis method was tested on runtime performance (loading, scripting, rendering, painting, system) and memory usage (min JS Heap and max JS Heap) by using Chrome Developer Tools for 30 tests. The results of the development show that there are 7 features in the application, namely (1) Peta; (2) Geoheritage; (3) Daerah; (4) Cagar Budaya; (5) Kuliner; (6) Seni Adat Tradisi; (7) Agenda. Runtime performance and memory usage test results show the average value on aspects (1) Loading: 33.60 ms; (2) Scripting: 1069.20 ms; (3) Rendering: 84.90 ms; (4) Painting: 22.33; (5) System: 429.67 ms; (6) Min JS Heap: 8.05 MB; and (8) Max JS Heap: 19.51 MB.

ARTICLE INFO

Article history
Received:
19 March 2021
Revised:
1 April 2021
Accepted:
5 April 2021

Keywords
Progressive Web App
Performance Analysis
PWA Development
Ionic Framework

1. Introduction

Gunungkidul is one of the districts in the Yogyakarta Special Region Province. Gunungkidul has various cultural potentials as the destinations for local and foreign tourists. To provide information about the potential culture in Gunungkidul Regency, several researchers developed an information system that can be accessed easily, cheaply, quickly, and fully [1]. One system that provides information on all cultural potentials in Gunungkidul is Cakrawala Budaya Dhaksinarga. Cakrawala Budaya Dhaksinarga is a geographic information system about potential culture based on an Android mobile application.

Mobile-based information systems offer several advantages as such as the system can be easily accessed by the public anywhere and anytime with an Android smartphone device. Furthermore, a mobile-based geographic information system application also can be used as a real-time navigation tool [2]. Although it has various advantages, Android-based mobile applications also have several drawbacks. For example, the application can only run on the Android operating system and the user must install the application to access information in the application.

One way to overcome this shortcoming is to use Progressive Web Apps. Progressive Web Apps (PWA) is a combination of web technologies that can provide user experiences to use native applications [3, 4]. The combination of web technology in the PWA makes PWA-based applications run and accessible without having to perform an installation process on a mobile device so that the application can run on any operating system, not only Android [5].

The advantages of this PWA can be used to solve the problems of the Cakrawala Budaya Dhaksinarga application which is only based on Android. Therefore, the objectives of this study are (1) developing a PWA-based application of the PWA-based Cakrawala Budaya Dhaksinarga; (2) Analyze the performance of the PWA-based Cakrawala Budaya Dhaksinarga application.

2. Method

The development of the Cakrawala Budaya Dhaksinarga application uses the Agile method with the Scrum approach. Scrum is an agile development framework where this framework enables rapid and adaptive development to change [6]. At its core, Scrum divides development into iterations not longer than four weeks (called sprints). At the end of each sprint, a shippable product increment is delivered to the user. For each newsprint, a sprint-planning meeting is held, at which tasks for the sprint are selected by the developers themselves in collaboration with other stakeholders [7]. Meanwhile, the application performance analysis method was carried out 30 times by using the Chrome Developer Tools.

2.1. Application Development Methods

The first stage in development is the Product Backlog. At this stage, a discussion was held between the developer and the Cultural Office of Gunungkidul to determine the features that will be developed. The results of this discussion used the features of the application, namely

- *Peta* (Map)

  This feature is used to view information about the culture in Gunungkidul, especially the culture position in the form of maps. Navigation is also embedded in this feature to find the travel route from the user's point to the intended cultural place.

- Geoheritage

  This feature contains information about natural tourist attractions in Gunungkidul which can be a recommendation for nature recreation. In this feature, there is a description of each natural tourist spot. There are also maps and navigation that users can use to trace the path to these natural attractions.

- *Daerah* (Regional)

  This feature provides information about the cultural collections that exist in each sub-district in Gunungkidul Regency. Information about the culture is differentiated based on the sub-district to make it easier for users when searching for a culture in a certain sub-district in Gunungkidul.

- *Cagar Budaya* (Cultural Heritage)

  This feature contains information about the cultural heritage of Gunungkidul which has been classified based on certain categories, for example, prehistoric categories, Hindu-Buddhist categories, Islamic categories, and traditional house categories. Within this feature, information such as images, descriptions, and navigation are embedded and can be used to get the location of cultural heritage.

- *Kuliner Kerajinan* (Culinary and Crafts)

  This feature contains information about culinary and handicrafts in Gunungkidul. Inside this feature, the information such as images, descriptions, and navigation are embedded and can be used to see the location where the culinary or craft is located.

- *Seni Adat Tradisi* (Arts and Traditions)

  This feature contains information about traditional arts and customs in Gunungkidul. Inside this feature, there is information such as images, descriptions, and navigation that can be used to see the location where the art or traditional customs are held.

- *Agenda*

  This agenda feature contains information about traditional activities or events that will be held in Gunungkidul. In this feature, there is information such as the date it was held, a description of the event, and pictures or photos of the activity.

The next stage is Backlog Refinement. At this stage, an internal developer discussion is carried out to determine the system architecture. At this stage, a system architecture agreement is produced as shown in Fig. 1.

The system architecture design determines the framework that will be used in the application. Based on the results of the joint analysis, the framework that will be used for the architecture is the Ionic Framework. Ionic Framework is a framework that can be used to develop hybrid mobile apps including progressive web apps. The Ionic framework has several advantages as a framework for progressive web apps, among others, it is based on standard web components, provides various cross-platform UI components, and has good performance for use in mobile devices [8].

The last stage is running a sprint or iteration where at this stage, application development is carried out according to the discussion results at the Product Backlog and Backlog Refinement stages. The duration of the sprint in this development is one week.

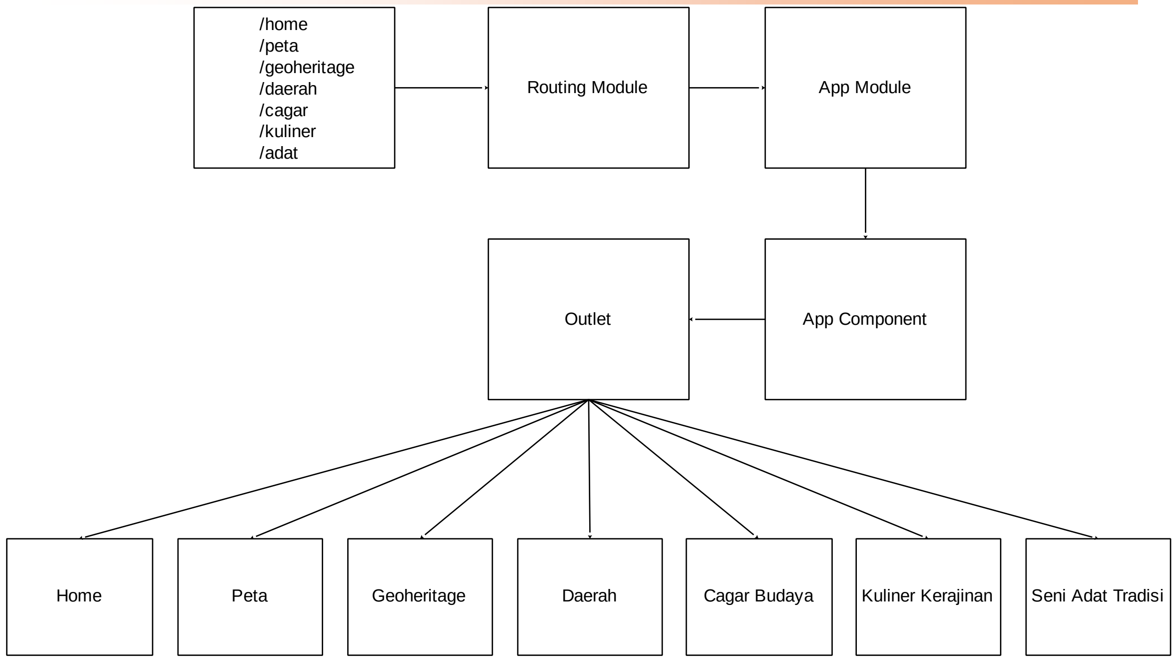


Fig. 1. Architecture of Cakrawala Budaya Dhaksinarga application

2.2. Performance Testing Methods

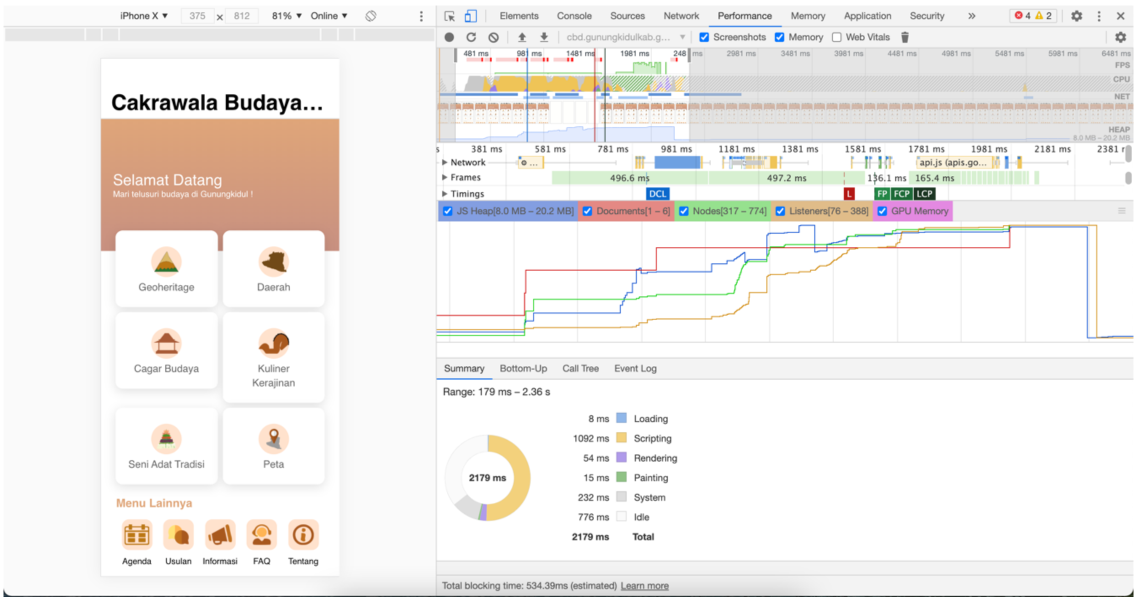
Performance analysis is carried out to test the performance of the system that has been developed. Testing the PWA-based Cakrawala Budaya Dhaksinarga application is carried out on two aspects, namely runtime and memory usage [9, 10]. Runtime testing is used to find out the time it takes to display all resources on a web when the user runs it. Meanwhile, memory usage testing is used to determine the amount of memory the device needs to run the web [11]. There are several aspects to runtime testing, namely

- Loading
- Scripting
- Rendering
- Painting
- System

While testing on memory usage is also carried out on several aspects, namely

- JS Heap
- Document
- Nodes
- Listener

To test all these aspects, Chrome Developer Tools is used [12]. Chrome Developer Tools is a tool provided on the Google Chrome web browser for inspecting, testing, and debugging a web or web-based application [11]. One of the features of Chrome Developer Tools is a performance test that can be used to test performance on a website which includes runtime testing (loading, scripting, rendering, painting, system, idle) and memory testing (JS Heap, documents, nodes, listeners). Fig. 2 is an example of using the Chrome Developer Tools performance test feature to test the PWA-based Cakrawala Budaya Dhaksinarga application.



```
Range: 179 ms – 2.36 s

   8 ms  Loading
1092 ms  Scripting
  54 ms  Rendering
  15 ms  Painting
 232 ms  System
 776 ms  Idle
2179 ms  Total

Total blocking time: 534.39ms (estimated) Learn more
```

Fig. 2. Analyze performance with chrome developer tools

The performance testing of the PWA-based Cakrawala Budaya Dhaksinarga application using Chrome Developer Tools was carried out 30 times. The test is carried out using a laptop device with the following specifications

- Model : MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports)
- Processor : 2,3 GHz Dual-Core Intel Core i5
- Memory : 8 GB 2133 MHz LPDDR3
- Graphics : Intel Iris Plus Graphics 640 1536 MB

Every experiment with Chrome Developer Tools recorded loading time, scripting, painting, system, idle, JS Heap, document, nodes, and listener. Configure the Chrome Developer tools using the following specifications

- Emulated Device : iPhone X (375 x 812)
- Network : Online
- CPU : No throttling

3. Results and Discussion

3.1. Application Development Results

The result of developing an application with the Scrum method is that the application is designed according to the features that have been set at the beginning. On the start page, several menu cards are provided which are features that can be accessed by the user. Another page in the application is a map page that contains information in the form of a location and a detail page that contains information that reviews the cultural heritage in more detail. Fig. 3, Fig. 4, and Fig. 5 is a page view of the application that has been made.

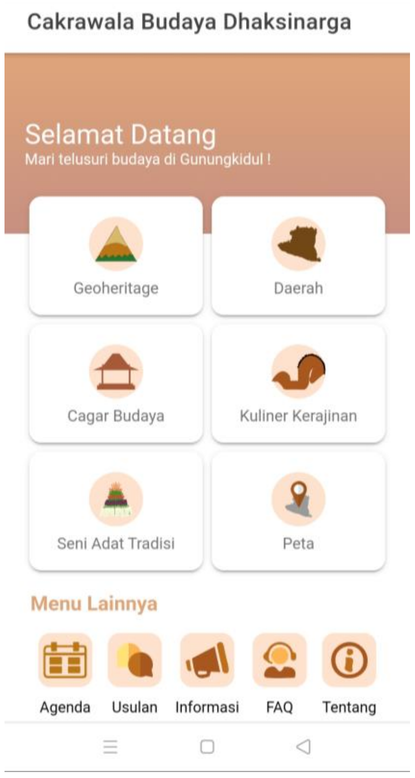


Fig. 3. Home page

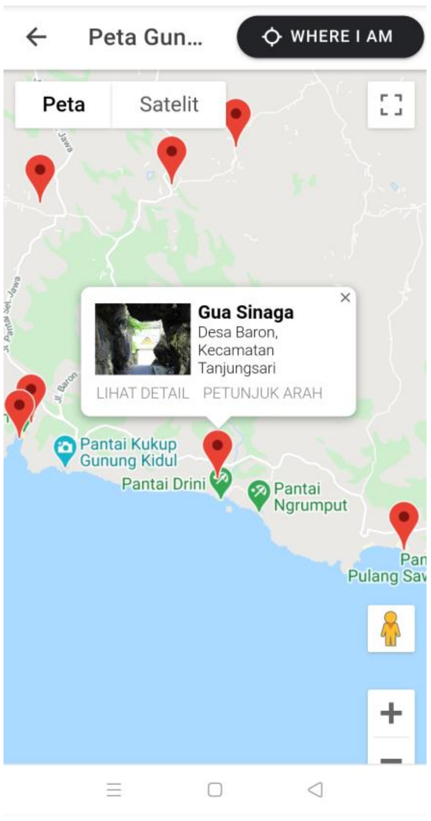
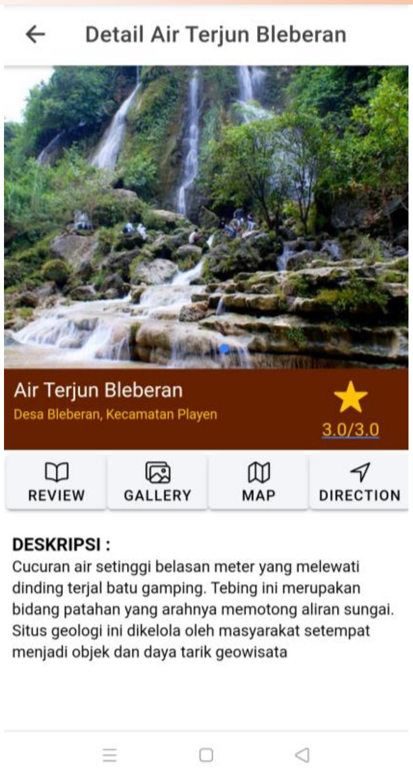


Fig. 4. Cultural potential map page



DESKRIPSI :
Cucuran air setinggi belasan meter yang melewati dinding terjal batu gamping. Tebing ini merupakan bidang patahan yang arahnya memotong aliran sungai. Situs geologi ini dikelola oleh masyarakat setempat menjadi objek dan daya tarik geowisata

Fig. 5. Potential details page

3.2. Performance Testing Results

The results of testing the performance of the Cakrawala Budaya Dhaksinarga application using Chrome Developer Tools 30 times can be seen in Table 1. In Table 1, there are two types of performance testing, namely runtime testing, and memory usage. There are 5 aspects in runtime testing, all of which use millisecond units. Whereas in memory usage there are 5 aspects where only the minimum JS Heap and maximum JS Heap measure memory usage in megabytes. The loading performance aspect of the 30 trial runtime tests can be seen in Fig. 6. Fig. 6 shows that the loading performance is in the range of 8 ms - 80 ms with an average of 33.60 ms

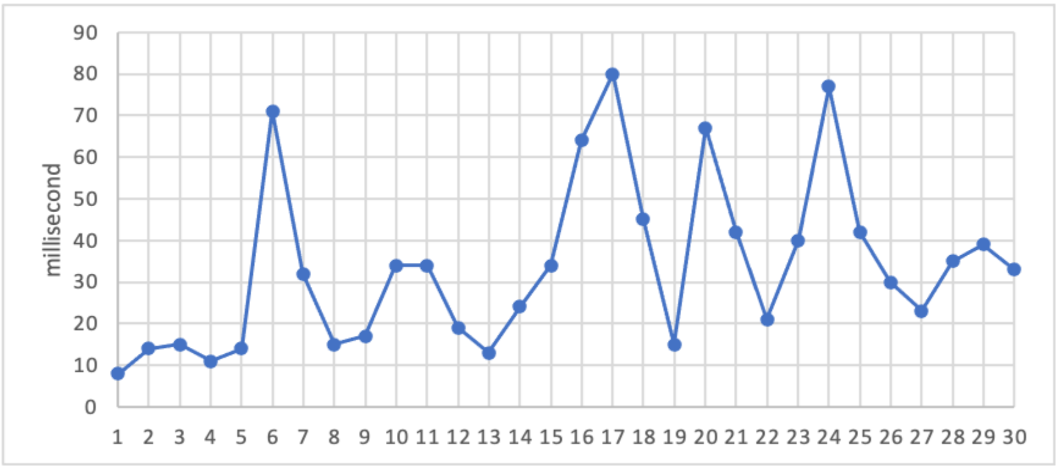


Fig. 6. Loading performance

Table 1. Performance testing results

Test	Runtime[a]						Memory Usage				
	Loading	*Scripting*	*Rendering*	*Painting*	*System*	*Total*	*Min JS Heap[b]*	*Max JS Heap[b]*	*Document*	*Nodes*	*Listener*
1	8	1126	41	18	775	1968	8,1	19,2	1 - 5	317 - 767	78 - 428
2	14	1028	46	76	356	1520	8,1	19,3	1 - 5	317 - 767	76 - 772
3	15	979	87	18	541	1640	8,1	19,4	1 - 5	317 - 767	78 - 548
4	11	1002	74	12	415	1514	8,1	19,2	1 - 5	317 - 767	76 - 492
5	14	1102	100	13	382	1611	8,1	19,4	1 - 5	317 - 767	78 - 554
6	71	906	61	14	571	1623	8	19,8	1 - 5	317 - 767	76 - 502
7	32	938	57	11	418	1456	8	19,5	1 - 5	317 - 767	76 - 494
8	15	912	51	29	286	1293	8,1	19,4	1 - 5	317 - 767	76 - 448
9	17	913	57	21	373	1381	8	19,4	1 - 5	317 - 767	76 - 538
10	34	1155	213	26	400	1828	8,1	19,4	1 - 5	317 - 767	78 - 518
11	34	1150	53	6	392	1635	8,1	19,3	1 - 5	317 - 767	78 - 506
12	19	1691	100	11	474	2295	8,1	19,4	1 - 5	317 - 767	78 - 556
13	13	1099	73	10	336	1531	8,1	19,7	1 - 5	317 - 767	78 - 462
14	24	1004	199	27	328	1582	8	19,8	1 - 5	317 - 767	76 - 476
15	34	1007	94	17	618	1770	8	19,3	1 - 5	317 - 767	76 - 478
16	64	925	95	17	574	1675	8,1	19,7	1 - 5	317 - 767	76 - 466
17	80	1002	72	89	516	1759	8	19,5	1 - 5	317 - 767	76 - 496
18	45	1171	75	22	433	1746	7,7	18,9	1 - 5	317 - 767	75 - 465
19	15	1034	69	31	328	1477	8,1	19,8	1 - 5	317 - 767	76 - 458
20	67	1163	152	35	389	1806	8,1	19,7	1 - 5	317 - 767	78 - 470
21	42	995	168	16	537	1758	8,1	19,7	1 - 5	317 - 767	78 - 458
22	21	1069	79	7	386	1562	8	19,4	1 - 5	317 - 767	76 - 504
23	40	1077	52	13	386	1568	8,1	19,8	1 - 5	317 - 767	76 - 496
24	77	1082	63	57	411	1690	8,1	19,8	1 - 5	317 - 767	78 - 482
25	42	1073	50	12	422	1599	8,2	19,3	1 - 5	317 - 767	76 - 576
26	30	1133	72	11	379	1625	8,1	19,8	1 - 5	317 - 767	78 - 460
27	23	935	99	10	308	1375	8	19,4	1 - 5	317 - 767	76 - 432
28	35	1082	63	9	274	1463	7,7	19	1 - 5	317 - 767	74 - 455
29	39	1113	55	14	408	1629	8,1	19,8	1 - 5	317 - 767	78 - 480
30	33	1210	77	18	474	1812	8,1	20,1	1 - 5	317 - 767	80 - 626
Avg	33.60	1069.20	84.90	22.33	429.67	1639.7	8,05	19,51			

[a.] Millisecond (ms); [b.] Megabyte (MB)

Scripting performance after 30 experiments showed that the time needed to run scripts in the application range from 906 ms - 1691 ms with an average value of 1069.20 ms. Fig. 7 shows a graph of scripting performance on 30 trials.

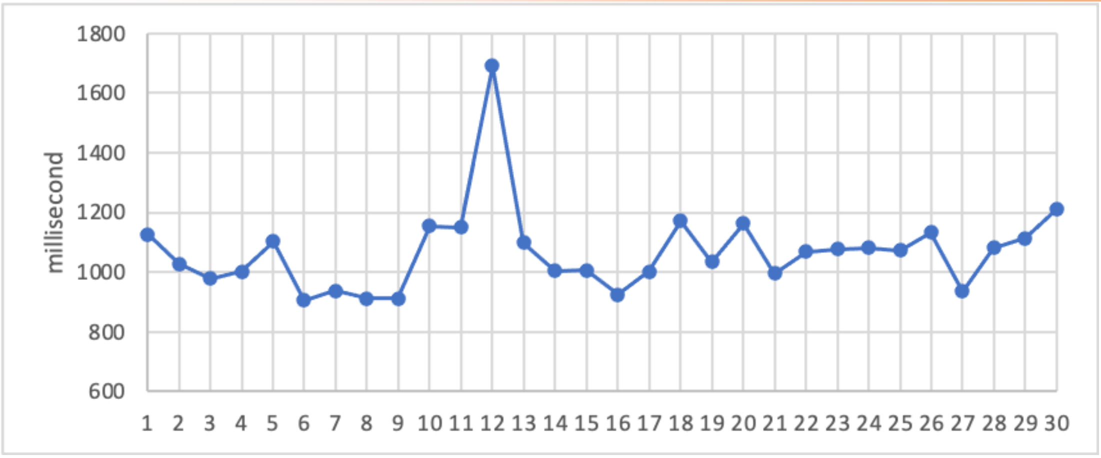


Fig. 7. Scripting performance

Fig. 8 is a graph of rendering performance in runtime testing performed 30 times. In Fig. 8. it can be seen that the time needed for the application to rendering is around 41 ms - 213 ms with an average of 84.90 ms.

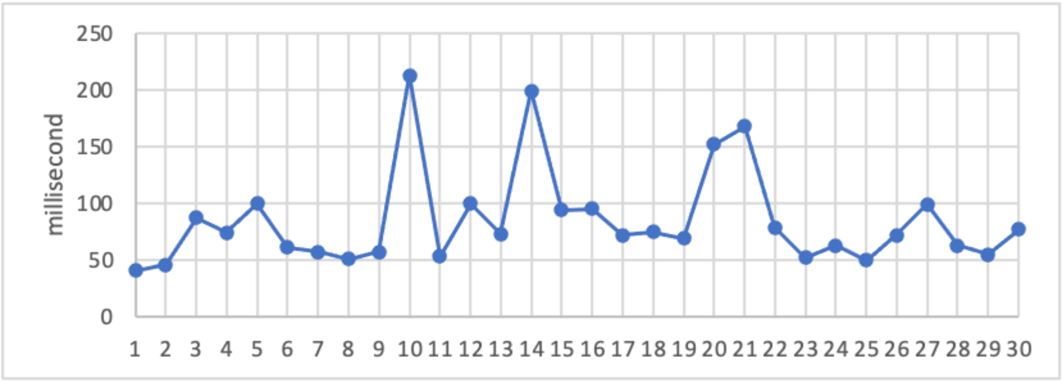


Fig. 8. Rendering performance

Painting performance on runtime testing can be seen in Fig. 9. Fig. 9 shows the speed needed to convert a vector into pixels in an experiment conducted 30 times ranging from 6 ms - 89 ms with an average of 22.33 ms.

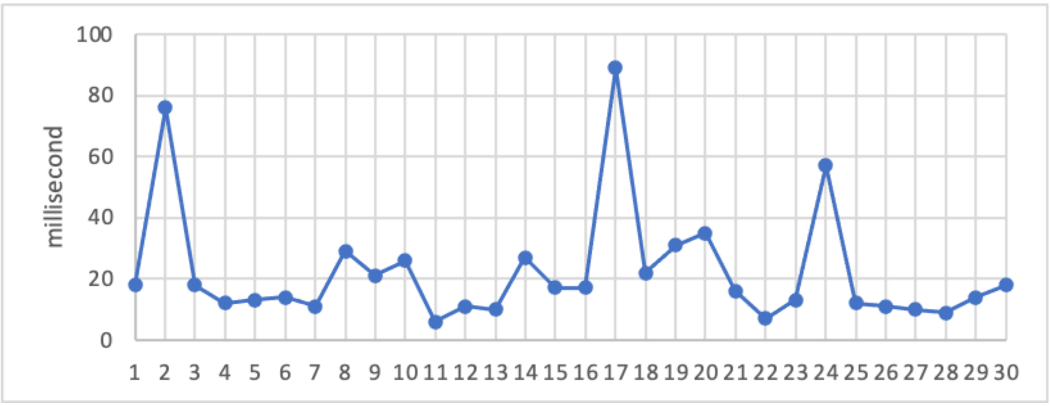


Fig. 9. Painting performance

The results of runtime testing on the system aspect can be seen in Fig. 10. In Fig. It can be seen that the time it takes for the system (browser) to run this application ranges from 274 ms - 775 ms with an average of 429.67 ms.

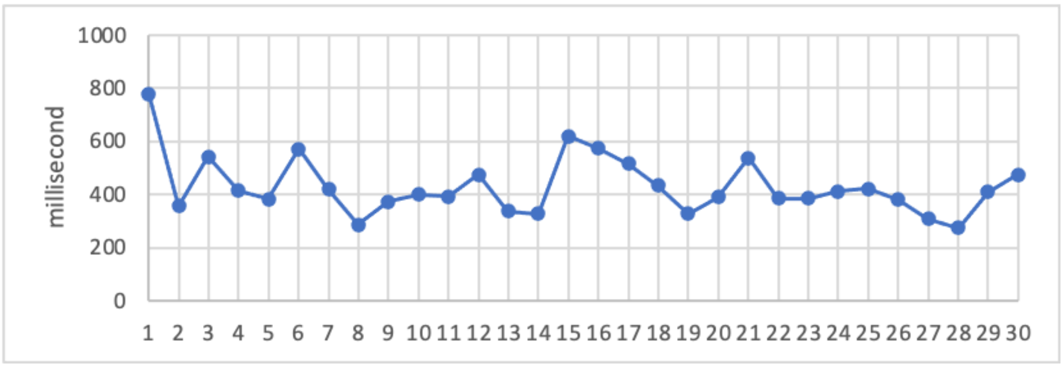


Fig. 10. System performance

In memory testing, there are two main aspects, namely minimum JS Heap and maximum JS Heap. Fig. 11 is the minimum JS Heap graph on tests carried out 30 times. From the test results, it is known that the minimum use of JS Heap memory is between 7.7 MB - 8.2 MB with an average of 8.05 MB.

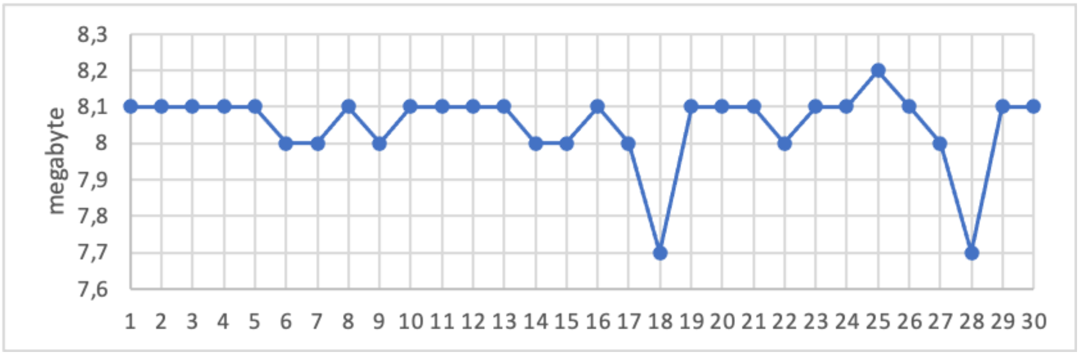


Fig. 11. Minimum JS heap performance

Fig. 12 shows the maximum JS Heap performance in the memory usage test that was conducted 30 times. The test results show that the maximum memory usage ranges from 18.9 MB - 20.1 MB with an average of 19.51 MB

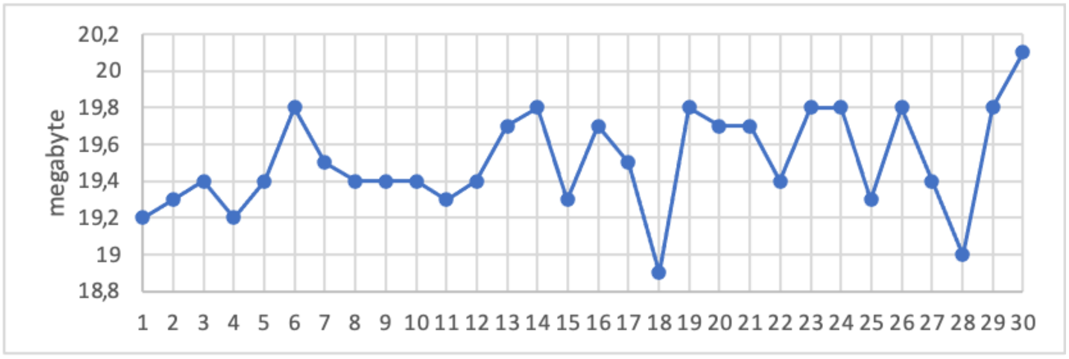


Fig. 12. Maximum JS heap performance

4. Conclusion

The development of the Cakrawala Budaya Dhaksinarga application based on Progressive Web Apps was carried out by using the Scrum method. This method is used because it is a method of developing software that is fast and adaptive to change. Ionic Framework is used as a progressive web apps framework for these applications. There are 7 main features in the Cakrawala Budaya Dhaksinarga application, namely (1) Peta (Map): information on the location of cultural potentials; (2) Geoheritage: information on the potential for nature-based culture; (3) Daerah (Region): information on the cultural potential of each sub-district in Gunungkidul; (4) Cagar Budaya (Cultural Heritage): information on cultural heritage by category; (5) Kuliner Kerajinan (Culinary and Crafts): information on the potential of culinary and handicrafts; (6) Seni Adat Tradisi (Arts and Traditions): information on the potential of indigenous and traditional arts; (7) Agenda: information on customary activities or events.

The Cakrawala Budaya Dhaksinarga application that has been developed is then tested for its performance which includes runtime testing and memory usage testing. This test was carried out 30 times by using the Chrome Developer Tools. The results of the overall runtime test show that the average time required for loading is 33.60 ms, scripting is 1069.20 ms, rendering is 84.90 ms, painting is 22.33 ms, and the system is 429.67 ms. Meanwhile, the memory usage test in JS Heap shows that the average memory required is at least 8.05 MB and a maximum of 19.51 MB.

References

[1] R. Agrarian, A. Suprayogi, and B. Yuwono, “Pembuatan Aplikasi Mobile Gis Berbasis Android Untuk Informasi Pariwisata Di Kabupaten Gunungkidul,” *J. Geod. Undip*, vol. 4, no. 2, pp. 241–247, 2015.

[2] E. Fernando, M. Irsan, D. F. Murad, Surjandy, and Djamaludin, “Mobile-based geographic information system for culinary tour mapping in Indonesia,” *2019 Int. Conf. Inf. Commun. Technol. ICOIACT 2019*, pp. 28–31, 2019, doi: 10.1109/ICOIACT46704.2019.8938511.

[3] M. Hajian, *Progressive Web Apps with Angular*. Oslo, Norway: Apress Media, LLC, 2019.

[4] B. Kumar NJ, “Progressive Web Apps a New Way to Experience Mobile,” *Int. J. Eng. Res. Technol.*, vol. 4, no. 22, pp. 1–2, 2016.

[5] D. Sheppard, *Beginning Progressive Web App Development*. Tinley Park, Illinois: Apress Media, LLC, 2020.

[6] D. McKenna, *The Art of Scrum: How Scrum Masters Bind Dev Teams and Unleash Agility*. Aliquippa, Pennsylvania: CA Press, 2016.

[7] M. Hron and N. Obwegeser, “Scrum in Practice: an Overview of Scrum Adaptations,” *Proc. 51st Hawaii Int. Conf. Syst. Sci.*, pp. 5445–5454, 2018, doi: 10.24251/hicss.2018.679.

[8] F. Cheng, *Build Mobile Apps with Ionic 4 and Firebase*, Second Edi. Sandringham, Auckland: Apress Media, LLC is, 2018.

[9] K. Zhu, J. Fu, and Y. Li, "Research the performance testing and performance improvement strategy in a web application," *ICETC 2010 - 2010 2nd Int. Conf. Educ. Technol. Comput.*, vol. 2, pp. 328–332, 2010, doi: 10.1109/ICETC.2010.5529374.

[10] K. Behl and G. Raj, “Architectural Pattern of Progressive Web and Background Synchronization,” *Proc. 2018 Int. Conf. Adv. Comput. Commun. Eng. ICACCE 2018*, no.

June, pp. 366–371, 2018, doi: 10.1109/ICACCE.2018.8441701.

[11] D. Odell, *Pro JavaScript Development: Coding, Capabilities, and Tooling*. New York: Apress Media, LLC, 2014.

[12] K. Basques, “Get Started With Analyzing Runtime Performance,” *Chrome DevTools Guides*, 2020. https://developers.google.com/web/tools/chrome-devtools/evaluate-performance (accessed Mar. 16, 2021).