Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 7, No. 2, August 2021, pp. 296-305
ISSN: 2338-3070, DOI: 10.26555/jiteki.v7i2.21169

Analysis and Implementation of Microservice Architecture Related to Patient Drug Schedule Based on FHIR Standard

Ariq Musyaffa Ramadhani, Andrian Rakhmatsyah, Rahmat Yasirandi
School of Computing Telkom University, Bandung, Indonesia

ARTICLE INFO

Article history:

Received July 20, 2021
Revised August 07, 2021
Accepted September 01, 2021

Keywords:

Centralized System;
Platform;
Medical Adherence;
FHIR

ABSTRACT

In several previous studies, smart devices have been developed to help improve a patient's medication adherence but have problems, namely data management that is not centralized and not integrated, so that mitigation is quite vulnerable. In this study, a platform was built that can manage data centrally and apply the FHIR (Fast Healthcare Interoperability Resources) health data standard. The main components used to implement the FHIR standard are resources and REST APIs. The resource is a data model that defines the structure and data elements that are exchanged. This data exchange is carried out on top of the REST API using the HTTP protocol. Platform testing uses positive/negative testing and stress testing methods to be able to see the performance of the platform. The test results show that the platform prototype can provide a response that is in accordance with the request given and has a very tolerant error value of 0% with a latency value of 3 to 22 seconds with a total of 100 to 130 users.

Ariq Musyaffa Ramadhani,
School of Computing, Telkom University, Bandung, Indonesia
Email: ariqramadhan@student.telkomuniversity.ac.id

1. INTRODUCTION

Healing of diseases such as cancer and hypertension does not only depend on skilled medical personnel but also with adherence to drug therapy but the level of awareness to consume drugs regularly is still low [1]. WHO has reported that patients who are in the long-term therapy have adherence rates of only 50% in developed countries, even lower rates in developing countries [2]. Research in [3] showed that CML patients required adherence rates of more than 90% to be able to improve treatment in cases of Imatinib use. According to a survey conducted in research [4] From 2,546 questionnaires spread across 63 countries, it shows that there are 32.7% of people whose level of adherence to taking their medication is high, 46.5% is moderate, and 20.7% is low in cases of Chronic Myelogenous Leukemia (CML). As stated in research [5], poor medication adherence can lead to a variety of conditions, namely, significant deterioration of the disease, treatment failure, and increased health care costs. In comparison, study in [6] says that low rates of medication adherence in the United States lead to an annual expenditure of $290 billion.

According to research in [2], medication adherence problems can be overcome by utilizing technological developments, especially on smartphones, considering that smartphone users are estimated to reach 2 billion users. However, relying solely on reminders is not enough to increase drug adherence rates. Research [7] said that the use of reminders is not enough. Other interventions are needed to increase the level of medication adherence. According to research [3] and [1], direct monitoring from the pharmacist or doctor greatly affects the increase in the rate of adherence to taking medication.

In research [8], Smart Medicine Box has been implemented, but both device configuration data and data from using the device are stored directly on the smartphone. Similar to research [8], research [9] developed a smart pill dispenser that can be connected to a smartphone as a reminder, but in this study, compliance data is only stored on smartphones. This can result in data mitigation being quite vulnerable, considering that the data stored is only on the smartphone. In research [10] has developed an IoT-based smart medicine dispenser. The developed smart medicine dispenser is connected to the internet network to be able to store medication

adherence data in the database. However, in this study, there was no monitoring feature available that could be used by medical parties.

Therefore, a platform that can accommodate patient medication compliance data and has a monitoring feature to be used by medical parties is needed. The platform must also have standards to improve system interoperability so that it has a wider utilization [11]. The research contribution is to analyze and designs a platform prototype that has standards to manage adherence data centrally and can manage mass use, and also can provide data services to interested stakeholders authorized.

2. RESEARCH METHOD

There are two types of user roles in the system built, namely doctor users and patient users (Fig. 1). Users can only view statistics on drug compliance data and laboratory report data from their patients in the doctor's role. In the role of the patient, the user becomes a source of input for data on drug compliance and laboratory results report data. Laboratory result report data is obtained from patient-user input manually through an Android-based application. Data on drug compliance is obtained through user interaction with an external entity that can monitor the user's medication's timeliness. External entities here can be in the form of Smart Medicine Box, Smart Pill Dispenser, and other similar devices. Research [8] used Bluetooth Low Energy (BLE) as a connection medium between the Smart Medicine Box and the user's smartphone. In this study, the external entity was simulated using an Android-based application installed on a smartphone. This application simulates triggers obtained from user interactions with external entities when taking drugs. The trigger is then sent to another smartphone with the main application installed via a Bluetooth connection, and then the adherence data is forwarded to the platform.

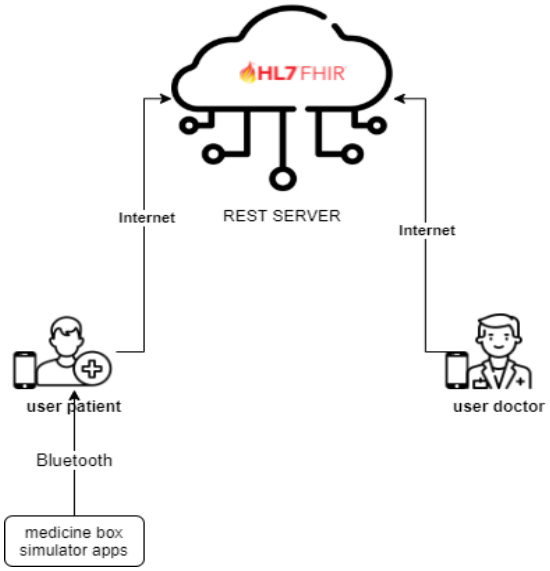


Fig. 1. Platform Design in general

2.1. Platform Prototype Design

The prototype platform uses the FHIR (Fast Healthcare Interoperability Resources) standard. The use of the FHIR standard aims to improve platform interoperability so that it can be interconnected with platforms that apply the same standard so that it has the potential to reach a broader range of users [12]. FHIR was chosen because it has been recognized by several large companies such as Amazon, IBM, Google, and Apple as the standard for exchanging health data [13]. The role of a platform is to provide the services required by the client. These services result from data processing, so data is the main thing in the services provided by the platform. As stated in the System Design section, the platform prototype provides services for doctors to monitor their patients, so that this platform prototype requires patient and doctor data models. Doctor users require data on drug compliance, and laboratory test results data are also needed as supporting data. The fields used in the patient and doctor data model can be seen in Table 1. The fields used in each data model are Table 1 [14].

Fields used in the schedule/adherence data model taking medication and the laboratory test results are based on an application called CML Today, which is available in the Play Store for Android Smartphones. Mapping models data into the FHIR Resource form is carried out to be able to apply the FHIR standard to the prototype platform built. Platform prototype using FHIR (Fast Healthcare Interoperability Resources). Four

FHIR Resource models are used, namely, Resource Practitioner to represent doctor user data, Resource Patient to represent patient-user data, Resource DiagnosticReport to represent data on patient laboratory test results, and MedicationStatement to represent drug schedule data. Platforms using the JSON data format. The JSON format is used because it has a level of popularity and effectiveness is quite high [15] and because of its simple structure [16].

Table 1. Fields used in each data model

Data Model	Field
Patients	a) Name b) Gender c) Photo d) Contact e) Doctor in charge f) Schedule/Adherence data g) Laboratory result data
Doctor	a) Name b) Gender c) Photo d) Contact e) Patients
Medication schedule/adherence	a) Data owner(patient) b) Schedule c) Time drug taken d) Medication name e) Dosage f) Note
Laboratory Results	a) Data owner(patient) b) Laboratory results validity period c) BCR_ABL value d) White blood cell value e) Hemoglobin value f) Trombosit value g) Hematocrit value h) Eritrosit value i) Note

2.2. REST Server Design

To be able to apply the FHIR standard also requires a REST Server that uses the HTTP (Hypertext Transfer Protocol) protocol as according to research [17] and [12]. REST Server is built using NodeJS and with the help of ExpressJS as a back-end framework. NodeJS is a tool that serves to make the JavaScript language run on the server-side. According to research [18], NodeJS and ExpressJS are an ideal combination for REST Server development because they can handle multiple requests simultaneously. Platform prototype using MongoDB as DBMS (Database Management System) to accommodate FHIR Resources. MongoDB is a NoSQL DBMS. NoSQL has advantages over other DBMSs, including high performance and high scalability [19].

Rest Server is built using the ExpressJS framework starting with the implementation of the data model (Fig. 2). The implementation of the data model is assisted by the Mongoose library. Mongoose is used to create data objects and to connect the REST Server to the database. Then proceed with making a Controller for each data model. The controller function is to handle the CRUD (Create, Read, Update, Delete) process on each data. Each type of controller can use the functions of other controllers if needed. Finally, create routes. Routes are used to create endpoint URIs. The endpoint itself is a URI (Uniform Resource Identifier) which is used to access the resource(data) [20].

2.3. Android Application Design

Research [21] developed an android application to track the Trans Semarang BRT, the application that was built also has two roles, but the researcher separates the two roles into two different applications. Research [22] has developed a platform for health monitoring named Mooble. The platform developed in this study has a patient role and a doctor/health staff role, but each role is also separated into two different applications. Even the application for the doctor/health staff role is only available in web form.

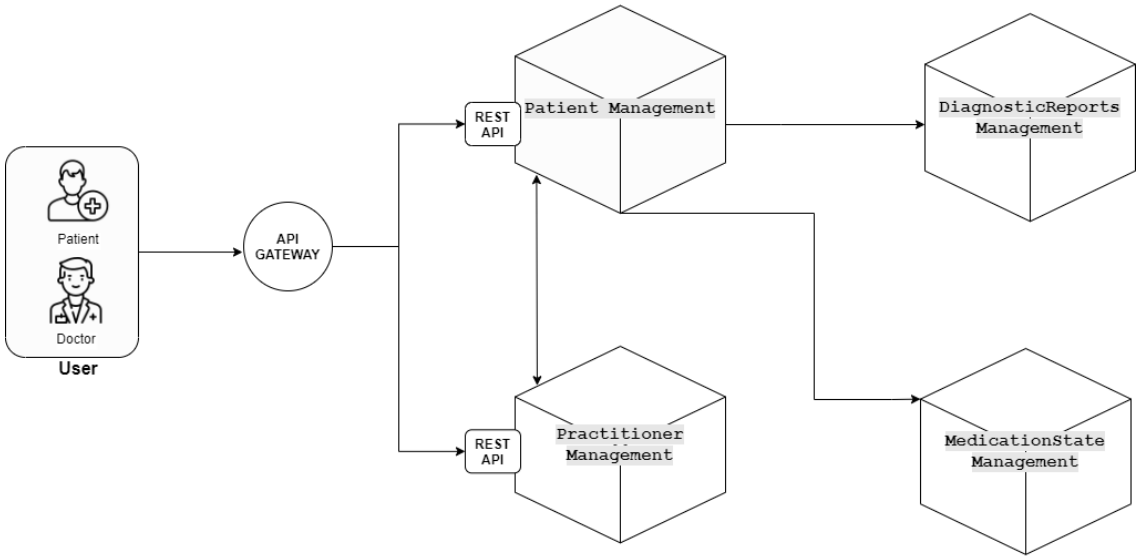


Fig. 2. REST Server design

Unlike the research of [21] and [22], in this study, both roles remain in the same Android-based application. There are two types of user roles that have their respective functions, namely patient and doctor roles. Users with patient roles are useful as medication adherence data entry, drug scheduling, and lab test result report data entry. Users with patient roles can also view statistics from medication adherence and lab test result reports. Meanwhile, users with a doctor role have features for monitoring medication adherence data and lab test result reports from their patients. Users with the doctor role cannot make changes or delete patient data.

The application is built using the Android Studio IDE and using the Kotlin language. Android Studio IDE is Google's official Integrated Development Environment and is specifically designed for Android application development [23], while Kotlin is a modern programming language that can run on the Java Virtual Machine and has interoperability with the Java programming language and other programming languages [24][25]. Kotlin is used because it is the recommended language by Google for Android application development [24].

The applications must be connected to the internet network to be able to connect and make requests to the prototype platform. To be able to make requests to the platform prototype, the application uses the URI endpoint that is already available on the platform prototype.

3. RESULTS AND DISCUSSION

In [8], [10], and [9] research, a platform that can manage compliance data has not been developed and does not yet have a monitoring feature for medical parties/doctors to be able to monitor patient compliance. Therefore, the testing in this study is quite different from previous studies.

Tests are carried out on the platform prototype to see whether the platform prototype can provide services according to user needs and tolerant response time. Positive/negative testing is used to check whether the response obtained is following what the platform should give if the request given is correct and provides a warning response if the request given is not appropriate. Stress testing is done to test whether the prototype platform can handle many requests within a particular time. The following are the results of testing with positive/negative methods of testing and stress testing.

3.1. Posittive/Negative Testing

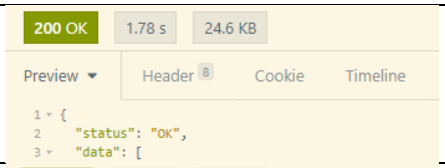
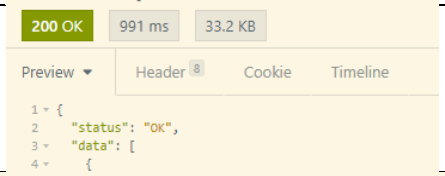
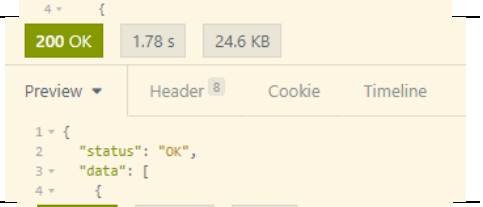
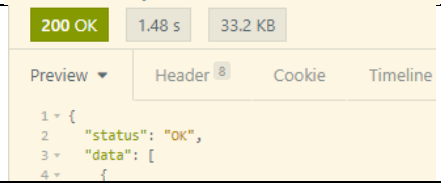
3.1.1.Posittive Test

A positive test is carried out on the platform prototype to see whether the response given is as needed or not if the request issued by the client is valid (Table 2). A positive test is done by requesting the platform prototype. Positive tests will be carried out on five different endpoints where two endpoints belong to doctor users, and the other three belong to patient users. In this test, the request is said to be valid if the parameter contains the registered id.

Each id used as a parameter is a registered id, and the response given is detailed data that matches the id provided through the client request. The correct response is also marked with a response message code 200,

which means success. In this test, it can be said that the platform prototype can respond according to the client's needs if the request given is valid.

Table 2. The testing result with a positive test method

No	Endpoint	Feature	Response Message	Response
1	/patients/:patientId	Get one patient data using patient id	200	
2	/patients/reports/:patientId	Get all laboratory data using patient id	200	
3	/patients/medicationState /:patientId	Get all medication adherence data using patient id	200	
4	/doctors/:doctorId	Get one doctor data using doctor id	200	
5	/doctors/patient/reports /:patientId	Get all laboratory data using patient id	200	

3.1.2.Negative Test

Negative tests are carried out on the platform prototype to see whether the response given is as needed or not if the request issued by the client is invalid (Table 3). A negative test is done by requesting the platform prototype with invalid parameters. Negative tests will be carried out on five different endpoints where two endpoints belong to doctor users, and the other three belong to patient users. In this test, the request is said to be invalid if the parameter contains an unregistered id.

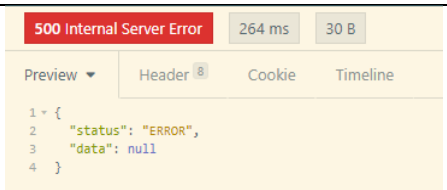
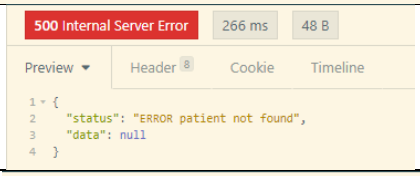
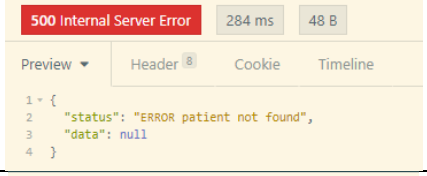
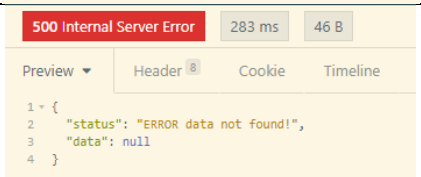
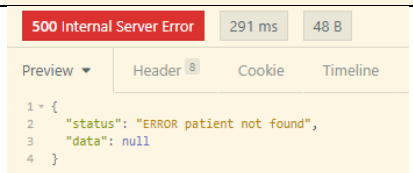
Each id used as a parameter is an unregistered id, and the response given is a warning message and is marked with a response message code 500, which means an error has occurred. In this test, it can be said that the platform prototype can provide a response warning message to the client if the request given is invalid.

3.2. Stress Testing

Stress testing is a test carried out to determine the limit of request and response numbers that the system can handle [26]. Stress testing is done to test the performance of the platform prototype when handling many requests at a particular time and checks the upper limits of the requests can the system can handle. Stress testing is done with the help of Apache JMeter software. Apache JMeter is open-source Java-based software designed as a tool to perform performance tests [27]. JMeter was chosen because research by [28] shows that JMeter is a tool that is widely used and has proven to have better performance compared to other tools such as Siege, LoadRunner, and Microsoft Visual Studio (TFS).

Tests are carried out on two endpoints with two different methods. Tests were carried out at two endpoints with two different methods. The number of requests starts at 100 requests in one second and will continue to increase until one of the endpoints reaches an error value of 100%

Table 3. The testing result with negative test method

No.	Endpoint	Feature	Response Message	Response
1	/patients/:patientId	Get one patient data using patient id	500	
2	/patients/reports/:patientId	Get all laboratory data using patient id	500	
3	/patients/medicationState/:patientId	Get all medication adherence data using patient id	500	
4	/doctors/:doctorId	Get one doctor data using doctor id	500	
5	/doctors/patient/reports/:patientId	Get all laboratory data using patient id	500	

The test environment on the server-side can be seen in Table 4. The server used is a virtual container provided by Heroku in which the platform prototype program has been installed while the client uses a physical computer that has the specifications as shown in Table 5. The client accesses the server via the internet. Heroku is a PaaS (Platform as a Service) based on containers. These containers are called "dynos," which contain applications and all necessary dependencies and run on a shared host [29].

Table 4. Server specification

Server Specification	
Processor	Dual Core
Memory	512MB

Table 5. Client specification

Client Specification	
Processor	Intel Core i5-4210U Quad Core 1.7Ghz
Memory	8GB
Internet Bandwidth	5 Mbps
Internet Speed	4 Mbps Upload/Download

The test produces three values, namely the average latency (Avg. Latency), error, and the average total bytes of data received from the platform prototype (Avg. Bytes). Latency is the total time from a request sent until a response is given [30], while the error indicates the percentage of the number of requests that the platform prototype failed to process. The results of testing with the stress testing method can be seen in Table 6.

Table 6. Stress testing result

No.	Endpoint	Method	Client	Avg. Latency(ms)	Error%	Avg. Bytes
1	/patients	GET	100	15065.90	0.00%	424440
2			130	22855.68	0.00%	369310
3			150	23672.13	8.67%	465400
4			300	28299.85	33.33%	383110
5			500	31451.33	85.60%	325680
6			1000	24070.64	78.90%	380130
7	/patients/report	POST	100	3555.18	0.00%	24810
8			130	8448.15	0.00%	13610
9			150	12387.08	0.00%	11080
10			300	20453.81	94.33%	32390
11			500	12309.52	99.80%	25290
12			1000	16552.89	100.00%	82810

3.1. Test Analysis

The test results with the positive/negative testing method show that the platform prototype can respond according to the conditions of the request given. All test results using the positive test method produce a response containing data in JSON and accompanied by a status code 200 as shown in Table 2, while all test results using the negative test method provide a warning message accompanied by a status code 500 as shown in Table 3. This indicates that no warning message appears during testing with a positive test and no data provided by the prototype platform when testing with a negative test.

Testing using stress testing (Fig. 3 and Fig. 4) on the number of clients 100 and 130 produces a good error value of 0%, which means that all requests provided have been successfully processed by the platform prototype even though the latency value is intolerant, especially at endpoints with the GET method, which reaches 15 to 22.8 seconds. In testing with 150 clients, the endpoint with the GET method showed an increase in the error value but was still classified as tolerant. In testing with 150 clients, both endpoints experienced an increase in latency values, but for endpoints with the POST method, they were still relatively tolerant, unlike endpoints with the GET method, which had reached 23 seconds. Testing with 300 clients gave a significant change in the error value, especially at the endpoint with the POST method, as well as the latency value, which was classified as intolerant at both endpoints. Then in testing with 500 clients, there was an increase in the error value, which was quite large at the endpoint with the GET method, while at the endpoint with the POST method, the change in the error value was not too large but was getting closer to 100%. In testing with 500 clients, the latency value at the endpoint with the GET method has increased, while the endpoint with the POST method has decreased by 8 seconds. In testing with 1000 clients, the endpoint with the POST method shows an error value of 100%, which means that all requests failed to be processed by the platform prototype. At the endpoint with the GET method, there is a slight decrease in the error and latency values, but both values are still classified as intolerant values.

Most of the errors were caused by two things. Namely, the request processing time that exceeded the Response timeout limit, which in this study was set for 20 seconds, and the platform prototype crashed during request processing which was marked with an error code 503 as seen in Fig. 5. Crashes occur due to insufficient memory capacity and processor capability on the prototype platform to handle many requests—crash message as seen in Fig. 6. From the testing results using the stress testing method, it can be concluded that the prototype platform can only take up to 130 requests in one second.

4. CONCLUSION

After testing with positive/negative testing and stress testing methods, the results show that the prototype platform can provide data management services. The test results with the positive/negative testing method show that the platform prototype can store and deliver the data needed according to the request given to provide centralized data management services. The stress testing test shows that the latency value is quite good, which is around 3 to 15 seconds, so that it can provide services for mass usage with the number of users up to 130 users. The author's suggestion for further research is to develop and test the prototype platform from the security side because this research only focuses on the functionality of the platform prototype.

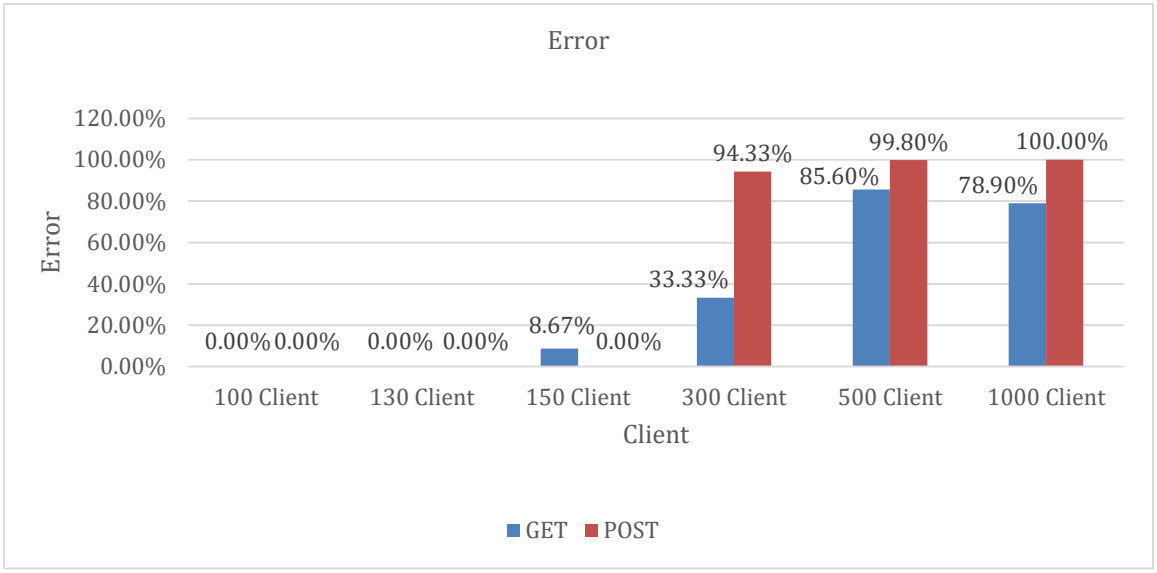


Fig. 3. Error value graph

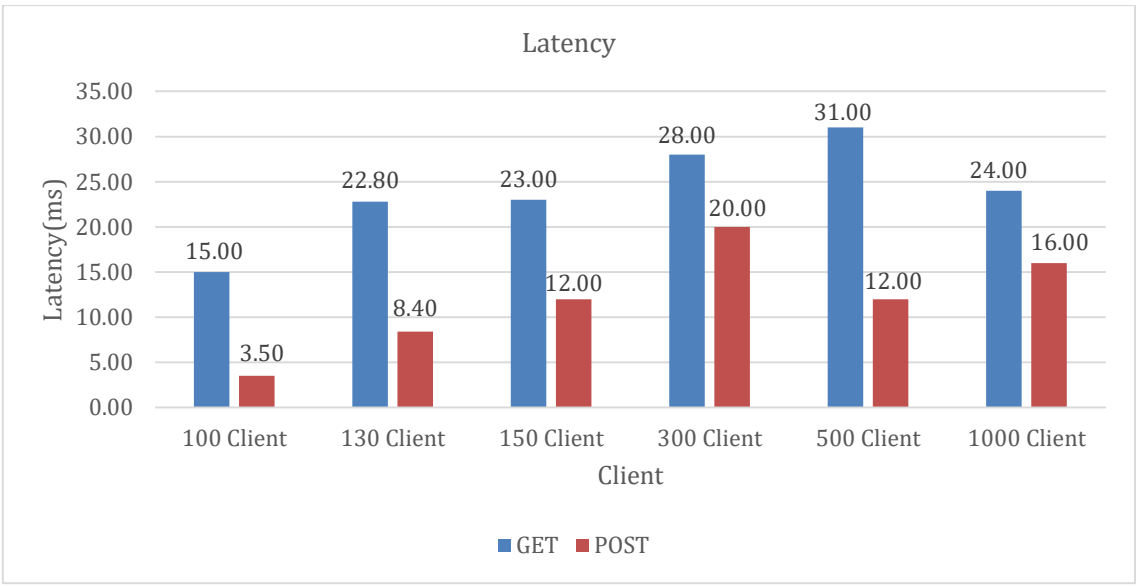


Fig. 4. Latency value graph

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: java.net.SocketTimeoutException/Non HTTP response message: Read timed out	590	59.00%	59.00%
503/Service Unavailable	410	41.00%	41.00%

Fig. 5. Details Errors that occurred during stress testing with 1000 client schemes on endpoints with the POST method

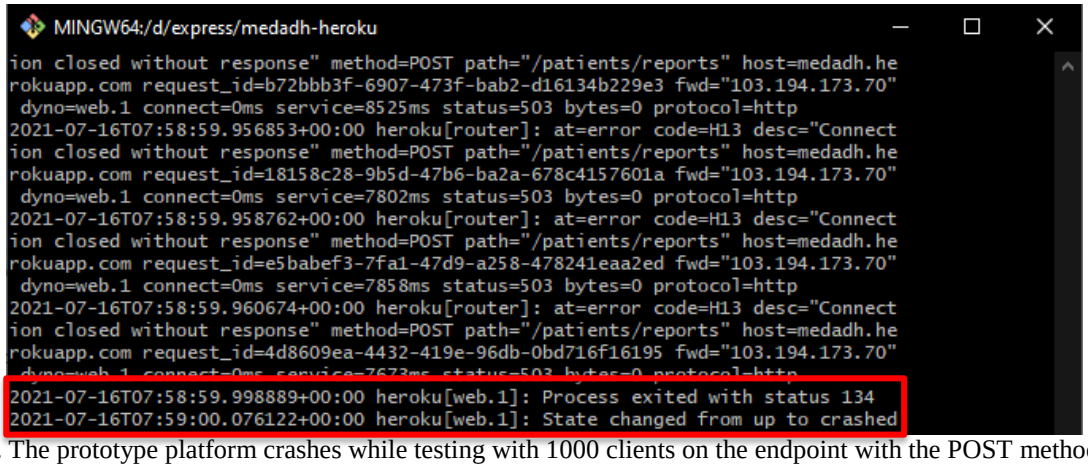


Fig. 6. The prototype platform crashes while testing with 1000 clients on the endpoint with the POST method

REFERENCES

[1] T. R. Zijp, P. G. M. Mol, D. J. Touw, and J. F. M. van Boven, "Smart Medication Adherence Monitoring in Clinical Drug Trials: A Prerequisite for Personalised Medicine?," *EClinicalMedicine*, vol. 15, pp. 3–4, 2019. https://doi.org/10.1016/j.eclinm.2019.08.013

[2] I. Ahmed *et al.*, "Medication adherence apps: Review and content analysis," *JMIR mHealth uHealth*, vol. 6, no. 3, 2018. https://doi.org/10.2196/mhealth.6432

[3] F. Santoleri, R. Lasala, A. Logreco, E. Ranucci, and A. Costantini, "Using a treatment diary to improve the medication adherence in patients with chronic myeloid leukaemia," *J. Oncol. Pharm. Pract.*, vol. 25, no. 5, pp. 1035–1041, 2019. https://doi.org/10.1177/1078155218759184

[4] J. Geissler *et al.*, "Factors influencing adherence in CML and ways to improvement: Results of a patient-driven survey of 2546 patients in 63 countries," *J. Cancer Res. Clin. Oncol.*, vol. 143, no. 7, pp. 1167–1176, 2017. https://doi.org/10.1007/s00432-017-2372-z

[5] W. Shahin, G. A. Kennedy, and I. Stupans, "The impact of personal and cultural beliefs on medication adherence of patients with chronic illnesses: A systematic review," *Patient Prefer. Adherence*, vol. 13, pp. 1019–1035, 2019. https://doi.org/10.2147/PPA.S212046

[6] E. Wiecek, F. S. Tonin, A. Torres-Robles, S. I. Benrimoj, F. Fernandez-Llimos, and V. Garcia-Cardenas, "Temporal effectiveness of interventions to improve medication adherence: A network meta-analysis," *PLoS One*, vol. 14, no. 3, pp. 7–9, 2019. https://doi.org/10.1371/journal.pone.0213432

[7] N. K. Choudhry *et al.*, "Effect of reminder devices on medication adherence: The REMIND randomized clinical trial," *JAMA Intern. Med.*, vol. 177, no. 5, pp. 624–631, 2017. https://doi.org/10.1001/jamainternmed.2016.9627

[8] F. A. Ramadhan, A. Rakhmatsyah, and R. Yasirandi, "Schedule Control System for Wearable Medicine Box Using Bluetooth Low Energy," *2019 Int. Conf. Comput. Sci. Inf. Technol. ICoSNIKOM 2019*, pp. 3–8, 2019. https://doi.org/10.1109/ICoSNIKOM48755.2019.9111618

[9] D. Mohanapriya, V. Deepika, M. ShanmughaPriya, and C. Sivasankari Yogeswari, "A Real Time Support System to Impart Medicine using Smart Dispenser," *2020 Int. Conf. Syst. Comput. Autom. Networking, ICSCAN 2020*, pp. 1–10, 2020. https://doi.org/10.1109/ICSCAN49426.2020.9262424

[10] V. Doshi, N. Mehta, S. Dey, and R. Prasad, "An IoT based smart medicine box," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 5, no. 1, pp. 205–207, 2019. https://www.ijariit.com/manuscript/an-iot-based-smart-medicine-box/

[11] C. Gulden *et al.*, "Prototypical clinical trial registry based on fast healthcare interoperability resources (FHIR): Design and implementation study," *JMIR Med. Informatics*, vol. 9, no. 1, pp. 1–13, 2021. https://doi.org/10.2196/20470

[12] S. El-Sappagh, F. Ali, A. Hendawi, J. H. Jang, and K. S. Kwak, "A mobile health monitoring-and-treatment system based on integration of the SSN sensor ontology and the HL7 FHIR standard," *BMC Med. Inform. Decis. Mak.*, vol. 19, no. 1, pp. 1–36, 2019. https://doi.org/10.1186/s12911-019-0806-z

[13] M. Lehne, S. Luijten, P. V. F. G. Imbusch, S. Thun, and others, "The Use of FHIR in Digital Health-A Review of the Scientific Literature.," *GMDS*, no. September, pp. 52–58, 2019. https://doi.org/10.3233/shti190805

[14] I. A. Sawaneh, A. Kamara, and J. H. Koroma, "A Computerized Patient's Database Management System," *Int. J. Comput. Sci. Inf. Technol. Res.*, vol. 6, no. 2, pp. 6–10, 2018. http://dx.doi.org/10.13140/RG.2.2.12642.22728

[15] T. Lv, P. Yan, and W. He, "Survey on JSON Data Modelling," *J. Phys. Conf. Ser.*, vol. 1069, no. 1, 2018. https://doi.org/10.1088/1742-6596/1069/1/012101

[16] A. R. Breje, R. Gyorödi, C. Gyorödi, D. Zmaranda, and G. Pecherle, "Comparative study of data sending methods for XML and JSON models," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 12, pp. 198–204, 2018. https://doi.org/10.14569/IJACSA.2018.091229

[17] M. Ayaz Sr, M. F. Pasha 2nd, M. Y. Alzahrani 3rd, R. Budiarto 4th, and D. Stiawan 5th, "Fast Health Interoperability

Resources Standard: A Systematic Literature Review (Preprint)," *JMIR Med. Informatics*, vol. 9, pp. 1–21, 2020. https://doi.org/10.2196/21929
[18] G. William, R. Anthony, and J. Purnama, "Development of NodeJS based Backend System with Multiple Storefronts for Batik Online Store," *ACM Int. Conf. Proceeding Ser.*, 2020. https://doi.org/10.1145/3429789.3429830
[19] J. Kumar and V. Garg, "Security analysis of unstructured data in NOSQL MongoDB database," *2017 Int. Conf. Comput. Commun. Technol. Smart Nation, IC3TSN 2017*, vol. 2017-Octob, pp. 300–305, 2018. https://doi.org/10.1109/IC3TSN.2017.8284495
[20] F. S. Alshraiedeh and N. Katuk, "A URI parsing technique and algorithm for anti-pattern detection in RESTful Web services," *Int. J. Web Inf. Syst.*, vol. 17, no. 1, pp. 1–17, 2021. https://doi.org/10.1108/IJWIS-08-2020-0052
[21] A. R. Wiratno and K. Hastuti, "Implementation of Firebase Realtime Database to Track BRT Trans Semarang," *Sci. J. Informatics*, vol. 4, no. 2, pp. 95–103, 2017. https://doi.org/10.15294/sji.v4i2.10829
[22] A. N. A. Yusuf, F. Y. Zulkifli, and I. W. Mustika, "Development of Monitoring and Health Service Information System to Support Smart Health on Android Platform," *4th Int. Conf. Nano Electron. Res. Educ. Towar. Adv. Imaging Sci. Creat. ICNERE 2018*, pp. 3–8, 2019. https://doi.org/10.1109/ICNERE.2018.8642592
[23] T. Hagos, "Android Studio," in *Learn Android Studio 3*, Apress, Berkeley, CA, 2018, pp. 5–17. https://doi.org/10.1007/978-1-4842-3156-2_2
[24] S. Bose, "a Comparative Study: Java Vs Kotlin Programming in Android Application Development," *Int. J. Adv. Res. Comput. Sci.*, vol. 9, no. 3, pp. 41–45, 2018. https://doi.org/10.26483/ijarcs.v9i3.5978
[25] V. Oliveira, L. Teixeira, and F. Ebert, "On the Adoption of Kotlin on Android Development: A Triangulation Study," *SANER 2020 - Proc. 2020 IEEE 27th Int. Conf. Softw. Anal. Evol. Reengineering*, pp. 206–216, 2020. https://doi.org/10.1109/SANER48275.2020.9054859
[26] B. De, "API Testing Strategy," in *API Management*, Apress, Berkeley, CA, 2017, pp. 153–164. https://doi.org/10.1007/978-1-4842-1305-6_9
[27] J. Agnihotri and R. Phalnikar, "Development of performance testing suite using apache JMeter," *Adv. Intell. Syst. Comput.*, vol. 673, pp. 317–326, 2018. https://doi.org/10.1007/978-981-10-7245-1_32
[28] R. Abbas, Z. Sultan, and S. Nazir B, "Comparative Analysis of Single-Core and," *Int. Conf. Commun. Technol.*, vol. 7, no. 6, pp. 117–130, 2017. https://doi.org/10.5121/ijcsit.2015.7610
[29] P. Danielsson, T. Postema, and H. Munir, "Heroku-based innovative platform for web-based deployment in product development at axis," *IEEE Access*, vol. 9, pp. 10805–10819, 2021. https://doi.org/10.1109/ACCESS.2021.3050255
[30] A. Khasawneh, H. Rogers, J. Bertrand, K. C. Madathil, and A. Gramopadhye, "Human adaptation to latency in teleoperated multi-robot human-agent search and rescue teams," *Autom. Constr.*, vol. 99, no. January 2018, pp. 265–277, 2019. https://doi.org/10.1016/j.autcon.2018.12.012