Developing of Middleware and Cross Platform Chat Application

Study Case: Telegram, LINE

Danny Sebastian[1], Restyandito[2], Kristian Adi Nugraha[3]
Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana, Yogyakarta, Indonesia

***Abstract*—The rapid development of technology has resulted in many new innovations on social media platforms. Now-a-days, there are many chat applications available, namely Whatsapp, Telegram, LINE, Viber, and many others. This in turn forces users to juggle between many chat applications as different applications can't communicate with each other. This research aims to develop a chat application which serves as a middleware to make communication between developed chat application and two conventional chat applications possible (Telegram and LINE). Several tests are done to ensure that the message exchange process (in text, picture, video, and file type) works well between the developed chat application as well as Telegram or LINE.**

***Keywords*—Telegram API; line API; chat application; flutter; middleware**

I. Introduction

The rapid development of technology has invented many new innovations on social media platforms. There are so many social media applications available, yet this doesn't stop the emergence of new social media applications. In Indonesia, there are so many social media platforms available, with Line, Telegram, Whatsapp, and Viber being some of the most notorious social media platforms offering its service in Indonesia. Social media platforms generally offer messages, pictures, voice message, file exchanges, and other things [1]. Every chatting application offers different features available for the users to use. For instance, Telegram offers the file upload feature, a feature that LINE has yet to offer [2] [3].

Every user has their own preferences in choosing which social media platforms they want to use. Oftentimes, the amount of acquaintances using a certain social media platform being the main consideration on which social media platform they are going to use. This happens because of the limitation in which users can only exchange messages within the same social media platforms. To this day, message exchanges between different social media platforms are still impossible.

Usually, each user has their own preferences in choosing chat apps. In fact, usually the selection of chat apps depends on the community group, each community group has their own favorite chat apps. This condition makes it difficult when someone joins several community groups, and each community group uses different chat apps. Different features offered by each social media platform provider and limitations on communicating using different platforms forced the users to choose which platforms they are going to use. Oftentimes, often users have to use a lot of social media and have an account in each chat application.

The writer feels the need to research and develop a custom chat application and middleware which will connect several social media platforms. In this research, LINE and Telegram chatting applications are used. This article is divided into five parts, namely, introduction, literature review, research methods, results and findings, and conclusions.

II. Literature Review

Chatting in Indonesian means communication between a person with another person or people [4]. In the computing world, chatting means communication between 2 or more people using computer devices [4]. Nowadays, chatting applications are growing rapidly, with many chatting applications being developed to fulfill the users' communication needs. There are so many features offered by social media platforms nowadays, namely files transfer, auto response/bot [5] [6], business features, gaming features [7], and many more. Chatting application providers aren't always big companies such as Whatsapp, Telegram, Viber, LINE, but also small developers. Hence, several chatting applications made it possible to communicate with other applications using Application Programming Interface (API) [2] [3].

Middleware is a software application which can connect a system with another system [7]. Middleware can be used to connect several systems within the same device or even on different devices connected to the internet [8] [9]. Middleware can also be used to connect applications on the same type of device or different types of devices, such as mobile device - mobile phone, mobile phone - television, mobile phone - computer, computer - computer, et cetera [10]. In developing a middleware application, there are a few solutions/methods, namely message oriented middleware [11], object-oriented middleware [12], Remote procedure call, database middleware, transactional middleware, portals, embedded middleware [13], and content-centric middleware [14].

Middleware development was also done to bridge a chatting application with other application. Several researchers have developed middleware for chatting applications to add features, such as Artificial Intelligence [15]. Some researchers developed middleware from scratch and some other used API provided by the chatting applications provider [16]. Other than API, webhook method also used to send messages between chat bots.

Several researches and development on custom chatting applications has been done. A custom chatting application equipped with Natural Language Processing was developed in one research [17]. In this research the system will automatically do sentiment analysis towards the message being sent. If the analyzed message has negative context, then the message will not be sent. In another research, a custom location based chatting application was developed [18], in which the application allows the users to find friends and communicate with other users within a certain distance. Another research was also done, intending to help the communication between faculty members (lecturer, assistant lecturer) and the students [19]. The custom chatting application developed was able to automatically create a group chat based on the subjects' registration done every semester.

Several researches on chatting application development using the API offered by big chatting application providers, namely Telegram, LINE, et cetera was also done. Some developed a smart home system using NodeMCU Microcontroller combined with Telegram API [20]. By using the application developed in this research, users are able to monitor and command their Internet of Things devices using Telegram. Telegram Bot API was used to send messages from Telegram to the Internet of Things devices. Other than that, the Telegram BOT API was also used to create an e-complaint application for a college [21]. In this application, the Telegram Bot API was used to receive complaints and calculate the complaints statistics based on the divisions being complained to.

In summary, custom chatting applications that were developed are Android based [1] [17] [18], iOS based [1] [22], website applications [4] [23], and desktop application [24]. Android based mobile applications can be developed using either Java or Kotlin, while iOS based mobile applications are developed using either objective-C or Swift. In the application development community, there is a new trend which is a cross-platform application, in which the developed applications can be compiled and create both an Android and iOS based application using a single code base. One of the frameworks used to create this cross-platform device is the Flutter Framework. Flutter Framework itself is an open source cross platform development framework developed by Google [25]. Flutter itself is based on the Dart Programming Language. Several technology company giants were using Flutter Framework to develop their products, namely, Alibaba and Google Ads.

A. Chat API

Application Programming Interface (API) is used by an application to exchange information with other applications [26]. API success relies on the API documentation provided by API for software development needs. Many chat applications have provided API which allows other application to access the chat applications' services.

Telegram provides API for software developers to connect their applications to Telegram's system. This API allows Telegram Bot creation [2]. Telegram Bot itself acts as an interface to run code from a server. Telegram API uses text in JSON format in passing data with other systems. This JSON formatted text allows developers to develop application using many different programming languages.

Line Messaging API is a service provided by LINE to exchange data between Line Platform and other application [3]. Just like Telegram Bot API, LINE Messaging API uses JSON to communicate with other applications. LINE Messaging API uses webhook method to pass data to the server.

B. NoSQL Database

NoSQL database are databases that don't use SQL command in which data was saved in an unstructured format and often time don't have relations with other table like SQL databases [27] [28]. NoSQL database was intended to save data in a flexible way in modern application development. In many cases, NoSQL databases used in real-time application development.

In general there are four types of NoSQL Database [29]:

- Graph databases: These databases uses graph theory concept. Example: Neo4j and Titan.
- Key-Value store databases: In these databases, data are stored in two parts, which are key and value. Example: Redis, DyanmoDB, Riak.
- Column Store databases: In these databases, data are stored in column of data. Example: BigTable, Cassandra.
- Document Databases: These databases are more extensive database than the key-value store. The value are saved in document type and stored in a ore complex format, like JSON. Example: MongoDB, CouchDB.

C. Flutter

Flutter is an open source mobile application development made by Google [30]. Flutter allows Android and iOS based application development using only one source code base [31]. Flutter uses Dart Programming Language. Flutter Framework uses widget concept in interface creation. There are many widgets provided, namely Column, Row, Icon, and many other widgets. The widget in Flutter acts are either visual component or as a container for other widgets [32].

III. Research Method

A. System Design

The system developed consists of two main applications, namely mobile chatting application and middleware application. The chatting application is used as an interface for the user to test the system. This chatting application was developed using Dart programming Language with Flutter Framework. While the middleware acts as a connector to connect the chatting application developed with Telegram API, LINE API.

Architecture of the system developed can be seen on Fig. 1. Message exchange process starts on one of the Conventional Chat Application/CCA (Telegram/LINE/Signal) to the CCA's chat API (step 1). Then the CCA's API will pass the message to the middleware to be received by the webhook

prepared (step 2). The middleware will then process the message fetched by saving the message's metadata, saving the file, image, video, our sound data to Firebase. The middleware developed uses 3 Firebase service, namely Firestore, Firebase Cloud Storage, and Firebase Cloud Messaging. Firestore is used to save the messages' metadata and content, such as the message's recipient, message's sender, chatting application, etc. Example of data stored in Firestore can be seen on Fig. 2 for text data and Fig. 3 for non-text data. Firebase Cloud Storage is used to store video, image, voice, and file message data. Example of data stored in Firebase Cloud Storage can be seen on Fig. 4. After being processed, the middleware will then pass the message to Flutter Chat Application/FCA (step 3).

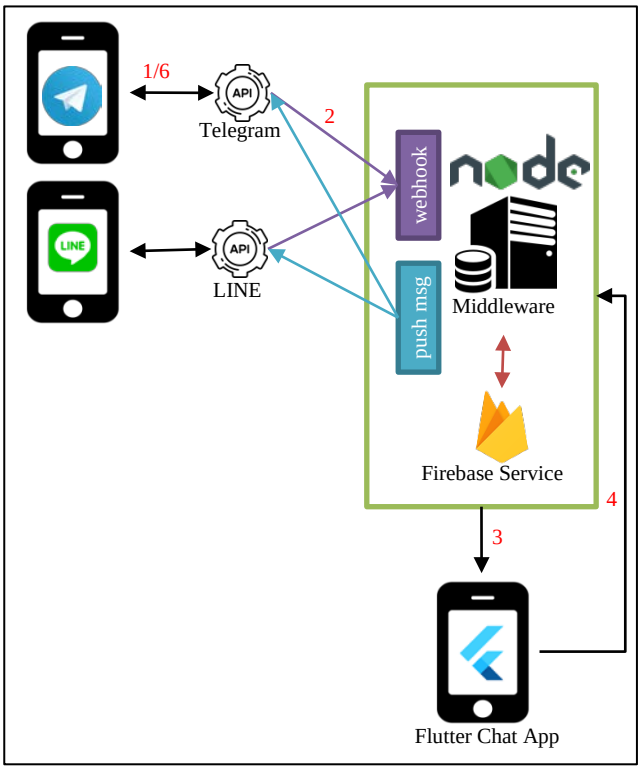

Fig. 1. Whole System Application Architecture.

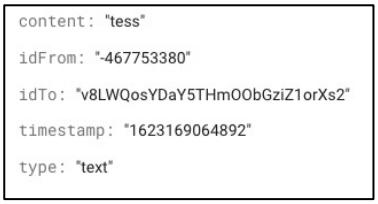

Fig. 2. Example of Data Stored in Firestore, Type:Text Message.

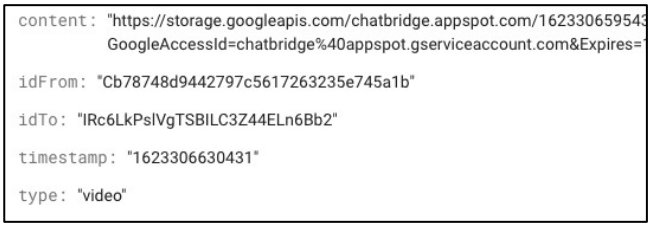

Fig. 3. Example of Data Stored in Firestore, Type:Video.

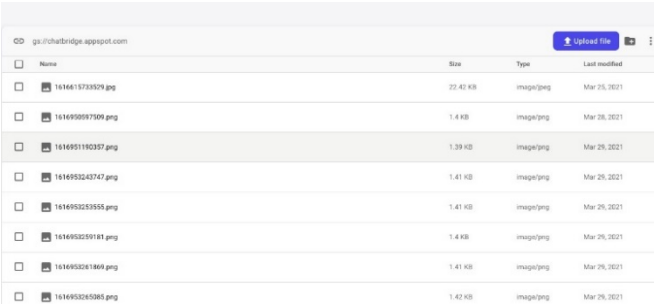

Fig. 4. Example of Data Stored in Firebase Cloud Storage.

Meanwhile, messages passed from FCA to CCA starts from (step 4), where FCA forward messages to the middleware. Messages from FCA will then be processed on the middleware and stored in Firebase services. After that, messages will then be forwarded to the recipient's CCA through Chat Application API using Push Message (step 5/6). Available API could be seen at Table I.

After the application has been developed, system test was done. This system test was done to make sure that the application has successfully work as expected. The system test scenario can be found on Table II. System test was done on Text, Image, Video, Sound and File type. For each message type, testing was done from CCA to FCA, and vice versa.

B. System Test

Performance testing was done to measure the time needed by the middleware to forward and receive messages. The performance testing scenario can be seen on Table III. In general, the testing was done using four types of data, namely, Text, Image, Video and File. For text data type, size of the data forwarded was classified based on the number of characters. As for the Image, Video and File data type, data was classified based on the file size (in megabyte). Testing for the Text, Image, and Video data was done on Telegram, and LINE. However, File data wasn't tested on LINE as the chatting application doesn't have file sharing feature.

TABLE I. APPLICATION PROGRAMMING INTERFACE (API) MIDDLEWARE

Method	URI	Description
POST	/telegram/webhook	Receive messages from telegram bot and forward it to the FCA
POST	/telegram/push	Receive messages from the FCA and forward them to Telegram API
POST	/line/webhook	Receive message from LINE bot and forward them to the FCA
POST	/line/push	Receive messages from the FCA and forward it to LINE API

TABLE II. SYSTEM TEST SCENARIO

Method	URI	Description
CCA-FCA, FCA-CCA	Text	Telegram, LINE
CCA-FCA, FCA-CCA	Image	Telegram, LINE
CCA-FCA, FCA-CCA	Video	Telegram, LINE
CCA-FCA, FCA-CCA	Sound	Telegram, LINE
CCA-FCA, FCA-CCA	File	Telegram

TABLE III. PERFORMANCE TEST SCENARIO

Type	Chat Application	Measurement	Size
Text	Telegram, LINE	Characters	400, 800, 1200, 1600, 2000, 2400, 2800, 3200, 3600, 4000
Image	Telegram, LINE	MB	1, 2, 3, 4, 5
Video	Telegram, LINE	MB	1, 2, 3, 4, 5
File	Telegram	MB	1, 2, 3, 4, 5

The system developed has start and end timer. In the FCA to CCA testing, the start timer was invoked when the "Send" button on the FCA is clicked, while the stop timer was invoked when the middleware sends out push message to the Chat API. As for the CCA to FCA testing, the start tier was invoked when the middleware webhook receive request, while the stop timer was invoked when the message has been forwarded by the middleware to FCA.

$$process\_time = \frac{(t_1+t_2+t_3+t_4+t_5)}{5} \quad (1)$$

In this testing, the possibility of unstable internet connection may be a problem. To tackle this problem, every test scenario was done 5 times and average processing time will be calculated to then be used as a final result. Average process tie formula can be seen on equation (1). For example, Telegram Text data type testing for 400 characters processing time was measured on 0.5 second, 0.7 second, and 0.63 second. Thus, the processing time for this test case is 0.61 second. Each test case will be carried out for testing from CCA to FCA and vice versa. Performance time result will be compared for CCA to FCA and FCA to CCA data.

IV. RESULT AND FINDINGS

A. System Test

System testing has been done and the result can be seen on Table IV. All system testing scenario can be done by the chat application's middleware and Flutter Chat Application (FCA). Captures of the Flutter Chat Application can be seen on Fig. 5. Based on the testing result, the middleware application developed has successfully able to forward messages from the Flutter Chat Application (FCA) to the Conventional Chat Application (CCA) and vice versa. This success also applies for all message types tested.

Currently the architecture and communication process starts by creating a chat group on Telegram/LINE, then an OTP request is made to be able to start communication between the custom chat app and the Telegram/LINE chat app. Currently, the architecture and communication processes in the middleware that are built are still unable to communicate between LINE and Telegram. This is because there is an OTP request that must be made so that communication can be carried out. Due to this limitation, it is necessary to adjust the add contact process. On the other hand, when chatting, the middleware needs to add fields recording where the message was sent from and where the message was sent.

TABLE IV. SYSTEM TEST RESULT

From-To	Message Type	CCA	Result
CCA-FCA	Text	Telegram	*Pass*
		LINE	*Pass*
FCA-CCA		Telegram	*Pass*
		LINE	*Pass*
CCA-FCA	Image	Telegram	*Pass*
		LINE	*Pass*
FCA-CCA		Telegram	*Pass*
		LINE	*Pass*
CCA-FCA	Video	Telegram	*Pass*
		LINE	*Pass*
FCA-CCA		Telegram	*Pass*
		LINE	*Pass*
CCA-FCA	File	Telegram	*Pass*
FCA-CCA		Telegram	*Pass*

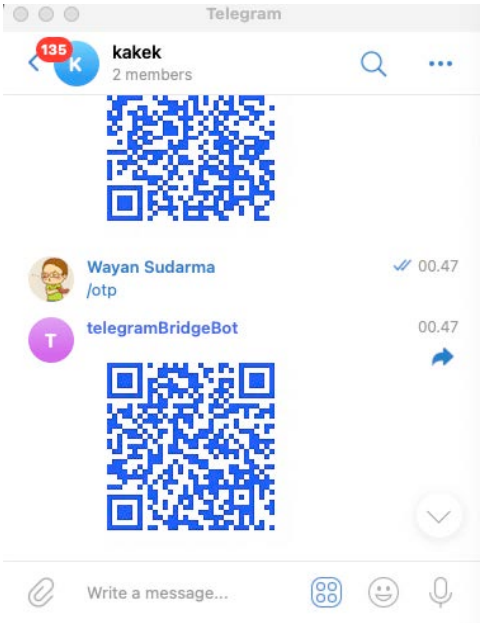

Fig. 5. Screenshot user Interface Application Flutter Chat Application.

B. System Performance Evaluation

System Performance Test was performed on each of the message type sent. The message types tested were text, picture, video, and file messages. Testing result for text data can be seen on Table V and Fig. 6. The results show that the messages sent from the CCA to the FCA took longer than the messages sent from the FCA to the CCA. As seen in Fig. 6 there was a significant increase in time needed to forward a message containing 2800 characters from FCA to Telegram (represented by the orange line).

This increase in time may be caused by the mobile network used for testing. However, in general there were no significant increases in time when the character-count is increased.

TABLE V. SYSTEM PERFORMANCE TEST (TEXT)

Char	Telegram		LINE	
length	Tele-FCA	FCA-Tele	LINE-FCA	FCA-LINE
400	144	1070.6	121.4	706
800	163.8	728.8	118	822
1200	164.4	721.8	156	667.2
1600	166.4	817	139.2	711.4
2000	281.4	972	122.6	669.4
2400	126.8	809	126.6	731.2
2800	150.8	1476.2	147	754
3200	289.8	762.4	216.2	1120.6
3600	194.8	766	232.2	783.6
4000	185.8	830.4	106.2	925.8

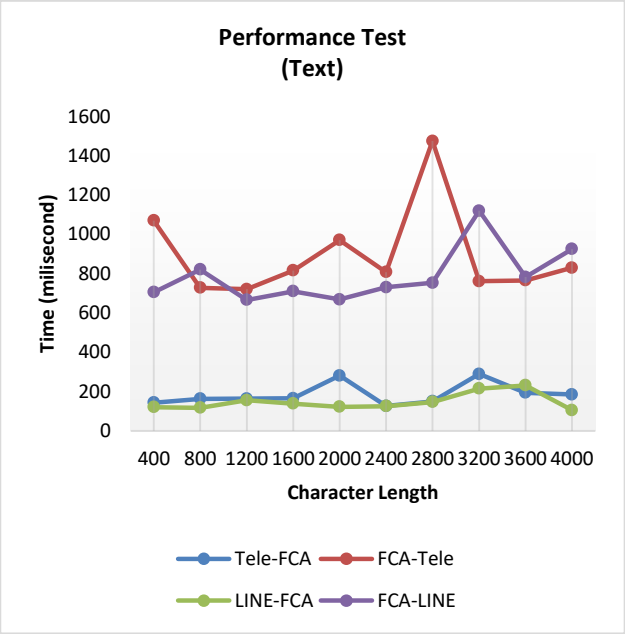

Fig. 6. System Performance Test (Text).

The second testing was done on image messages. The test result can be seen on Table VI and Fig. 7. It can be seen that there is a significant time increase on messages sent from the FCA to Telegram (displayed in orange line) and from FCA to LINE (displayed in yellow line) for image with 5MB in size. Based on the testing, there is no significant time difference between messages sent from the FCA to the CCA and from the CCA to the FCA.

The third testing was done on video messages. The test result can be found on Table VII and Fig. 8. Based on the testing done, it can be seen that there is a significant increase in time aligned with the increase of file size for all scenarios. However, a significant increase in time was most noticeable on messages sent from the CCA to the FCA (displayed in blue and grey line). In general, it can be seen that messages sent from the CCA to the FCA (displayed in blue and grey line) require more time than messages sent from the FCA to the CCA (displayed in orange and yellow line).

TABLE VI. SYSTEM PERFORMANCE TEST (IMAGE)

Size	Telegram		LINE	
(MB)	Tele-FCA	FCA-Tele	LINE-FCA	FCA-LINE
1	5939.6	8217	10526.2	5164.4
2	4592.8	10162	8015.4	5383.6
3	5426.2	6477.2	8793.2	5379.2
4	5979.6	5059.2	8945.4	4903.2
5	5629.4	21017.6	8009.2	21473.2

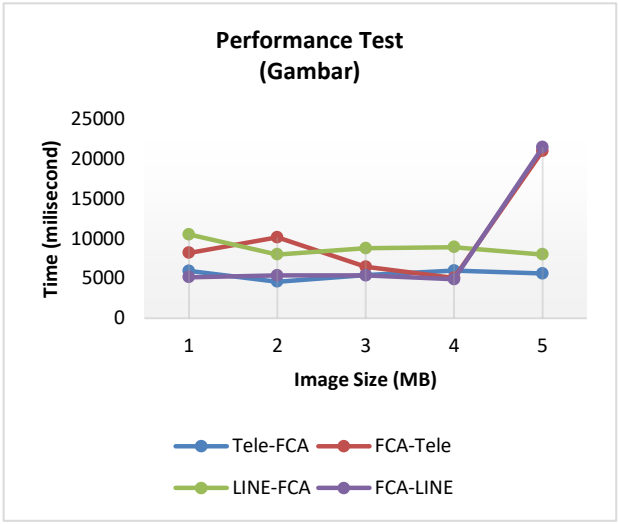

Fig. 7. System Performance Test (Image).

TABLE VII. SYSTEM PERFORMANCE TEST (VIDEO)

Size	Telegram		LINE	
(MB)	Tele-FCA	FCA-Tele	LINE-FCA	FCA-LINE
1	10294.75	6816.2	24761	8083.8
2	20547.6	9381.4	32932.6	10983.8
3	23186.6	8190.6	47755.4	12659.2
4	34914.2	10264.4	51798.8	17165.6
5	48456	10883.6	58812.6	20204

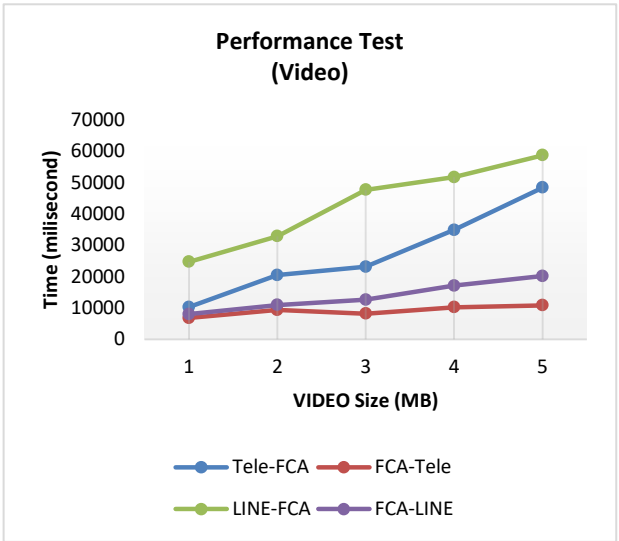

Fig. 8. System Performance Test (Video).

The fourth testing was done on file type messages. This testing was created specifically for Telegram as LINE does not yet offer this message type. The testing result can be seen on Table VIII and Fig. 9. Based on the testing done, it can be seen that there is a significant increase in processing time as the file size increases.

TABLE VIII. SYSTEM PERFORMANCE TEST (FILE)

Size	Telegram	
(MB)	Tele-FCA	FCA-Tele
1	15183.6	9957
2	21697	12236
3	25806.4	15778.6
4	35879.4	17901.8
5	40866.8	22704.6

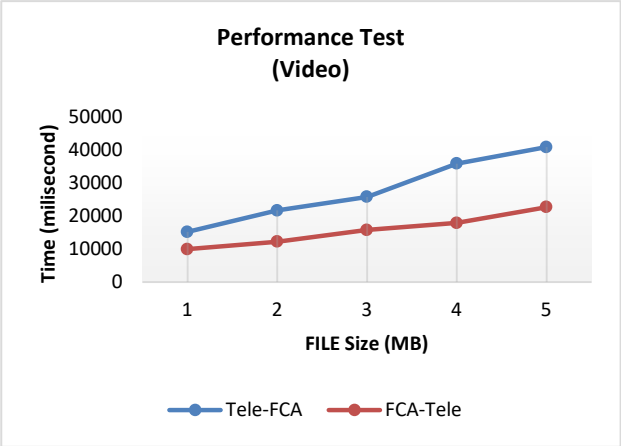

Fig. 9. System Performance Test (File).

V. CONCLUSION

Based on this research, it is concluded that:

- Middleware application was able to exchange messages between the developed chatting application based on Flutter and Conventional Chatting Application (Telegram and LINE), with text, pictures, videos, voice, and file being the type of messages exchanged.
- For video and file messages, there is a correlation between file size and the time needed to forward the message. The bigger the file, the longer it takes to send the file.

Suggestion for future research,

- Adding other conventional chatting application which can be served by the middleware.

ACKNOWLEDGMENT

The writers would like to thank Duta Wacana Christian University and Indonesian Ministry of Research and Technology. This research was funded by Indonesian Ministry of Research and Technology with contract number: 311/E4.1/AK.04.PT/2021 dated 12 July 2021, 3281.5/LL5/PG/2021/22 July 2021 and 264/D.01/LPPM/2021/23Juli 2021.

REFERENCES

[1] N. Sabah, J. M. Kadhim and B. N. Dhannoon, "Developing an End-to-End Secure Chat Application," IJCSNS, vol. 17, no. 11, p. 108, 2017.

[2] Telegram, "Telegram APIs," Telegram, [Online]. Available: https://core.telegram.org/. [Accessed 12 08 2020].

[3] LINE Corp, "LINE Messaging API," LINE Corp, 2021. [Online]. [Accessed 20 06 2021].

[4] D. Henriyan, D. P. Subiyanti, R. Fauzian, D. Anggraini, M. V. G. Aziz and A. S. Prihatmanto, "Design and implementation of web based real time chat interfacing server," in 2016 6th International Conference on System Engineering and Technology (ICSET), Bandung, Indonesia, 2016.

[5] R. Parlika, S. I. Pradika, A. M. Hakim and K. R. NM, "BOT Whatsapp Sebagai Pemberi Data Statistik Covid-19 Menggunakan PHP, Flask, dan MySQL," Jurnal Informatika dan Sistem Informasi (JIFoSI), vol. 1, no. 2, pp. 282-293, 2020.

[6] M. Vorontsov and S. I. Radmir, "Automation of Message Sending Processes Using Specialized Software," in 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg, Moscow, Russia, 2021.

[7] A. N. Wulanjani, "Discord Application: Turning a Voice Chat Application for Gamers into a Virtual Listening Class," in Education 4.0: Trends and Future Perspectives in English Education, Linguistics, Literature, and Translation, Semarang, Indonesia, 2018.

[8] K. Geihs, "Middleware challenges ahead," Computer, vol. 34, no. 6, pp. 24-31, 2001.

[9] M. Saranya and A. A. Priya, "A Study on Middleware Technologies in Cloud Computing," International Journal for Innovative Research in Science & Technology (IJIRST), vol. 4, no. 3, pp. 31-36, 2017.

[10] L. F. Meloni, G. C. Costa, G. Kobayashi and C. S. Kurashima, "Implementation of chat application for ginga middleware technology using second screen," in 2016 IEEE international symposium on consumer electronics (ISCE), Sao Paulo, Brazil, 2016.

[11] J. Yongguo, L. Qiang, Q. Changshuai, S. Jian and L. Qianqian, "Message-oriented Middleware: A Review," in 2019 5th International Conference on Big Data Computing and Communications (BIGCOM), QingDao, China, 2019.

[12] M. Henning, "A new approach to object-oriented middleware," IEEE Internet Computing, vol. 8, no. 1, pp. 66-75, 2004.

[13] J. Zhang, M. Ma, P. Wang and X.-d. Sun, "Middleware for the Internet of Things: A survey on requirements, enabling technologies, and solutions," Journal of Systems Architecture, vol. 117, 2021.

[14] G. S. Wedpathak, "An Approach of Software Engineering through Middleware," International Journal of Engineering and Management Research (IJEMR), vol. 5, no. 1, pp. 127-138, 2015.

[15] P. Thosani, M. Sinkar, J. Vaghasiya and R. Shankarmani, "A Self Learning Chat-Bot From User Interactions and Preferences," in 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020.

[16] C. E. Swandi, K. A. Nugraha, D. Sebastian and Restyandito, "Middleware Development to Connect Telegram Messenger and Instant Messenger for the Elderly," in The 5th International Conference on Information Technology and Digital Applications (ICITDA 2020), Yogyakarta, Indonesia, 2021.

[17] S. Karthick, R. J. Victor, S. Manikandan and B. Goswami, "Professional chat application based on natural language processing," in 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), Bangalore, India, 2018.

[18] M. Kamruzzaman, "Localized Chat Application," Daffodil International University, 2018.

[19] V. Efendy, K. A. Nugraha and D. Sebastian, "Implementasi Chat Room dan Push Notification pada e-Class Berbasis Mobile," Jurnal Teknik Informatika dan Sistem Informasi, vol. 5, no. 2, pp. 267-282, 2019.

[20] Y. Findawati, A. Idris, Y. Rachmawati and E. A. Suprayitno, "IoT-Based Smart Home Controller Using NodeMCU Lua V3 Microcontroller and Telegram Chat Application," in International Conference on Engineering, Technologies, and Applied Sciences (ICETsAS), Bengkulu, Indonesia, 2020.

[21] N. Rosid, A. Rachmadany, M. Multazam, A. Nandiyanto, A. Abdullah and I. Widiaty, "Integration telegram bot on e-complaint applications in college," in The 2nd Annual Applied Science and Engineering Conference (AASEC 2017), Bandung, Indonesia, 2018.

[22] H. Engoren and E. Zorn, "Bridgr: An iOS Application for Organizing and Discussing Long-Distance Carpooling," 2019.

[23] E. Kho, V. C. Mawardi and N. J. Perdana, "Web-based Live Chat Application uses Advanced Encryption Standard Methods and Rivest Shamir Adleman," in 3rd Tarumanagara International Conference of the Applications of Technology and Engineering (TICATE), Jakarta, Indonesia, 2020.

[24] N. V. Vukadinovic, "WhatsApp Forensics: Locating Artifacts in Web and Desktop Clients," Purdue University Graduate School, West Lafayette, 2019.

[25] M. Szczepanik and M. Kedziora, "State Management and Software Architecture Approaches in Cross-platform Flutter Applications," in 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2020), 2020.

[26] M. Meng, S. Steinhardt and A. Schubert, "Application programming interface documentation: what do software developers want?," Journal of Technical Writing and Communication, vol. 48, no. 3, pp. 295-330, 2018.

[27] A. Davoudian, L. Chen and M. Liu, "A survey on NoSQL stores," ACM Computing Surveys (CSUR), vol. 51, no. 2, pp. 1-43, 2018.

[28] A. Moniruzzaman and S. A. Hossain, "Nosql database: New era of databases for big data analytics-classification, characteristics and comparison," International Journal of Database Theory and Application, vol. 6, no. 4, 2013.

[29] A. Gupta, S. Tyagi, N. Panwar, S. Sachdeva and U. Saxena, "NoSQL Databases: Critical Analysis and Comparison," in 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), Gurgaon, India, 2017.

[30] Flutter Dev, "Flutter Documentation," Google, 2017. [Online]. Available: https://flutter.dev/docs. [Accessed 26 06 2021].

[31] K. Wasilewski and W. Zabierowski, "A Comparison of Java, Flutter and Kotlin/Native Technologies for Sensor Data-Driven Applications," Sensors, vol. 21, no. 10, 2021.

[32] S. Santoso, D. J. Surjawan and E. D. Handoyo, "Pengembangan Sistem Informasi Tukar Barang Untuk Pemanfaatan Barang tidak Terpakai dengan Flutter Framework," Jurnal Teknik Informatika dan Sistem Informasi (JuTISI), vol. 6, no. 3, pp. 589-598, 2020.