IMPLEMENTASI *CHATBOT* PADA WHATSAPP UNTUK MONITORING SUMBER DAYA SERVER

Laily Rohmawati [1] **, Muhammad Agung nugroho** [2] **, Wagito** [3]

[1,2,3] *Informatika Universitas Teknologi Digital Indonesia*
email : lailyrahma8@gmail.com[1], m.agung.n@utdi.ac.id[2], wagito@utdi.ac.id[3]

Abstraksi

Monitoring server merupakan metode untuk melakukan pemantauan kinerja dari suatu sumber daya pada suatu server. Monitoring server membantu syadmin untuk mengetahui penggunaan sumber daya sistem seperti, CPU, penggunaan memori, *input* dan *output*, *network*, dan *disk*. Untuk menjaga stabilitas performa server diperlukan mekanisme monitoring. Monitoring menjadi bagian dari tugas system administrator dalam melakukan pemantauan server selama 24 jam. Namun *system administrator* tidak dapat terus-menerus melakukan pemantauan secara penuh selama 24 jam, sehingga peneliti melakukan percobaan implementasi *chatbot* untuk dapat memonitor server secara *real time* dan dapat diakses di mana saja. Dari Hasil pengujian, aplikasi *chatbot* yang dikembangkan dapat melakukan tugas dalam memonitor server dengan menampilkan informasi sumber daya aplikasi atau layanan yang berjalan pada server secara *real time* melalui *chat* pada whatsapp.

***Kata Kunci*:**

chatbot, manajemen server, monitoring Server, whatsapp API

Abstract

Server monitoring is a method for monitoring the performance of a resource on a server. Server monitoring helps sysadmins determine the use of system resources such as CPU, memory usage, input and output, network, and disk. To maintain the stability of server performance, a monitoring mechanism is needed. Monitoring is part of the system administrator's duties in monitoring the server for 24 hours. However, the system administrator cannot continuously monitor for 24 hours, so the researchers experimented with implementing chatbots to monitor servers in real time. From the test results, the developed chatbot application can perform tasks in monitoring servers by displaying information on application resources or services running on the server in real-time via chat on WhatsApp..

***Keywords*:**

chatbot, management server, monitoring server, whatsapp API

Pendahuluan

Sistem monitoring jaringan merupakan metode dalam melakukan pengamatan dan monitoring sistem jaringan komputer yang berjalan dan memungkinkan deteksi dini saat terjadi permasalahan pada jaringan. Sistem monitoring diterapkan pada perangkat atau server tertentu yang terhubung ke jaringan lokal ataupun jaringan internet. Monitoring dapat memberikan informasi terkait kondisi server atau perangkat yang dimonitor. Informasi yang diperoleh dapat menjadi data untuk menganalisis kondisi server atau jaringan. Namun, permasalahan yang umum terjadi di dalam perusahaan yang memiliki server baremetal salah satunya yaitu masalah fleksibilitas dalam proses pengawasan jika terjadi permasalahan human error terutama sysadmin yang bertugas dalam mengamati kinerja server selama 24 jam [1]. *Chatting* merupakan aplikasi yang digunakan untuk berkomunikasi melalui pesan text, gambar, audio, ataupun video antar perangkat dalam lingkup jaringan. Implementasinya membuat aplikasi ini berjalan dengan efisien dan ekonomis [2]. Aplikasi ini berjalan dari sisi *client* dan *server*, dimana *server* berjalan memiliki *socket* dan terhubung dalam *port* tertentu dan menunggu permintaan klien untuk melakukan sambungan. Sementara itu, klien berjalan pada perangkat tertentu dengan memanfaatkan koneksivitas dengan *server* melalui *port* tertentu.

WhatsApp merupakan aplikasi yang memiliki fungsi untuk berkirim pesan instan, yang dapat bekerja seperti sistem SMS (Short Messege Service) pada *feature phone*. Namun WhatsApp tidak mmenggunakan pulsa, tetapi memanfaatkan layanan data dengan internet [3]. WhatsApp atau dikenal dengan WA menjadi salah satu media sosial paling aktif digunakan oleh masyarakat Indonesia dimana sebanyak 83% pengguna internet atau sekitar 124 juta adalah pengguna WhatsApp [4].

Pada penelitian sebelumnya, sistem *chatbot* yang digunakan untuk monitoring server [5] memanfaatkan teknologi chat dari telegram API [6]. Pengembangan penelitian dari sistem monitoring tersebut dapat diterapkan dengan untuk monitoring

keamanan sistem [7] dan pengembangan sistem yang dapat berjalan pada multikanal [8]. Dari penelitian-penelitian tersebut, penulis akan melakukan perancangan dan implementasi untuk mengembangkan sistem monitoring *chatbot* yang dapat diimplementasikan pada aplikasi chat whatsapp yang secara fitur dapat memberikan informasi sumber daya server dan memberikan notifikasi peringatan kepada *sysadmin* saat layanan pada server mengalami kendala.

Tinjauan Pustaka

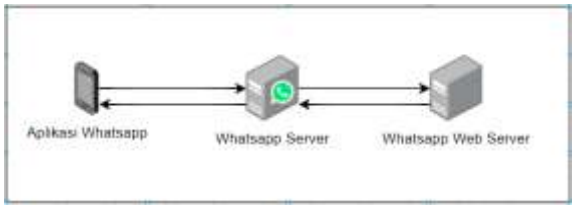
Penelitian ini memanfaatkan teknologi *web services* yang merupakan aplikasi dengan data (*database*), perangkat lunak (*software*) atau bagian dari perangkat lunak yang diakses secara *remote* dengan berbagai perangkat. Web service dapat diidentifikasikan dengan menggunakan URL seperti web pada umumnya. Namun yang menjadi perbedaan *web service* dengan web pada umumnya adalah interaksi yang diberikan. URL *web service* hanya menggandung kumpulan informasi, perintah, konfigurasi atau sintaks yang berguna dalam membangun fungsi tertentu dari aplikasi [9]. Selain itu, model layanan modern saat ini mulai menerapkan konsep *devops* untuk pengembangan dan proses otomasi dalam infrastruktur dan aplikasi, *DevOps* merupakan budaya atau mekanisme dimana proses antara *developer* aplikasi dan tim *ops engineer* berkolaborasi agar dapat melakukan proses *build*, *test* dan *release* perangkat lunak dapat dilakukan lebih cepat dan handal. Praktik implementasi *DevOps* yang baik dapat menghasilkan produk yang stabil dan memiliki *added value*. Konsep ini diterapkan pada pengembangan aplikasi *chatbot* untuk pembelajaran di sekolah [10]. Teknologi *chatbot* dapat diterapkan sebagai knowledge based untuk menangani permasalahan akademik dengan model FAQ [11], model ini dapat diimplementasikan dalam menjawab pertanyaan-pertanyaan akademik mahasiswa pada kampus STMIK AKAKOM.

Penerapan web service dan model *devops* dapat dimanfaatkan untuk membangun otomisasi dengan pengembangan aplikasi *bot* berbasis chat. Salah satu aplikasi chat yang dapat digunakan adalah telegram. Telegram merupakan aplikasi perpesanan (chat) yang digunakan masyarakat untuk berkomunikasi. Telegram mempunyai fitur-fitur yang dapat membuat berkomunikasi menjadi mudah dan menarik. Selain untuk masyarakat pada umumnya yang penerapannya hanya untuk berkomunikasi, telegram juga mempunyai fitur khusus bagi pengembang yang ingin membuat aplikasi chat ini dimodifikasi sesuai keinginan pengembang [12]. *Bot* telegram juga dapat digunakan untuk monitoring sistem keamanan jaringan dengan memberikan notifikasi saat terjadi penyerangan pada sistem, pengembangan ini memanfaatkan teknologi NodeJS [7].

Dalam implementasi lain, *Bot* telegram dapat dioptimalisasikan untuk proses monitoring pada server. Monitoring server yang digunakan memanfaatkan teknologi load balancing dan microservice berbasis container. Bot ini dapat memberikan informasi mengenai kondisi server yang dimonitoring dengan menerapkan metode PPDIOO untuk memberikan akurasi dan kecepatan informasi. [6]. Monitoring server dengan menggunakan *bot* telegram dapat juga diaplikasikan dalam monitoring infrastruktur jaringan yang mengembangkan aplikasi multikanal [8]. *Bot* telegram dapat terintegrasi pada server openNMS dan perangkat router mikrotik sehingga *system administrator* dapat melakukan proses monitoring pada server, perangkat jaringan, dan layanan. *Bot* ini dikembangkan dengan bahasa pemrograman NodeJS.

Metode Penelitian

Tahap perancangan sistem *chatbot* ini merupakan tahap desain yang dapat menjadi solusi permasalahan dari tahap analisis. Desain di sini meliputi desain sistem dan desain perangkat lunak. Penelitian ini memanfaatkan fitur Bot dari Whatsapp yang digunakan untuk merespon pesan. Arsitektur dari Whatsapp bot sebagai berikut.



Gambar 1. Arsitektur Sistem

Sysadmin mengirimkan perintah melalui aplikasi whatsapp yang sudah terinstall. Pesan diterima di Whatspp Server dan diteruskan ke Whatsapp Web Server untuk kemudian pertanyaan yang diajukan akan diproses oleh Whatsapp Web Server untuk memperoleh data/jawaban akan dikirimkan kembali ke Whatsapp Server dan diteruskan ke Aplikasi Whatsapp. Cara berkomunikasi dari whatsapp server ke server aplikasi *bot* dengan menggunakan teknologi *headless chrome*. Google chrome diinstall di server untuk menjalankan whatsapp web yang digunakan sebagai whatsapp bot. Sederhananya adalah menjalankan whatsapp dengan google chrome tanpa tampilan, atau melalui command line. Dalam menjalankan browser untuk mengakses whatsapp web digunakan library venom. Library venom juga berfungsi untuk menangani sejumlah aktivitas yang berlangsung pada whatsapp web, seperti mendapatkan pesan baru dan mengirim pesan.

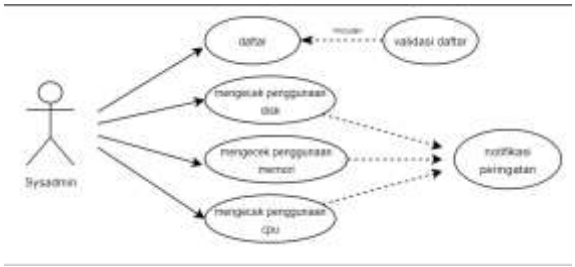
Dalam *chatbot* yang dikembangkan merepresentasikan data dalam *Vector Space Model* yang digunakan secara luas dalam berbagai model

klasifikasi teks [13]. Dataset kami direpresentasikan sebagai satu set pasangan tag kelas string seperti

$$Dataset = \{(S1, C1), (S2, C2), \dots, (Sq, Cq)\}$$

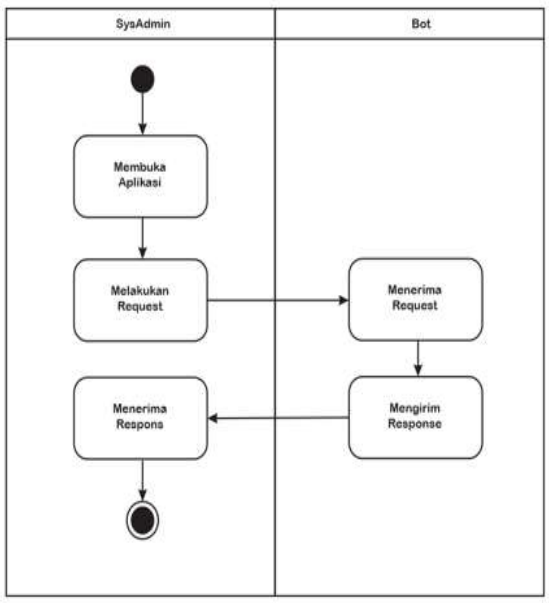
Di sini C1 milik *tag* yang telah ditentukan yang mewakili topik kalimat. Setiap contoh kalimat Si direpresentasikan sebagai vektor istilah di mana setiap indeks mewakili satu istilah/kata unik, Si = {kata 1 , kata 2 ... , kata n }. Metode *linier contextual* [14] pada *chatbot* menganggap konteks sebagai daftar linier dan hanya menyimpan konteks percakapan terakhir. Karena jaringan saraf memprediksi jawaban dan mempertimbangkan lebih dari satu *tag*; *tag* kemungkinan tertinggi yang memiliki ketergantungan konteks yang relevan mendapat prioritas di sini.

Untuk mengembangkan aplikasi ini, penulis menggunakan *use case diagram* dibawah ini menjelaskan urutan proses dari aktor dan sistem untuk mencapai suatu tujuan dapat dilihat pada Gambar 2.



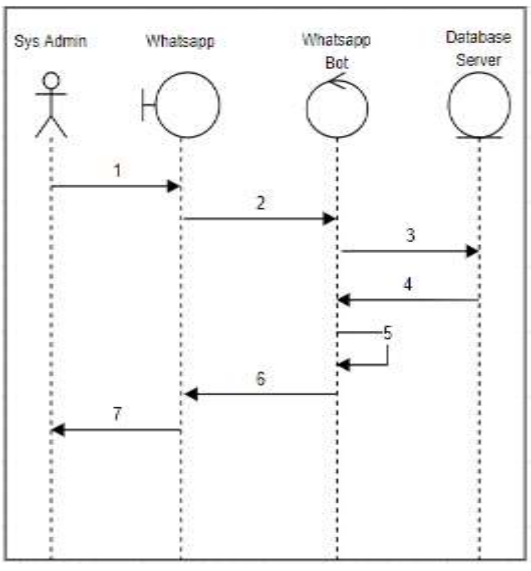
Gambar 2. *Use Case Diagram*

Gambar 2 menampilkan kegiatan sysadmin yang dapat melihat informasi yang ada di dalam sistem server setelah berhasil mendaftar di *bot* whatsapp. *Activity diagram* berikut menggambarkan bagaimana *sysadmin* melakukan *request* terhadap informasi data server melalui *bot* Whatsapp.



Gambar 3. Diagram Activity Bot

Dari gambar di atas dapat simpulkan, aplikasi akan mengirimkan informasi (*response*) berdasarkan apa yang telah *user request*. *Sequence Diagram* menjelaskan secara detail tentang urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*, interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antara operasi, dan informasi yang diperlukan oleh masing-masing operasi (Suhendar dan Gunadi). Di dalam *Sequence Diagram*, terdapat pelaku (aktor), *boundary class*, *control class*, dan *entity class*. Boundary Class merupakan class yang memodelkan interaksi antara satu atau lebih aktor dengan sistem. *Boundary Class* memodelkan bagian dari sistem yang bergantung pada pihak lain disekitarnya dan merupakan pembatas sistem dengan dunia luar. *Control Class* digunakan untuk memodelkan "perilaku mengatur", khusus untuk satu atau beberapa *use case* saja. *Entity Class* memodelkan informasi yang harus disimpan oleh sistem. *Entity Class* memperlihatkan struktur data dari suatu sistem. *Sequence diagram* menjelaskan cara kerja dari aplikasi *chatbot* yang dikembangkan. menunjukkan proses dimana sysadmin saat mengirimkan perintah melalui whatsapp. Dalam proses (1) Sys admin mengirimkan sebuah perintah, kemudian diterukan ke proses (2) dimana pesan dari *sysadmin* melalui whatsapp akan diteruskan ke *whatsapp bot*. Pada proses (3) *Whatsapp bot* akan melakukan pengambilan data dari database untuk mengambil data user. Kemudian pada proses (4) *Database* meneruskan data yang telah diminta, dan melakukan proses (5) verifikasi data user dan data yang diminta, dan selanjutnya pada proses (6) *Whatsapp bot* akan mengirim respon hasil data sesuai yang diminta ke whatsapp. Dalam proses (7) whatsapp menampilkan informasi ke *sysadmin*.



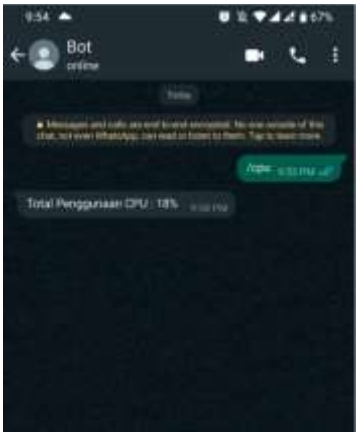
Gambar 4. Sequence Mengirim Perintah

Hasil dan Pembahasan

Setelah dilakukan uji coba terhadap aplikasi pada penelitian ini dapat disimpulkan bahwa aplikasi ini mampu berjalan dengan baik pada kondisi jaringan yang stabil, ini dikarenakan data yang diambil dan ditampilkan adalah data secara *realtime*. Jadi setiap *sysadmin* mengakses melalui *chatbot* whatsapp maka akan secara otomatis melakukan *request* langsung ke server, sehingga *chatbot* whatsapp akan menampilkan data informasi yang telah diolah saat itu juga. Kelebihan *monitoring server* menggunakan bot whatsapp sendiri yaitu memudahkan system administrator memperoleh informasi sumber daya server secara real-time. *Sysadmin* dapat langsung mengakses informasi server. Untuk mengetahui keadaan server secara realtime melalui *chatbot* whatsapp setelah melakukan registrasi terlebih dahulu. Memberikan notifikasi secara akurat kepada sysadmin yang terjadwal rapi sesuai dengan sistem scheduler yang sudah diatur. Penilaian pada uji coba ini menggunakan informasi valid atau tidak valid, banyak masukan yang ditampung untuk sedikit demi sedikit memperbaiki aplikasi yang dibangun. Untuk implementasi dari aplikasi *chatbot* dimulai dengan proses registrasi seperti dijabarkan pada gambar 5.

Gambar 5. Registrasi chat

Proses registrasi digunakan untuk mendata nomor administrator yang mengelola server yang akan dimonitoring. Setelah proses registrasi sysadmin dapat melakukan check server dengan masukan perintah yang telah ditentukan. Misalkan sysadmin melakukan proses monitoring untuk aktivitas CPU dengan memberikan masukan perintah /cpu, sehingga hasil yang diharapkan dapat ditampilkan seperti gambar 6.

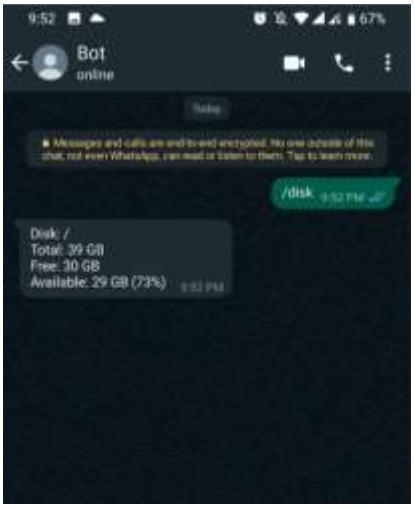


Gambar 6. Mengecek Penggunaan CPU

Ujicoba proses selanjutnya adalah dengan memberikan masukan untuk mengetahui informasi memori dengan perintah /memori. Informasi yang dihasilkan dideskripsikan pada gambar 7, dimana total memori adalah 3072MB dengan penggunaan memori 3020MB dan ketersedian memori tersisa 2%.

Gambar 7. Mengecek Penggunaan Memori

Ujicoba selanjutnya adalah memberikan masukan kepada *chatbot* untuk melihat respon dari hasil input /disk untuk melakukan monitoring terhadap ketersediaan storage pada server. Dari hasil pada gambar 8, menunjukkan sistem *chatbot* dapat memberikan informasi berupa *total storage* dan jumlah *storage* yang telah digunakan. Sajian data dalam bentuk ukuran MB dan persentase. Secara umum, *chatbot* sudah dapat memberikan respons sesuai dengan kondisi ketersediaan *storage* pada server.



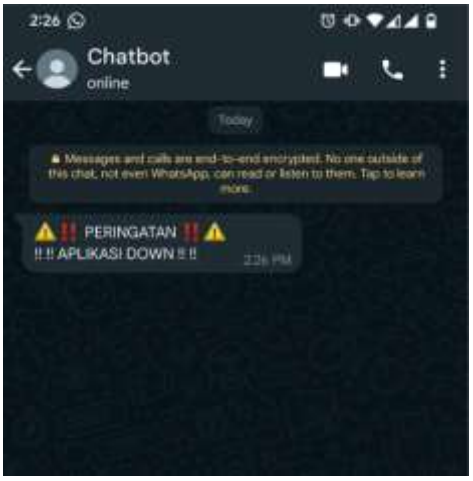
Gambar 8. Ujicoba penggunaan *storage*

Untuk ujicoba selanjutnya, *chatbot* akan melakukan skenario notifikasi saat berada dalam kondisi tertentu, misalkan kejadian saat CPU, memori, *storage* melebihi kapasitas atau berada dalam kondisi yang perlu dicermati dengan kondisi penggunaan diatas 70%. Notifikasi ini melakukan pengecekan di server setiap 5 menit, saat kondisi penggunaan CPU server melebihi batas pemakaian sesuai batasan penggunaan yang sudah diatur sebelumnya, *sysadmin* tidak perlu memberikan masukan perintah secara berulang untuk monitoring. Proses ini divisualisasikan pada gambar 9.

Gambar 9. Notifikasi peringatan penggunaan CPU

Pada ujicoba selanjutnya, penulis melakukan model skenario dengan kondisi aplikasi yang berjalan pada server berhenti atau down. Kemudian melihat respons dari *chatbot*, untuk mengetahui *chatbot* dapat memberikan informasi tentang kondisi yang terjadi saat aplikasi di server down. Hasil ujicoba ini, menunjukkan bahwa aplikasi *chatbot* dapat memberikan respon saat kondisi aplikasi pada server tidak berjalan, hasil ini ditampilkan pada gambar 10.

Hasil tahapan pengujian diatas telah dideskripsikan pada tabel 1 dengan melakukan pengujian berbasis metode *blackbox* dimana fungsi yang diujikan adalah proses pengecekan pada CPU, Memori, *Disk Storage*, pengujian proses registrasi, dan pengujian *chatbot* dalam menangani kondisi saat CPU, memori, disk melebihi batas maksimal yang ditetapkan.



Gambar 10. Peringatan Aplikasi Down

Dari hasil pengujian *blackbox* pada tabel 1 dideskripsikan kondisi yang terjadi, *output* yang diharapkan, dan *output* yang dihasilkan. Pada hasil uji proses *chatbot* menunjukkan status bahwa aplikasi *chatbot* dapat berjalan sesuai harapan dan dapat dijadikan validasi untuk pengujian fungsi *chatbot*.

Tabel 1. Hasil Pengujian Blackbox

No	Fungsi yang di uji	Kondisi	Output	Status
1	Request monitoring CPU	Menampilkan informasi penggunaan CPU	Total Penggunaan CPU: 80%	Valid
		Menampilkan informasi penggunaan CPU mencapai 60%	Total Penggunaan CPU: 60%	Valid
		Belum mendaftar dan Total penggunaan CPU mencapai 80%	Maaf, nomor anda belum terdaftar	Valid
2	Request monitoring Memory	Sudah mendaftar dan Memori tersedia: 75MB	Total memory: 3072MB Memory digunakan: 2997MB Memory tersedia: 75MB (2%)	Valid
		Belum mendaftar dan Memori tersedia: 75MB	Maaf, nomor anda belum terdaftar	Valid
3	Request kondisi Disk	Sudah mendaftar dan Disk tersedia: 35GB	Disk: / Total: 39GB Free: 36GB Available: 33GB (85%)	Valid
		Belum mendaftar dan Disk tersedia:	Maaf, nomor anda belum	Valid

		35GB	terdaftar	
4	Kirim peringatan CPU	Penggunaan CPU melebihi batas maksimal	CPU digunakan mencapai 95%	Valid
		Penggunaan CPU tidak melebihi batas maksimal	-	Valid
5	Kirim peringatan Memory	Sisa memory kurang dari batas maksima	Memory digunakan mencapai 97%	Valid
		Sisa memory tidak kurang dari batas maksimal	-	Valid
6	Kirim peringatan Disk	Disk yang tersedia kurang dari batas maksimal	Kapasitas Disk tinggal 9%	Valid
		Disk yang tersedia tidak kurang dari batas maksimal	-	Valid
7	registrasi	Belum melakukan registrasi	Nomor anda berhasil didaftarkan	Valid
		Telah melakukan registrasi	Nomor anda sudah terdaftar	Valid

Kesimpulan dan Saran

Aplikasi *chatbot* yang dikembangkan dapat digunakan pengguna melalui aplikasi Whatsapp. Proses *request* yang menjadi masukan pada aplikasi *chatbot* untuk memonitor sumber daya dapat berjalan dengan baik dan tervalidasi dengan pengujian. Hasil pengujian aplikasi *chatbot* yang berjalan diatas whatsapp dapat diimplementasikan sesuai dengan perencanaan rancangan awal dari aplikasi ini yaitu aplikasi dapat menampilkan informasi mengenai sumber daya pada server seperti jumlah penggunaan penyimpanan *hardisk*, *memory*, dan CPU. Hasil pengujian juga menunjukkan bahwa aplikasi dapat memberikan respon peringatan kepada pengguna atau *system administrator* saat sumber daya server melebihi kapasitas atau saat layanan tidak berjalan.

Pengembangan penelitian ini kedepannya dapat dengan mengembangkan fitur aplikasi tambahan berupa informasi sumber daya yang ditampilkan lebih lengkap pada server, menujukkan informasi visualisasi *chart*, dan terintegrasi dengan sistem manajemen server.

Daftar Pustaka

[1] L. ROHMAWATI, "PERANCANGAN DAN IMPLEMENTASI WHATSAPP BOT UNTUK MONITORING SERVER," skripsi, Universitas Teknologi Digital Indonesia, 2022. doi: 10/10_165410174_LAMPIRAN.pdf.

[2] M. Agung, R. Roslina, and R. E. Sari, "Implementasi Aplikasi Pembuatan Chat," *J. Mhs. Fak. Tek. Dan Ilmu Komput.*, vol. 1, no. 1, pp. 293–306, 2020.

[3] I. M. Pustikayasa, "Grup whatsapp sebagai media pembelajaran," *Widya Genitri J. Ilm. Pendidik. Agama Dan Kebud. Hindu*, vol. 10, no. 2, pp. 53–62, 2019.

[4] V. Dihni, "Indonesia Pengguna WhatsApp Terbesar Ketiga di Dunia," *Databoks Katadata*, 2021. https://databoks.katadata.co.id/datapublish/2021/11/23/indonesia-pengguna-whatsapp-terbesar-ketiga-di-dunia (accessed Dec. 30, 2022).

[5] D. Karismata, "Perancangan Bot untuk Remote Monitoring pada Server menggunakan Telegram Bot API," Skripsi, Universitas Kristen Satya Wacana, 2016. [Online]. Available: https://repository.uksw.edu/bitstream/123456789/11449/2/T1_672012109_Full%20text.pdf

[6] F. Muriyanto, "MONITORING SERVER MENGGUNAKAN BOT TELEGRAM DENGAN LOAD BALANCING MICROSERVICE DOCKER," skripsi, STMIK AKAKOM YOGYAKARTA, 2020. Accessed: Dec. 30, 2022. [Online]. Available: https://eprints.utdi.ac.id/9020/

[7] A. Nurhayati, "MONITORING SISTEM KEAMANAN JARINGAN BERBASIS TELEGRAM BOT PADA LOCAL AREA NETWORK," *J. Inform. Commun. Technol. JICT*, vol. 1, no. 2, pp. 45–53, 2019.

[8] R. J. Putra, N. P. Sastra, and D. M. Wiharta, "PENGEMBANGAN KOMUNIKASI MULTIKANAL UNTUK MONITORING INFRASTRUKTUR JARINGAN BERBASIS BOT TELEGRAM," *J. SPEKTRUM*, vol. 5, no. 2, 2018, doi: https://doi.org/10.24843/SPEKTRUM.2018.v05.i02.p19.

[9] P. B. Ramadhanu and A. T. Priandika, "Rancang Bangun Web Service Api Aplikasi Sentralisasi Produk Umkm Pada Uptd Plut Kumkm Provinsi Lampung," *J. Teknol. Dan Sist. Inf.*, vol. 2, no. 1, pp. 59–64, 2021.

[10] E. Wijaya and P. Pebriantara, "Rancangan Bangun Aplikasi Pembelajaran dengan Memanfaatkan *Chatbot* API Dialogflow dan Moodle Berbasis Android Pada SMA IT ALIA Tangerang," *Best Account. Inf. Syst. Inf. Technol. Bus. Enterp.*, vol. 3, no. 2, pp. 328–335, Dec. 2018, doi: 10.34010/aisthebest.v3i2.1522.

[11] M. A. Nugroho, A. Damayanti, M. F. Rifai, and S. Windarti, "PENGEMBANGAN APLIKASI QnA UNTUK PENDAFTARAN MAHASISWA BARU STMIK AKAKOM," *J. Inf. Syst. Manag. JOISM*, vol. 3, no. 1, pp. 18–23, Jan. 2021, doi: 10.24076/JOISM.2021v3i1.408.

[12] A. Fathurrozi and F. Karimah, "PELAYANAN DAN INFORMASI CUSTOMER SERVICE BERBASIS BOT TELEGRAM DENGAN ALGORITMA FORWARD CHAINING PADA CV.PRIMGUARD INDONESIA," *J. Inform. Inf. Secur.*, vol. 2, no. 2, Art. no. 2, Dec. 2021, doi: 10.31599/jiforty.v2i2.884.

[13] R. Suryana, M. Aryanto, R. Kurniawan, K. S. G. P. Satmata, Y. Yulianti, and A. Saifudin, "Pengembangan Kecerdasan Buatan Whatsapp *Chatbot* untuk Mahasiswa," *J. Teknol. Sist. Inf. Dan Apl.*, vol. 5, no. 1, Art. no. 1, Jan. 2022, doi: 10.32493/jtsi.v5i1.15487.

[14] M. Strutynskiy, "A concept of an intent-based contextual chat-bot with capabilities for continual learning." 2020.