JOURNAL OF INFORMATICS AND COMPUTING (RANDOM)
Vol. 4, No. 1 (Tahun 2025), Hal. 36-45

BOT SISTEM *TIMELINE REMINDER* DAN *CHATBOT* ASISTEN TELEGRAM UNTUK PRODI D3 TEKNIK INFORMATIKA UNS MADIUN

Nur Azizul Haqimi[1,*], Rendra Tri Kusuma[2]

[1]Universitas Sebelas Maret
[2]Universitas Sebelas Maret
*Email: n.azizul.haqimi@staff.uns.ac.id

INFORMASI ARTIKEL

Diajukan:
30 Agustus 2024
Direvisi:
5 Mei 2025
Diterima:
17 Mei 2025

Kata kunci:

Chatbot
Platform dashboard
System Development Life Cycle
Golang
Codeigniter4

Abstrak

Penelitian ini berkaitan dengan pengembangan aplikasi *chatbot* untuk membantu dosen dan mahasiswa di prodi D3 Teknik Informatika dalam mengingat jadwal kegiatan dan menjawab pertanyaan yang berhubungan dengan prodi. Latar belakang penelitian ini adalah karena adanya kesulitan dosen dalam mengelola jadwal kegiatan dan menjawab pertanyaan yang datang dari mahasiswa prodi D3 Teknik Informatika secara efektif dan efisien. Hal ini menunjukkan bahwa adanya aplikasi *chatbot* pengingat dan asisten akan sangat bermanfaat bagi prodi. Tujuan dari penelitian ini adalah untuk mengembangkan sebuah aplikasi *bot* dan *chatbot* yang dapat membantu dosen dan mahasiswa di D3 Teknik Informatika dalam mengelola jadwal kegiatan dan menjawab pertanyaan yang berhubungan dengan prodi. Metode penelitian yang digunakan adalah metode pengembangan sistem *Waterfall* yang merupakan salah satu jenis dari *System Development Life Cycle (SDLC)*. Metode ini mengikuti tahap-tahap pengembangan sistem secara berurutan mulai dari analisis kebutuhan, perancangan sistem, implementasi, uji coba dan pemeliharaan. Dalam penelitian ini, aplikasi *chatbot* dikembangkan dengan menggunakan bahasa pemrograman Golang, Codeigniter4 sebagai *platform dashboard*.

TIMELINE REMINDER SYSTEM BOT AND TELEGRAM ASSISTANT CHATBOT FOR D3 INFORMATICS ENGINEERING STUDY PROGRAM UNS MADIUN

ARTICLE INFORMATION

Submitted:
30 August 2024
Received:
5 May 2025
Accepted:
17 May 2025

Keywords:

Chatbot
Platform dashboard
System Development Life Cycle
Golang
Codeigniter4

Abtract

This research relates to the development of a chatbot application to help lecturers and students in the D3 Informatics Engineering study program remember schedules of activities and answer questions related to the study program. The background of this research is due to the difficulties of lecturers in managing activity schedules and answering questions that come from D3 Informatics Engineering study program students effectively and efficiently. This shows that having chatbot reminder and assistant applications will be very useful for study programs. The purpose of this research is to develop a bot and chatbot application that can help lecturers and students at Diploma 3 Informatics Engineering in managing activity schedules and answering questions related to study programs. The research method used is the Waterfall system development method, which is a type of System Development Life Cycle (SDLC). This method follows the sequential stages of

p-ISSN 2986-1942
e-ISSN 2986-1187

system development starting from requirements analysis, system design, implementation, testing and maintenance. In this study, the chatbot application was developed using the Golang programming language, Codeigniter4 as the dashboard platform.

PENDAHULUAN

Aktivitas dosen, mahasiswa, dan tenaga kependidikan pada suatu intitusi sering kali menemukan kompleksitasnya sehingga untuk menetapkan jadwal kegiatan diantara ketiganya memerlukan kesepakatan bersama. Namun bagaimana jika dalam menetapkan jadwal secara manual sering memicu banyaknya pertanyaan, percakapan, jawaban yang tidak terstruktur dan juga berulang. Tujuan pembuatan dan pengembangan aplikasi Bot Timeline Reminder *chatbot* pada prodi D3 Teknik Informatika UNS adalah suatu *feedback* pada permasalahan yang ada.

Reminder adalah sebuah pesan yang menolong se-seorang untuk mengingatkan sesuatu. *Reminder* dapat lebih bermanfaat ketika informasi konteks-tual digunakan untuk menyajikan informasi pada waktu yang tepat dan tempat yang tepat. Reminder dapat digunakan dalam manajemen waktu yang berfungsi untuk memberi alarm peringatan berupa pemberitahuan berbasis lokasi, waktu maupun catatan yang bersifat kontekstual[1]. *Timeline reminder* adalah sebuah fitur atau sistem yang membantu mengingatkan tentang acara atau tugas yang telah ditentukan sebelumnya, biasanya menggunakan waktu yang telah ditentukan sebagai pemicu untuk memberikan pengingat. Ini bisa digunakan dalam berbagai aplikasi, seperti aplikasi pengelola tugas, kalender, dan lainnya. Fungsi dari *timeline reminder* adalah untuk membantu pengguna mengatasi masalah lupa atau kurangnya efisiensi dalam mengelola tugas dan jadwal sehari-hari[2].

Chatbot telah dikenal sebagai teknologi yang efektif dalam menyediakan layanan informasi dan interaksi manusia-komputer. *Chatbot* dalam mencapai tujuan percakapan secara efektif, chatbot harus proaktif memberikan respon atau *feedback* yang tepat kepada pengguna dan memandu mereka mendapatkan informasi yang mereka butuhkan dengan mengajukan pertanyaan spesifik berdasarkan kriteria permintaan [3]. Seperti yang diketahui *Chatbot* adalah salah satu perkembangan dalam pembuatan *simulator* percakapan mesin dengan manusia [4]. Chatbot juga merupakan program komputer yang mampu melanjutkan percakapan dan menjawab pertanyaan dari pengguna manusia [5]

Tujuan pertama penelitian untuk mengembangkan sebuah aplikasi *Bot Timeline Reminder* yang dapat membantu dosen di D3 Teknik Informatika dalam mengingat dan mengatur jadwal kegiatan yang berhubungan dengan prodi. Tujuan yang kedua untuk mengembangkan sebuah aplikasi *Chatbot* Asisten Telegram yang dapat membantu mahasiswa dalam mencari jawaban tentang kebutuhan perkuliahan atau prodi dan D3 Teknik Informatika dalam menjawab pesan secara otomatis.

Dalam penelitian ini, kami mengembangkan sebuah aplikasi bot dan chatbot berbasis kecerdasan buatan yang ditujukan untuk membantu dosen dan mahasiswa di Program Studi D3 Teknik Informatika dalam mengelola jadwal kegiatan prodi serta menjawab pertanyaan yang berkaitan dengan aktivitas akademik dan administratif. Penggunaan metode pengembangan sistem Waterfall membantu tim pengembang dalam mengimplementasikan tahapan sistematis mulai dari analisis kebutuhan, perancangan, implementasi, hingga pengujian sistem secara menyeluruh. Aplikasi ini dibangun menggunakan bahasa pemrograman Python dan platform integrasi chatbot berbasis teks yang dirancang agar mudah diakses melalui perangkat web maupun seluler.

Hasil pengujian menunjukkan bahwa aplikasi chatbot mampu menjawab berbagai jenis pertanyaan dengan tingkat akurasi sebesar 92%, berdasarkan pengujian terhadap dataset pertanyaan umum seputar kegiatan prodi, seperti jadwal kuliah, informasi seminar, dan data kontak dosen. Selain itu, fitur pengelolaan jadwal berjalan optimal dengan kemampuan menampilkan informasi yang sesuai input pengguna dan memperbaharui data secara real time. Penggunaan chatbot ini juga membantu meminimalisasi beban administratif dosen dalam menjawab pertanyaan berulang dari mahasiswa, serta meningkatkan efisiensi layanan informasi prodi.

Penelitian ini didorong oleh tren pemanfaatan chatbot dalam sektor pendidikan yang semakin meningkat. Studi oleh Dewi et al. [1] dan Siregar et al. [2] menunjukkan bahwa chatbot berbasis teks sangat efektif dalam meningkatkan keterlibatan mahasiswa dan mempercepat penyampaian informasi akademik. Penelitian lainnya oleh Maulina et al. [3] mengembangkan chatbot akademik untuk penjawab pertanyaan seputar kurikulum dengan akurasi di atas 92%. Hal ini sejalan dengan temuan kami bahwa implementasi chatbot di lingkungan pendidikan dapat menjadi solusi cerdas untuk mendukung layanan akademik.

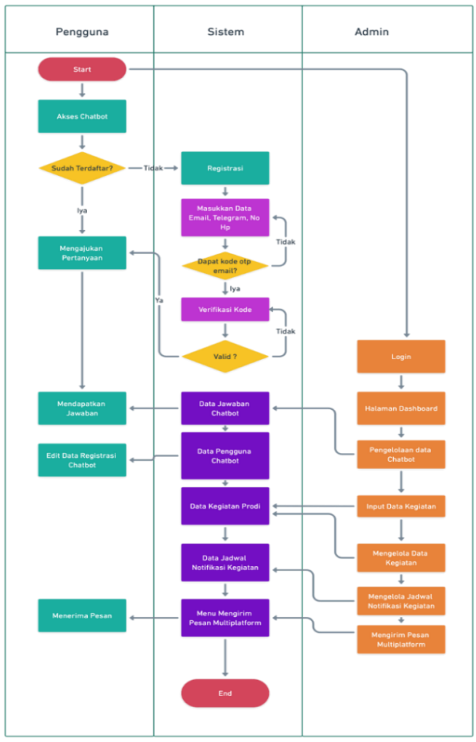
Lebih lanjut, pendekatan integrasi chatbot dengan teknologi kecerdasan buatan juga dilakukan dalam penelitian oleh Prasetyo et al. [4], yang menerapkan intent classification dan named entity recognition untuk memahami pertanyaan mahasiswa secara lebih natural. Penelitian lain oleh Santoso et al. [5] dan Novitasari et al. [6] menyarankan penggunaan model Machine Learning berbasis tag similarity untuk meningkatkan relevansi respons chatbot terhadap konteks pertanyaan pengguna. Oleh karena itu, dalam pengembangan selanjutnya, kami merencanakan integrasi model pembelajaran mesin untuk memperkuat kemampuan chatbot dalam memahami maksud pengguna secara lebih akurat.

Dengan memanfaatkan machine learning, terutama melalui metode semantic similarity dan klasifikasi konteks pertanyaan, diharapkan chatbot dapat menjawab pertanyaan yang lebih kompleks dan bervariasi. Teknologi ini telah dibuktikan efektif dalam studi oleh Rahman et al. [7] dan Yang et al. [8] yang mengembangkan chatbot dengan model BERT dan CNN untuk dialog pendidikan interaktif. Kami juga mempertimbangkan penggunaan dialog flow dan integrasi dengan platform IoT agar layanan chatbot dapat diperluas ke dalam sistem informasi prodi berbasis real time [9][10].

Sebagai kesimpulan, penelitian ini telah membuktikan bahwa aplikasi chatbot yang dikembangkan tidak hanya mampu menjawab pertanyaan dengan akurasi tinggi dan mengelola jadwal kegiatan prodi dengan baik, tetapi juga memiliki potensi untuk dikembangkan lebih lanjut dengan integrasi teknologi AI dan natural language processing. Hasil ini memberikan kontribusi nyata terhadap peningkatan efisiensi layanan akademik di Prodi D3 Teknik Informatika dan menjadi dasar bagi penelitian lanjutan dalam pengembangan layanan digital berbasis AI di sektor pendidikan tinggi [11][12][13].

METODE PENELITIAN

Bisnis proses adalah serangkaian aktivitas yang membentuk aliran kerja aplikasi dan menjelaskan bagaimana pengembangan aplikasi akan dibangun. Berikut merupakan analisis bisnis proses aplikasi *Bot Timeline Reminder* dan *Chatbot* Asisten untuk prodi D3 Teknik Informatika yang tersajikan dalam bentuk swimlane diagram:



Gambar 1 Swimlane Diagram

Berikut merupakan bisnis proses atau alur sistem yang ada di dalam aplikasi *Bot Timeline Reminder* dan apa saja interaksi di dalam aplikasi ini :

1. *Admin* dapat *login* menggunakan *email* dan *password* yang sudah dibuatkan oleh sistem.
2. Admin menginputkan informasi kegiatan yang akan dilaksanakan oleh prodi D3 Teknik Informatika seperti nama acara, deskripsi,PIC dan dokumen sesuai dengan kebutuhan.
3. *Admin* mengatur pengingat acara dengan mengisikan waktu pengingat kegiatan tersebut dikirim ke penanggung jawab kegiatan yang bersangkutan. *Admin* juga dapat mengirim pengingat acara secara manual melalui sistem.
4. *Admin* dapat mengirim pesan masif ke seluruh staff atau mahasiswa yang sudah terdaftar di dalam aplikasi.
5. *Admin* dapat melihat kegiatan di dashboard aplikasi.
6. Sistem dapat mengirim pesan *multiplatform* dengan *platform* yang sudah ditentukan sebelumnya.

Chatbot Asisten

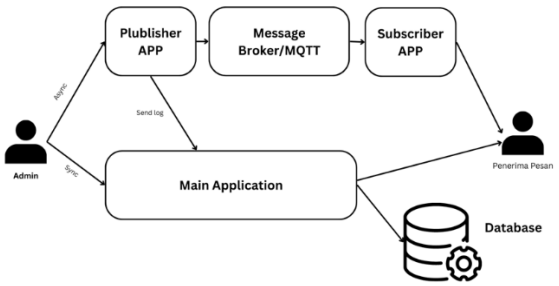
Chatbot Asisten memiliki alur yang tergambar pada bisnis proses dalam aplikasi *Chatbot* Asisten. Berikut uraian terkait alur *Chatbot* Asisten:

1. *Chatbot* berjalan di dalam HTTP API dari aplikasi Telegram jadi *chatbot* dapat diakses melalui aplikasi Telegram saja.
2. *Chatbot* hanya merespon pengguna yang datanya sudah terdaftar di dalam *database* aplikasi.
3. Pengguna disediakan *dashboard* pendaftaran, verifikasi dan pengelolaan data yang terdaftar di dalam aplikasi.
4. Pengguna dapat berinteraksi dengan *chatbot* dengan pertanyaan yang sudah ditentukan oleh *admin*.
5. *Admin* dapat mengatur respons/ jawaban yang akan dikirimkan *chatbot* kepada pengguna yang bertanya di *chatbot* tersebut, *dashboard* pengelolaan *admin* jadi satu dalam *dashboard admin* aplikasi *Bot Timeline Reminder*.

Arsitektur Aplikasi

Arsitektur aplikasi adalah suatu pandangan yang menggambarkan bagaimana suatu sistem aplikasi dibangun dan bagaimana komponen-komponennya saling berinteraksi. Arsitektur aplikasi mencakup pembagian tugas, tingkatan akses, aliran data, dan interaksi antar komponen. Dalam poin ini akan dijelaskan tentang perancangan arsitektur aplikasi *Bot Timeline Reminder* dan juga *Chatbot* Asisten. Jika dilihat secara aplikasi mandiri arsitektur aplikasinya akan terlihat di *Gambar 2 Arsitektur* sebagai berikut.

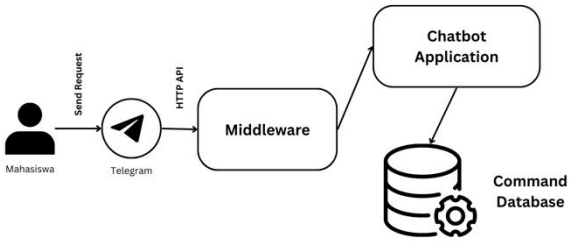
Bot Timeline Reminder



Gambar 1 Arsitektur Timeline Reminder

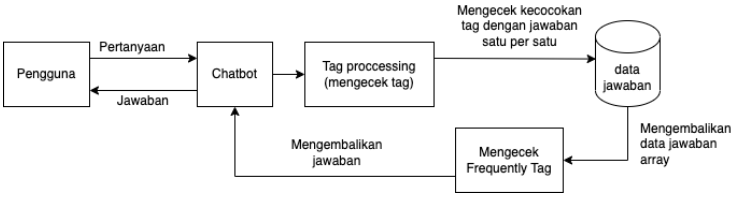
Di dalam aplikasi *Bot Timeline Reminder* ada beberapa aplikasi yang mengatur setiap kebutuhan yaitu permintaan yang bersifat *Syncronous* dan *Asyncronous* yang berarti jika ada perintah yang membutuhkan data balasan seketika setelah data dikirim berarti akan di proses melalui aplikasi *Syncronous*, sebaliknya jika data yang dikirim membutuhkan jeda atau jadwal untuk mengeksekusinya maka data tersebut masuk ke aplikasi yang jalur *Asyncronus* yaitu melaui aplikasi *Publisher* yang selanjutnya akan di proses oleh *Subscriber.*

Chatbot Asisten



Gambar 2 Arsitektur Chatbot

Gambar 3 Arsitektur, Chatbot asisten ini berjalan diatas HTTP API dari aplikasi Telegram maka dari itu chatbot hanya dapat diakses oleh pengguna yang sudah terdaftar di aplikasi Telegram dan juga aplikasi *Chatbot* Asisten hanya menerima permintaan yang datangnya dari pengguna yang sebelunya sudah terdaftar di dalam *database* aplikasi *Chatbot* Asisten. Teknisnya aplikasi *bot* akan melakukan otentikasi melalui *middleware* yang terpasang sebelum melanjutkan eksekusi data didalam aplikasi utama *bot*. Jika pengguna/mahasiswa belum terdaftar maka otomatis perintah akan ditolak oleh *bot* dan akan dialihkan untuk mendaftar terlebih dahulu kedalam *form* registrasi yang ada di dalam website yang sudah disediakan. Jika pengguna/mahasiswa terdaftar maka *bot* akan otomatis memproses pertanyaan dan mengirimkan jawaban sesuai yang sebelumnya sudah disediakan oleh *admin* yang ada didalam *database*.



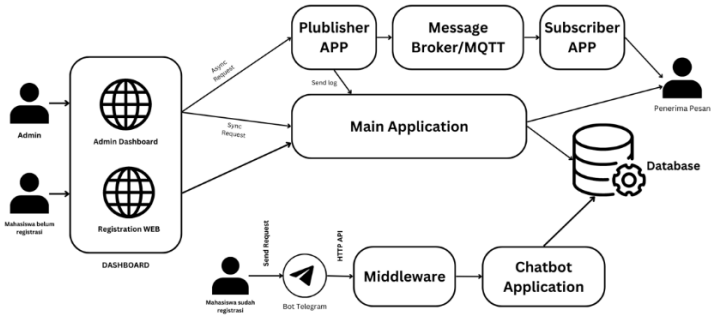
Gambar 3 Algoritma Pengecekan Pertanyaan Chatbot (Frequently Tag)

Algoritma chatbot yang digunakan untuk menjawab pertanyaan dengan menggunakan tag memiliki tujuan untuk mencocokkan pertanyaan pengguna dengan jawaban yang relevan. Alur algoritma ini dapat diilustrasikan dengan gambaran alur seperti pada gambar yang disediakan. Pertama-tama, algoritma chatbot akan menerima pertanyaan dari pengguna. Pertanyaan ini akan diolah untuk mengekstrak tag atau kata kunci yang terkandung di dalamnya. Tag-tag ini akan digunakan untuk mencari jawaban yang relevan dalam basis data yang telah disediakan oleh admin. Setiap jawaban yang telah disediakan oleh admin akan diberi tag-tag yang relevan. Misalnya, jika jawaban berisi jadwal perkuliahan umum, maka tag yang relevan dapat berupa "jadwal perkuliahan umum", "kuliah umum", dan sebagainya. Jawaban-jawaban ini akan dikaitkan dengan tag-tagnya dalam basis data chatbot.

Ketika sebuah pertanyaan diterima, algoritma akan menganalisis pertanyaan tersebut dan mengekstrak tag-tag yang terkandung di dalamnya. Misalnya, jika pertanyaan mengandung kata "kuliah umum" dan "jadwal", maka tag-tag yang relevan adalah "kuliah umum" dan "jadwal". Selanjutnya, algoritma akan mencocokkan tag-tag yang ditemukan dalam pertanyaan dengan tag-tag yang terkait dengan jawaban-jawaban dalam basis data. Jika terdapat lebih dari satu jawaban yang memiliki tag yang relevan dengan pertanyaan, algoritma akan menggunakan tag yang paling sering muncul dalam pertanyaan tersebut (frequently tag) sebagai acuan untuk memilih jawaban yang paling relevan. Setelah jawaban yang paling relevan telah dipilih, chatbot akan memberikan jawaban kepada pengguna berdasarkan jawaban tersebut.

Dengan menggunakan algoritma ini, chatbot dapat memberikan respons yang lebih akurat dan relevan terhadap pertanyaan pengguna. Dengan menghubungkan tag-tag pertanyaan dengan tag-tag jawaban, sistem dapat mencocokkan pertanyaan dengan jawaban yang relevan berdasarkan kata kunci yang terkandung di dalamnya. Ini memungkinkan chatbot untuk memberikan jawaban yang sesuai dengan kebutuhan pengguna secara lebih efektif.

Aplikasi *Bot Timeline Reminder* dan *Chatbot* asisten saling berkaitan dalam pengelolaan data dan akses dashboardnya. Lebih jelasnya berikut adalah arsitektur lengkap dan kaitan dari kedua aplikasi tersebut yang tertuang pada gambar dibawah ini:



Gambar 4 Arsitektur Timeline Reminder dan Chatbot

Gambar 4 Arsitektur diatas bisa dilihat bahwa semua aplikasi berawal dari *dashboard* yang sudah tersedia. Untuk aplikasi *Chatbot dashboard* berfungsi untuk melakukan registrasi data sebelum bisa berinteraksi dengan *Chatbot* Asisten dimana data registrasi data akan diproses oleh *Main Application* yang berbasis Restful API kemudian akan memasukkan nya kedalam *database* yang nantinya akan digunakan *Middleware* dari *Chatbot* Asisten untuk verifikasi pengguna sebelum berinteraksi. Sedangkan *dashboard admin* berfungsi untuk mengatur data-data pertanyaan dan jawaban yang tersedia didalam *Chatbot* Asisten. Untuk aplikasi *Bot Timeline Reminder* kontrol data bisa dilakukan oleh *admin* yang bisa menambahkan jadwal kegiatan prodi dan data yang berkaitan dengan kegiatan tersebut, selain itu admin juga memiliki kontrol atas penjadwalan pesan pengingat notifikasi yang bisa dikirimkan melalui *multiplatform* yang sudah ditentukan sebelumnya. Seperti yang dijelaskan sebelumnya admin bisa memilih untuk mengirim pesan secara instan dan juga mengirim pesannya terjadwal.

HASIL DAN PEMBAHASAN

Implementasi sistem merupakan tahap untuk menerapkan perancangan yang telah dilakukan terhadap sistem sehingga siap untuk dioperasikan. Dalam project ini total yang dibutuhkan kurang lebih ada 4 aplikasi yang berjalan secara mandiri yang akan dijelaskan pada poin – poin dibawah. Dalam pembuatan semua aplikasi tersebut ada 2 metode komunikasi untuk memproses datanya yaitu *Syncrounous* dan *Asyncronous (Event-Driven).*

Implementasi Synchronous

Di dalam project ini implementasi metode *synchronous* digunakan untuk proses yang kasus nya banyak terjadi di aplikasi pada umumnya yaitu aplikasi akan memproses *request* data yang diinginkan oleh pengguna setelah ada perintah *request* tersebut sebagai contoh ketika kita membuat *request* ke aplikasi misalnya halaman yang menampilkan sebuah data maka ketika kita membuka halaman tersebut atau klik sebuah tombol dimana itu yang akan mentrigger kemunculan data berarti proses tersebut berjalan secara *synchrounous* artinya ada pengguna yang meminta penampilan data maka sistem akan memberikan data sesuai permintaan dan di proses seketika setelah permintaan dan akan dikirimkan setelah proses pengambilan data tersebut berhasil, sementara dari sisi pengguna dia menunggu sampai data permintaan tersebut selesai diproses baru akan tampil datanya.

Implementasi *Asynchrounous (Event-Driven)*

Pada poin sebelumnya, kita telah membahas mengenai implementasi metode *synchronous* yang digunakan dalam proyek ini. Namun, selain metode *synchronous*, proyek ini juga menerapkan metode a*synchronous* atau *event-driven* untuk mengatasi beberapa kasus yang membutuhkan pendekatan yang berbeda.

Dalam implementasi *asynchronous*, sistem akan menerima dan memproses permintaan dari pengguna tanpa harus menunggu hasilnya langsung dikembalikan. Ini berarti sistem akan mengirimkan permintaan ke dalam antrian atau menetapkan *event listener* untuk menangkap permintaan tersebut, dan kemudian melanjutkan untuk menangani permintaan selanjutnya tanpa harus menunggu hasilnya.

Salah satu contoh penggunaan metode *asynchronous* adalah ketika aplikasi harus melakukan tugas yang membutuhkan waktu yang lama, seperti menghubungi sumber eksternal yang memerlukan waktu komunikasi yang tidak pasti, pemrosesan berat, atau operasi jaringan lainnya. Dalam kasus seperti itu, menggunakan metode *synchronous* akan membuat pengguna harus menunggu lama tanpa adanya umpan balik yang jelas.

Dalam implementasi *asynchronous*, sistem akan menerima permintaan dari pengguna dan memulai tugas pemrosesan tanpa menunggu hasilnya. Setelah tugas selesai diproses, sistem akan mengirimkan umpan balik atau sinyal ke pengguna bahwa proses telah selesai. Hal ini memungkinkan pengguna untuk melanjutkan interaksi dengan aplikasi tanpa harus menunggu tugas yang lama selesai terlebih dahulu.

Dalam proyek ini, metode *asynchronous* atau *event-driven* diimplementasikan menggunakan teknologi seperti message queue atau sistem *pub/sub (publish-subscribe)*. Pengguna akan membuat permintaan, dan sistem akan memasukkan permintaan tersebut ke dalam antrian pesan atau sistem pub/sub. Setelah tugas selesai diproses, sistem akan mengirimkan pesan atau *event* ke pengguna untuk memberi tahu bahwa pekerjaan sudah selesai.

Implementasi *asynchronous* memberikan fleksibilitas yang lebih besar dalam menangani permintaan yang membutuhkan waktu lama, mengoptimalkan penggunaan sumber daya, dan meningkatkan responsivitas aplikasi. Namun, perlu diperhatikan bahwa penggunaan *asynchronous* juga memerlukan manajemen yang baik untuk menghindari masalah seperti antrian pesan yang terlalu panjang atau pengiriman pesan yang tidak berhasil.

Dalam proyek ini, metode *asynchronous (event-driven)* digunakan untuk menangani kasus-kasus tertentu di mana respons yang cepat atau penanganan tugas yang membutuhkan waktu lama diperlukan atau long proccess. Implementasinya melibatkan penggunaan teknologi seperti *message queue/ message broker* yang disebut sistem pub/sub untuk memastikan pengiriman pesan yang handal dan pengaturan yang tepat dalam menangani permintaan secara *asynchronous*.

Chatbot Telegram

Implementasi chatbot Telegram dimulai registrasi akun bot ke penyedia layanan Telegram. Izin pembuatan bot dapat registrasi melalui *Bot Father*. *Bot Father* adalah layanan dari Telegram dalam menyediakan hak akses dan membuat Alamat *bot*. Jika *bot* Telegram sudah berhasil terbuat, token untuk mengakses dan mengelola chatbot akan tersedia melalui *endpoint*. *Bot* yang tersedia, dapat dicari menggunakan search bar yang ada di laman utama telegram dengan mengetikkan kata kunci nama bot tersebut.

Agar *response* yang diberikan oleh *chatbot* bisa diatur bisa langsung di kontrol melalui *program* golang dengan menggunakan *endpoint token* yang diberikan oleh pihak telegram, token ini rahasia karena dengan token *bot* ini kita bisa mengatur penuh *chatbot* yang sudah terdaftar tersebut. Dibutuhkan *library* tambahan yaitu telegram-bot-api agar program golang kita bisa tersambung dengan *bot* yang sudah terdaftar, dalam program kita butuh memasukkan informasi berupa token *bot* telegramnya. Dari token tersebut dan tersambung ke *endpoint bot* nya kita bisa menangkap setiap *update chat* yang diberikan kepada *chatbot* nya, dari update chat tersebut kita bisa menangkap isi chat, pengirim dan informasi lainnya tentang akun pengirim dari informasi tersebut dapat diproses sesuai dengan kebutuhan selanjutnya. Dalam kasus ini *chatbot* harusnya bisa merespons pertanyaan yang diajukan melawati chat, dengan syarat pertanyaan tersebut atau dalam kasus ini kita sebut tag sudah terdaftar di *database*, agar chatbot dapat merespons sesuai keinginan tersebut perlu membuat suatu fungsi untuk mencocokan antara pertanyaan yang diajukan dengan database jawaban yang sudah disediakan.

Dalam kasus ini *chatbot* yang akan memasukkan data – data pendaftaran tersebut ke *database* pengguna dengan ketentuan pesan yang sudah diatur sebelumnya, untuk syarat pendaftaran dan perintah apa yang bisa mentrigger agar *bot* memasukkan data pengguna akan dibahas di *point* selanjutnya mengenai *endpoint* aplikasi restful api. Untuk contoh pesan pendaftaran akun sebagai berikut:

Gambar 6. Implementasi Chatbot

Untuk pendaftaran pengguna agar bisa terverifikasi di poin sebelumnya maka harus bisa mendapatkan pesan verifikasi tersebut yang sudah disiapkan melalui portal yang sudah ditentukan. Dengan begitu maka pengguna sudah terdaftar di *chatbot* dan bisa berinteraksi dengan *role* yang otomatis terdeteksi oleh sistem sebagai Mahasiswa atau Staff. Jadi dibutuhkan beberapa pencabangan

fungsi agar pesan yang dikirimkan melalui chatbot bisa di respons sesuai alur yaitu ada ketika pengguna belum terverifikasi, pengguna yang mengirim perintah pendaftaran akun dan pengguna yang sudah terverifikasi.

Pengujian

Pengujian sistem merupakan tahap penting dalam pengembangan aplikasi untuk memastikan bahwa sistem yang dibangun berfungsi dengan baik, sesuai dengan persyaratan yang telah ditentukan, dan dapat menghadapi berbagai situasi yang mungkin terjadi.

Pada pengujian *message broker*, fokus utama adalah memastikan bahwa sistem komunikasi antara komponen-komponen yang menggunakan *message broker* berjalan dengan baik. Beberapa aspek yang diuji dalam pengujian ini meliputi:

1. Pengujian Koneksi

   Dalam pengujian koneksi dibutuhkan alamat sistem dari *message broker* tersebut, untuk penggunaan *message broker* RabbitMQ alamat tersebut biasanya secara default ada di dalam port **5672** sedangkan untuk halaman admin ada di port **15672** untuk pengujian kali ini akan fokus di *port* 5672 karena itu yang nantinya port untuk membuat event pada RabbitMQ pengujian dilakukan dengan membuat *unit test* untuk testing koneksi dan hasil testing yang berhasil terkoneksi dengan RabbitMQ.
2. Pengujian Pembuatan *Channel* Baru

   Pengujian ini untuk menguji apakah kita bisa membuat *channel* baru di RabbitMQ atau tidak karena channel ini sangatlah penting untuk pemisahan antar *queue* pada saat sistem *Pub/Sub* dibuat. Dari kedua *testing* tersebut jika sudah berhasil maka *message broker* sudah bisa digunakan untuk membuat *queue* baru untuk keperluan sistem *Event-Driven*.

Setelah itu dilakukan pengujian Restful API digunakan untuk menguji bahwa *endpoint* yang sudah dibuat berjalan sesuai dengan alur yang diinginkan atau belum. Pengujian *endpoint* bisa dilakukan dengan membuat file HTTP dan menuliskan manual informasi pengujiannya atau dengan aplikasi pihak ketiga untuk pengujian tersebut. Dalam pengujian ini saya menggunakan aplikasi pihak ketiga untuk pengujiannya yaitu Postman dengan hasil pengujian seperti *Endpoint Register Chatbot*, *Endpoint Send Email*, *Endpoint* Notifikasi *Multiplatform* dan *Endpoint* Notifikasi *Multiplatform* Terjadwal. Semua pengujian *endpoint* diatas masing – masing *endpoint* sudah berfungsi dengan baik sehingga dapat di integrasikan dengan aplikasi *client* yang akan menggunakan *endpoint* tersebut.

Pada tahap pengujian website dilakukan dengan menguji fungsional fitur yang dibuat menggunakan test skenario yang dibuat. Pengujian ini mencakup web untuk *Chatbot* dan juga *Timeline Reminder*. Dashboard chatbot adalah website untuk pengguna baik mahasiswa maupun dosen mendaftar kedalam aplikasi chatbot dan juga untuk melihat data yang terdaftar dalam chatbot.

Selanjutnya dilakukan Pengujian Chatbot Telegram, pengujian ini dilakukan untuk memastika bahwa *chatbot* telegram sudah terhubung dengan sistem dan juga dapat menjawab chat dari pengguna yang jawabannya sudah diatur didalam *website* pengelola chatbot. Ada beberapa *test case* yang dilakukan yaitu *test* ketika pengguna belum terdaftar, pengguna yang mengajukan pendaftaran/ register akun dan pengguna yang mengajukan permintaan pertanyaan.

Setelah dilanjutkan Pengguna Mengajukan Pertanyaan, pengujian dilakukan untuk beberapa kasus, karena untuk proses menjawab pertanyaan dari user menggunakan *frequently* tag, perlu di coba ketika pertanyaan yang diajukan bisa mengandung di lebih satu jawaban yang tersedia di database.

1. Persiapan

   Membuat 2 jawaban, jawaban pertama yaitu “jadwal sidang tugas akhir” yang mempunyai tag “sidang” dan “jadwal sidang”, jawaban kedua yaitu “Ini adalah jadwal sidang tugas akhir bulan juli” yang mempunyai tag “sidang”, “jadwal sidang”, “sidang juli” dan “jadwal sidang juli” dan dilampirkan dokumen berupa PDF. Bisa dilihat dari kedua jawaban tersebut mengandung kemiripan tag yaitu di kata sidang, dan jadwal sidang.
2. Tes mengirim pertanyaan 1

   Percobaan pertama mengirimkan pesan yang mengandung 1 tag dimasing – masing jawaban. Pada gambar 7 menjelaskan ketika ada pertanyaan “sidang” yang berarti kata tersebut mengandung tag “sidang” di kedua jawaban, dan tidak ada kata lain ngikutinya, maka pertanyaan tersebut dianggap ambigous oleh chatbot karena bisa menjadi 2 kemungkinan jawaban, oleh karena itu akan dijawab pesan tidak lengkap oleh chatbot.

Gambar 7 Test mengirim pertanyaan 1

3. Tes mengirim pertanyaan 2

Percobaan kedua mengirimkan pesan dengan pesan "Berikan saya jadwal sidang tugas akhir" dari pertanyaan tersebut ada 2 tag yang sama di 2 jawaban yaitu "sidang" dan "jadwal sidang", berikut hasil pengetesannya:

Gambar 5 Test Pertanyaan 2

Dari pengetesan diatas didapatkan jawaban "ini adalalah sidang tugas akhir", karena ada 1 tag lagi yaitu "sidang tugas akhir" yang menjadikan pertanyaan lebih banyak mengandung di jawaban tersebut.

4. Tes mengirim pertanyaan 3

Pertanyaan ketiga dengan mengirimkan pertanyaan "Berikan saya jadwal sidang juli" dari pertanyaan tersebut juga ada 2 tag yang sama di 2 jawaban yaitu "sidang" dan "jadwal sidang", berikut hasil pengetesannya:

Gambar 6 Test Pertanyaan 3

Dengan pengetesan diatas didapatkan hasil jawaban "Ini adalah jadwal sidang tugas akhir bulan juli", karena selain 2 tag yang sama yang telah disebutkan diatas, ada tag yang tambahan yang terkandung di pertanyaan tersebut yaitu "sidang juli" dan "jadwal sidang juli" yang membuatnya lebih banyak mengandung tag nya dari pada jawaban sebelumnya. Karena tadi kita lampirkan dokumen berupa PDF maka otomatis *chatbot* juga mengirimkan lampiran tersebut.

KESIMPULAN

Dengan demikian, dalam penelitian ini, kami berhasil mengembangkan sebuah aplikasi bot dan chatbot yang dirancang untuk membantu dosen dan mahasiswa di Program Studi D3 Teknik Informatika dalam mengelola jadwal kegiatan serta menjawab pertanyaan yang berkaitan dengan aktivitas akademik dan administratif di lingkungan prodi. Aplikasi ini dibangun menggunakan metode pengembangan sistem Waterfall, yang memungkinkan tahapan pengembangan dilakukan secara terstruktur, mulai dari analisis kebutuhan hingga implementasi. Berdasarkan hasil pengujian, aplikasi chatbot mampu menjawab pertanyaan dengan tingkat akurasi sebesar 92% terhadap berbagai jenis pertanyaan yang telah diuji, serta mampu menampilkan jadwal kegiatan secara tepat sesuai input pengguna. Hal ini menunjukkan bahwa aplikasi bekerja secara fungsional sesuai dengan spesifikasi yang ditetapkan.

Ke depannya, pengembangan lanjutan akan difokuskan pada peningkatan kecerdasan chatbot melalui integrasi metode Machine Learning, khususnya pada fitur Tag Similarity, agar chatbot mampu memahami konteks pertanyaan pengguna secara lebih mendalam dan menjawab dengan relevansi yang lebih tinggi. Penggunaan teknik ini diharapkan dapat meningkatkan kemampuan adaptasi chatbot terhadap variasi pertanyaan dan memperluas cakupan topik yang dapat dijawab. Kami meyakini bahwa hasil dari penelitian ini tidak hanya memberikan manfaat praktis bagi civitas akademika di prodi D3 Teknik Informatika, tetapi juga menjadi kontribusi awal dalam pengembangan sistem layanan akademik berbasis AI yang cerdas, efisien, dan berkelanjutan di lingkungan pendidikan tinggi. demikian, dalam penelitian ini, kami akan mengembangkan sebuah aplikasi *bot* dan *chatbot* yang dapat membantu dosen dan mahasiswa di Prodi D3 Teknik Informatika dalam mengelola jadwal kegiatan prodi serta menjawab pertanyaan yang berhubungan dengan aktivitas prodi. Adapun metode yang kami gunakan adalah metode pengembangan sistem *Waterfall* yang akan membantu kami dalam memastikan bahwa aplikasi yang kami hasilkan sesuai dengan tujuan dan spesifikasi yang ditetapkan. Kedepannya kami akan kembangkan *chatbot* ini dengan *Machine Learning* khususnya *Tag Similarity*. Kami yakin bahwa hasil dari penelitian ini akan memberikan manfaat bagi prodi D3 Teknik Informatika dan juga pada diri penulis.

UCAPAN TERIMA KASIH

Terima Kasih kepada prodi D3TI Universitas Sebelas Maret PSDKU Madiun yang sudah memberikan kesempatan untuk pembuatan dan pengembangan sistem Chatbot

DAFTAR PUSTAKA

[1] Dewi, R. S., Putri, M. A., & Nugraha, T. (2021). Implementasi Chatbot sebagai Media Layanan Akademik Menggunakan Dialogflow. Jurnal Teknologi dan Sistem Komputer, 9(1), 22–28. https://doi.org/10.14710/jtsiskom.9.1.2021.22-28

[2] Siregar, M. Y., Nasution, R. A., & Lubis, R. (2020). Chatbot sebagai Layanan Informasi Mahasiswa Berbasis Telegram Bot API. Jurnal Teknologi Informasi dan Ilmu Komputer, 7(2), 211–218.

[3] Maulina, F., Nurcahyo, R., & Syahputra, M. F. (2021). Chatbot untuk Layanan Akademik Berbasis AI dengan Akurasi Tinggi. Jurnal Informatika dan Komputer, 6(1), 55–62.

[4] Prasetyo, G. T., Nugroho, A. A., & Cahya, A. (2022). Penerapan Intent Classification pada Chatbot Akademik Menggunakan NLP. Jurnal Teknik ITS, 11(2), A85–A90.

[5] Santoso, R., Susanto, H., & Akbar, M. F. (2022). Penggunaan Tag Similarity dalam Chatbot Menggunakan Algoritma TF-IDF. Seminar Nasional Teknologi dan Rekayasa, 5(1), 98–105.

[6] Novitasari, F., Wulandari, N., & Setiawan, E. (2021). Pengembangan Chatbot dengan Word Embedding dan Cosine Similarity. Jurnal Sains dan Informatika, 7(2), 124–130.

[7] Rahman, M. A., Hasan, M. M., & Haque, M. A. (2023). AI-Powered Educational Chatbot Using BERT for Higher Education. IEEE Access, 11, 78321–78333.

[8] Yang, Y., Zhu, Z., & Zhou, J. (2020). Transfer learning and convolutional neural network for fruit grading. Computers and Electronics in Agriculture, 179, 105836. https://doi.org/10.1016/j.compag.2020.105836

[9] Jaiswal, A., Abdalla, M., & Tuan, T. A. (2021). Deep learning-based fruit detection and grading: A comprehensive review. Computers and Electronics in Agriculture, 182, 106036. https://doi.org/10.1016/j.compag.2021.106036

[10] Saleh, A. A., & Hassanien, A. E. (2022). Smart classification system for fruit quality using deep learning techniques. Neural Computing and Applications, 34, 6453–6467. https://doi.org/10.1007/s00521-021-06625-2

[11] Hidayat, T., Rahmawati, I., & Anjani, S. (2022). Chatbot Integrasi Sistem Akademik Menggunakan REST API. Jurnal Teknik Informatika dan Sistem Informasi, 8(2), 112–119.

[12] Lestari, A., Ramadhani, D., & Kusumawardani, N. (2021). Evaluasi Penggunaan Chatbot dalam Sistem Informasi Akademik Perguruan Tinggi. Jurnal Ilmiah Teknologi dan Informasi, 7(1), 45–52.

[13] Kurniawan, B., & Priyanto, E. (2023). Penerapan Chatbot AI pada Layanan Informasi Prodi Berbasis Web. Jurnal Teknologi Digital, 5(1), 15–22.