Pengembangan Sistem Helpdesk Menggunakan Chatbot Dengan Metode Retrieval-Augmented Generation (RAG)

Ilza Ichsanudin Rachman Pratama[1], Bonda Sisephaputra[2]
[1,2] Teknik Informatika, Teknik Informatika, Universitas Negeri Surabaya
[1]ilza.19092@mhs.unesa.ac.id
[2]bondasisephaputra@unesa.ac.id

Abstrak— Penelitian ini bertujuan untuk mengembangkan sistem helpdesk menggunakan chatbot berbasis metode Retrieval-Augmented Generation (RAG) guna meningkatkan efisiensi layanan administratif di lingkungan kampus. Sistem ini dibangun menggunakan FastAPI sebagai backend dan Next.js sebagai frontend, yang memastikan antarmuka yang responsif dan ramah pengguna. Chatbot memanfaatkan kemampuan text generation dari model Gemini yang dikombinasikan dengan pencarian semantik menggunakan FAISS DATABASE dari OpenAI, sehingga memungkinkan penyampaian informasi yang akurat dan relevan secara real-time. Selama fase pengujian, sistem ini berhasil menangani berbagai pertanyaan pengguna secara bersamaan, memberikan respons yang cepat dan andal. Hasil penelitian menunjukkan bahwa sistem ini memudahkan akses terhadap informasi akademik dan administratif, serta menyederhanakan interaksi pengguna dengan layanan helpdesk kampus. Pengembangan di masa depan akan mencakup deployment server dan penyesuaian konfigurasi prompt untuk meningkatkan akurasi serta kapabilitas sistem.

Kata Kunci— Sistem helpdesk, chatbot, RAG, pencarian semantik

I. PENDAHULUAN

Dalam era digital yang terus berkembang, teknologi kecerdasan buatan (AI) telah memberikan dampak signifikan pada berbagai aspek kehidupan, termasuk di bidang pendidikan dan layanan administratif kampus. Perguruan tinggi, sebagai institusi yang mengelola banyak data akademik dan administratif, sering kali dihadapkan pada tantangan untuk menyediakan layanan yang cepat, akurat, dan efisien kepada mahasiswa dan staf. Salah satu solusi yang semakin populer adalah penerapan chatbot, yaitu asisten virtual yang dapat merespons pertanyaan pengguna secara otomatis.

Chatbot telah terbukti efektif dalam mengurangi beban kerja staf administratif dengan menangani pertanyaan yang berulang dan memberikan informasi yang akurat secara cepat. Penggunaan chatbot di kampus dapat mengurangi beban kerja staf akademik dan meningkatkan efisiensi dalam menangani pertanyaan mahasiswa [1]. Namun, sebagian besar chatbot konvensional masih terbatas pada kemampuan pencocokan pola atau string matching, sehingga sering kali gagal memberikan jawaban yang sesuai dengan konteks dan kebutuhan spesifik pengguna. Keterbatasan ini dapat diatasi dengan penggunaan model kecerdasan buatan yang lebih canggih, seperti Retrieval-Augmented Generation (RAG).

RAG adalah metode yang menggabungkan dua pendekatan utama dalam pemrosesan bahasa alami (Natural Language Processing/NLP), yaitu retrieval (pencarian) dan generation (generasi). Dalam konteks ini, RAG memungkinkan chatbot untuk mencari informasi yang relevan dari basis data atau sumber pengetahuan yang telah disediakan, kemudian menggabungkannya dengan kemampuan generatif untuk menghasilkan respons yang lebih kaya, relevan, dan kontekstual. Dengan demikian, chatbot yang menggunakan metode RAG tidak hanya mampu memberikan jawaban yang tepat, tetapi juga dapat memberikan informasi tambahan yang bermanfaat bagi pengguna [2].

Penelitian ini bertujuan untuk mengembangkan sistem chatbot berbasis RAG yang dapat meningkatkan efisiensi layanan administratif di lingkungan kampus. Dengan menggunakan pendekatan ini, diharapkan chatbot dapat membantu mahasiswa dan staf kampus dalam mengakses informasi akademik, administratif, serta berbagai layanan lainnya dengan lebih mudah, cepat, dan tepat. Selain itu, penerapan RAG diharapkan dapat mengurangi beban kerja staf administrasi kampus sehingga mereka dapat lebih fokus pada tugas-tugas yang lebih kompleks [3].

Berbagai penelitian sebelumnya telah membuktikan efektivitas penggunaan chatbot dalam lingkungan akademik. Misalnya, Ardiansyah et al. menemukan bahwa chatbot yang menggunakan metode string matching mampu memberikan jawaban yang akurat pada sistem informasi mahasiswa, sementara Mendoza et al. menunjukkan bahwa chatbot dapat memperbaiki interaksi antara mahasiswa dan dosen dalam kegiatan akademik [4][5]. Namun, penggunaan chatbot berbasis RAG masih relatif baru, terutama dalam konteks layanan administratif kampus, sehingga penelitian ini memberikan kontribusi penting dalam pengembangan teknologi ini.

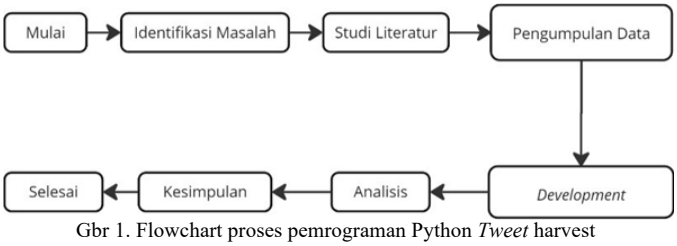
Selain meningkatkan efisiensi, penggunaan chatbot berbasis RAG diharapkan dapat memberikan pengalaman pengguna yang lebih baik. Sebuah studi yang dilakukan oleh Yuditha Ichsani menemukan bahwa kualitas interaksi pengguna dengan sistem informasi kampus sangat mempengaruhi tingkat kepuasan mereka [6]. Integrasi chatbot yang cerdas dan responsif dalam sistem informasi kampus diharapkan dapat memberikan interaksi yang lebih dinamis dan relevan, sehingga meningkatkan kepuasan pengguna secara keseluruhan.

Dengan latar belakang ini, penelitian ini difokuskan pada pengembangan dan implementasi chatbot yang menggunakan

metode RAG. Penelitian ini akan menguji bagaimana chatbot dapat berinteraksi dengan pengguna, menyampaikan informasi administratif, serta menyelesaikan pertanyaan dengan lebih cepat dan akurat. Selain itu, penelitian ini akan menganalisis dampak implementasi chatbot terhadap efisiensi layanan administrasi kampus, serta kepuasan pengguna yang diukur melalui umpan balik dari mahasiswa dan staf [7].

Penelitian ini juga diharapkan memberikan kontribusi praktis bagi institusi pendidikan lainnya yang tertarik untuk mengadopsi teknologi AI dalam meningkatkan layanan administratif mereka. Hasil dari penelitian ini dapat menjadi dasar pengembangan lebih lanjut di bidang chatbot berbasis AI, khususnya yang menggunakan metode RAG, serta aplikasi teknologi AI lainnya dalam lingkungan akademik. Dengan demikian, penelitian ini tidak hanya berfokus pada implementasi teknologi, tetapi juga memberikan wawasan strategis tentang bagaimana AI dapat diintegrasikan ke dalam proses administrasi pendidikan yang lebih luas [8].

II. METODE PENELITIAN



Gbr 1. Flowchart proses pemrograman Python *Tweet* harvest

A. *Identifikasi Masalah*

Berdasarkan hasil eksplorasi dari peneliti, dirumuskan pertanyaan penelitian yang jelas, terukur, relevan dengan kebutuhan saat ini, dan berpotensi memberikan kontribusi yang berarti bagi pengembangan sistem informasi administrasi kampus. Perumusan masalah ini melibatkan pemahaman mendalam tentang tantangan yang dihadapi oleh pengguna dan administrasi kampus, serta meninjau tren terkini dalam pengembangan teknologi chatbot.

B. *Studi Literatur*

Pada tahap ini studi literatur dilakukan dengan menyelidiki secara mendalam literatur yang ada terkait penggunaan chatbot dalam layanan administratif kampus, dengan fokus khusus pada metode RAG. Informasi dikumpulkan dari berbagai sumber, seperti jurnal akademik, proceeding konferensi, dan studi kasus, untuk membangun konteks teoretis penelitian.

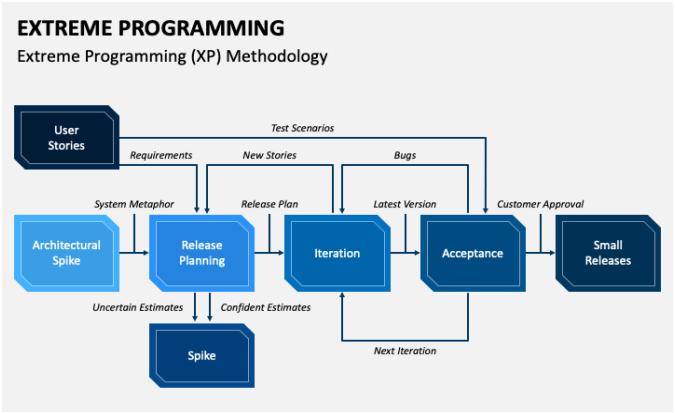
C. *Pengumpulan Data*

Pengumpulan data merupakan proses untuk mengumpulkan informasi guna mengevaluasi hasil penelitian dan eksperimen yang sedang dilakukan [10], Tahap pengumpulan data berfokus pada pengembangan chatbot berbasis Retrieval-Augmented Generation (RAG). Penelitian ini akan menggunakan data yang tersedia secara umum dari Universitas Negeri Surabaya (UNESA), yang dapat diakses melalui website resmi unesa.ac.id. Data ini mencakup informasi penting yang berpotensi membantu chatbot dalam menjawab berbagai pertanyaan yang diajukan oleh pengguna.

Proses pengumpulan data akan melibatkan langkah-langkah berikut:

a. Identifikasi Sumber Data: Mengidentifikasi halaman-halaman di website unesa.ac.id yang mengandung informasi relevan, seperti panduan akademik, informasi fakultas dan jurusan, kalender akademik, berita dan pengumuman, serta FAQ.
b. Ekstraksi Data: Mengambil data dari halaman-halaman tersebut secara sistematis. Ini mungkin melibatkan scraping data atau mengunduh dokumen yang tersedia secara publik.
c. Penyimpanan Data: Menyimpan data yang telah dikumpulkan dalam format teks, seperti PDF, DOC, atau file teks biasa, untuk memudahkan pemrosesan lebih lanjut.
d. Pemrosesan dan Pengindeksan: Memproses data untuk memastikan konsistensi dan keakuratan, serta mengindeks data agar dapat diakses dengan cepat oleh chatbot.

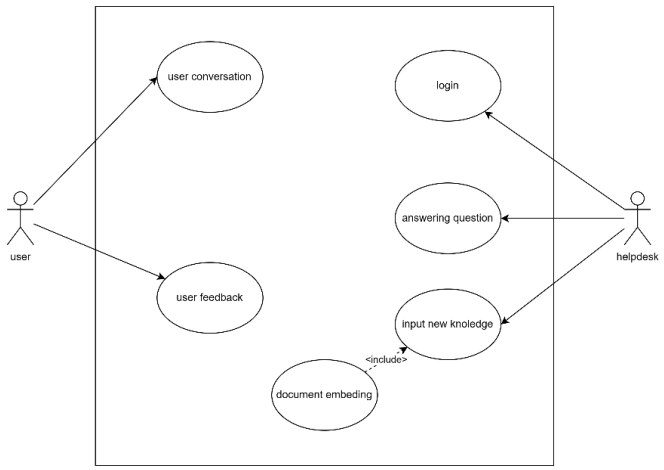
D. *Development*



Gbr 2. Extreme Programming

1) *User Stories*

Dalam *User stories* akan mendeskripsikan fitur dari 2 aktor/pengguna (*User* dan *helpdesk*). *User stories* ditampilkan dalam bentuk *use case diagram, activity diagram, high-level flow diagram,* dan *sequence diagram.*



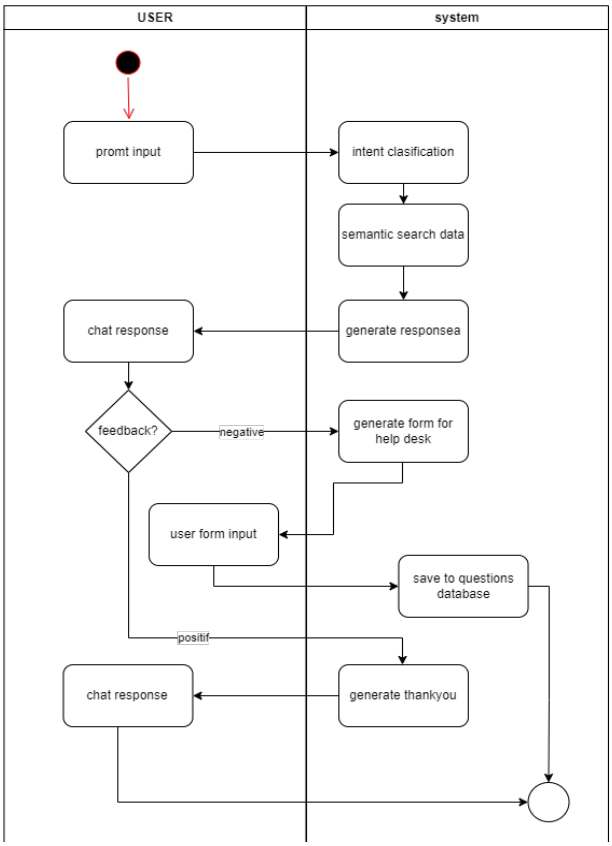
Gbr 3. Use Case Diagram

Use case diagram pada Gbr 3. diatas memiliki 2 aktor yaitu *User* (Pengguna) dan *Helpdesk* (Admin). Berikut fitur-fitur yang tersedia untuk kedua aktor tersebut:

1.1 User (Pengguna)

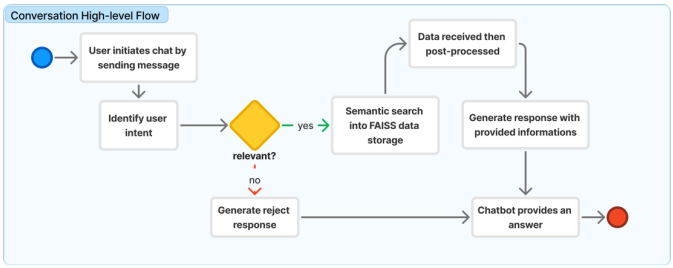
1.1.1 Fitur User Conversation

Pengguna dapat memasukkan pertanyaan atau pernyataan melalui *interface* chat yang disediakan oleh *chatbot*. Berikut *activity diagram* dari fitur *user conversation* pada gambar 3.4:



Gbr 4. Activity Diagram Fitur User Conversation

Adapun Gbr 5. dibawah ini merupakan *high-level flow* fitur *User conversation*:



Gbr 5. High-Level Flow Fitur User Conversation

a. User initiates chat by sending message

Langkah pertama pengguna mulai berinteraksi dengan sistem chatbot. Pesan yang dikirim oleh penggunac bisa berupa pertanyaan, permintaan informasi, atau keluhan. Chatbot kemudian menerima pesan ini sebagai input untuk diproses lebih lanjut. Proses ini menunjukkan bagaimana sistem memfasilitasi komunikasi antara pengguna dan teknologi, memulai seluruh alur kerja chatbot.

b. Identify User Intent

langkah berikutnya adalah mengidentifikasi maksud pengguna dari pesan tersebut. Sistem chatbot menganalisis teks pesan untuk memahami apa yang sebenarnya diinginkan atau ditanyakan oleh pengguna.

c. Relevance check

Pada tahap ini, chatbot memverifikasi apakah permintaan tersebut dapat dijawab dengan informasi yang tersedia dalam database pengetahuan. Jika permintaan relevan, proses akan dilanjutkan ke langkah pencarian semantik. jika permintaan dianggap tidak relevan atau di luar cakupan chatbot, sistem akan menghasilkan respons penolakan

d. Semantic search into FAISS data storage

Jika permintaan dianggap relevan, sistem melakukan pencarian semantik dalam penyimpanan data FAISS (Facebook AI Similarity Search). Dengan menggunakan pencarian semantik, chatbot dapat mengakses dan mengambil data yang relevan dari sumber daya pengetahuan yang luas, memastikan bahwa jawaban yang diberikan berdasarkan informasi yang akurat dan terbaru.

e. Data received then post-processed

Setelah data yang relevan ditemukan melalui pencarian semantik, data tersebut kemudian diterima dan diproses lebih lanjut. Proses ini melibatkan pembersihan data, penggabungan informasi dari berbagai sumber, atau format ulang data agar sesuai dengan format yang diperlukan untuk menghasilkan jawaban.

f. Generate response with provided informations

Dengan data yang telah diproses, langkah berikutnya adalah menghasilkan respons yang akan diberikan kepada pengguna. Sistem menggunakan model generatif untuk menyusun jawaban berdasarkan informasi yang tersedia.

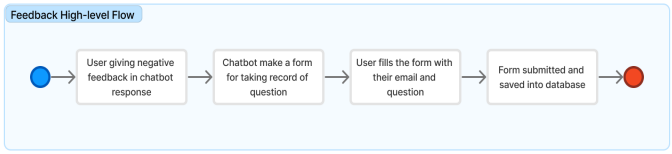
g. Chatbot provides an answer

Langkah terakhir dalam alur percakapan adalah ketika *chatbot* memberikan jawaban kepada pengguna. Jawaban

ini diharapkan relevan, informatif, dan kontekstual, sesuai dengan permintaan awal pengguna. Setelah melalui semua tahap sebelumnya, *chatbot* kini siap untuk menyampaikan jawaban yang telah dihasilkan

1.1.2 *Fitur User Feedback*

Fitur ini memungkinkan pengguna untuk mengisi formulir yang dapat digunakan untuk menyampaikan pertanyaan yang lebih terperinci dan chatbot tidak dapat menjawabnya, form akan diteruskan ke Helpdesk dan akan diinformasikan kembali ke User melalui email. Gbr 5. dibawah ini merupakan high-level flow fitur User feedback.



Gbr 5. High-Level Flow Fitur User Feedback

a. *User giving negative feedback in chatbot response*
Langkah pertama adalah ketika pengguna memberikan umpan balik negatif terhadap respons *chatbot*. Pengguna dapat mengindikasikan ketidakpuasan atau masalah dengan jawaban yang diberikan oleh *chatbot*.

b. *Chatbot make a form for taking record of question*
Setelah menerima umpan balik negatif, chatbot secara otomatis membuat formulir untuk mencatat pertanyaan atau masalah yang dihadapi pengguna. Formulir ini bertujuan untuk mengumpulkan informasi lebih lanjut mengenai umpan balik yang diberikan oleh pengguna..

c. *User fills the form with their email and question*
Pengguna kemudian mengisi formulir yang disediakan oleh *chatbot*. Dalam formulir ini, pengguna dapat memasukkan detail pertanyaan mereka serta alamat email

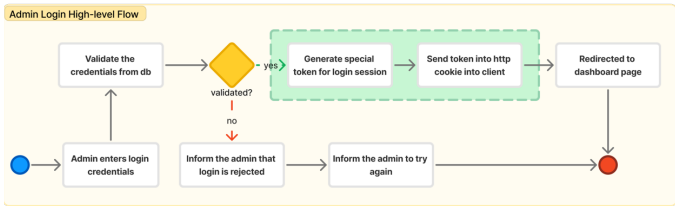
d. *Form submitted and saved into database*
Setelah pengguna mengisi dan mengirimkan formulir, data yang terkandung di dalamnya disimpan dalam *database*.

1.2 *Helpdesk (Admin)*

Fitur untuk *Helpdesk* atau Admin terdiri dari fitur *Login Admin, Answering Question,* dan *Input New Knowledge.* Fitur ini ditampilkan dan dijelaskan dalam bentuk *activity diagram, high-level flow diagram,* dan *sequence diagram*.

1.2.1 *Fitur Login Admin*

Staf Helpdesk atau admin memiliki kemampuan untuk login ke sistem. Ini dilakukan untuk mengakses fitur administratif, seperti pengelolaan basis pengetahuan atau untuk membantu pengguna dalam menjawab pertanyaan yang belum masuk dalam knowledge base. Gbr 6. dibawah ini merupakan high-level flow fitur Login Admin.



Gbr 6. High-Level Flow Fitur Login Admin.

a. *Admin enters login credentials*
Langkah pertama adalah admin memasukkan kredensial login mereka, seperti nama pengguna dan kata sandi, ke dalam sistem. Proses ini adalah titik awal di mana admin mencoba mengakses sistem dengan identitas mereka.

b. *Validate the credentials from db*
Setelah admin memasukkan kredensial mereka, sistem akan memvalidasi informasi tersebut dengan mencocokkannya dengan data yang ada di *database*. Langkah ini penting untuk memastikan bahwa hanya admin yang sah yang dapat mengakses sistem.

c. *Validated?*
Sistem kemudian akan mengecek apakah kredensial yang dimasukkan sesuai dengan yang ada di *database*. Jika kredensial valid, proses *login* akan berlanjut ke langkah berikutnya. Jika tidak valid, sistem akan menolak *login* tersebut.

d. *Generate special token for login session*
Jika kredensial valid, sistem akan menghasilkan token khusus untuk sesi *login* tersebut. Token ini berfungsi sebagai identifikasi unik untuk sesi *login* admin dan membantu menjaga keamanan selama sesi berlangsung.

e. *Send token into http cookie into client*
Token yang dihasilkan kemudian dikirim dan disimpan dalam bentuk cookie HTTP di sisi klien (*browser* admin). Penyimpanan token ini memungkinkan sistem untuk mengelola sesi *login* dan memastikan bahwa admin yang sah tetap terotentikasi selama sesi mereka.

f. *Redirected to dashboard page*
Setelah token berhasil disimpan, admin akan diarahkan ke halaman *dashboard*. Pengalihan ini biasanya dikelola oleh *session* manager, yang memastikan bahwa admin telah berhasil *login* dan memiliki akses yang tepat ke fitur-fitur yang diperlukan.

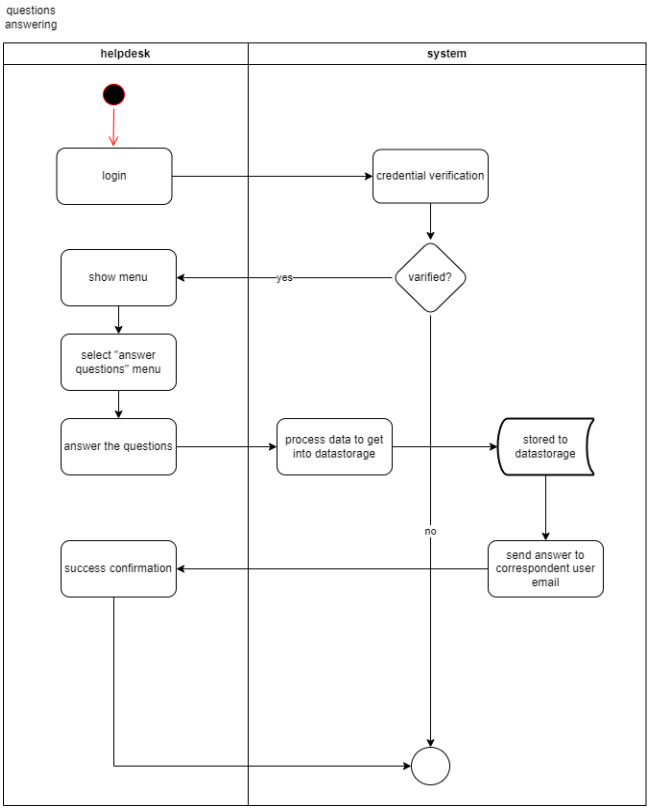
g. *Inform the admin that login is rejected*
Jika kredensial yang dimasukkan tidak valid, sistem akan memberitahukan admin bahwa *login* mereka ditolak. Pemberitahuan ini penting agar admin memahami bahwa ada masalah dengan kredensial mereka dan mereka perlu mencoba lagi dengan informasi yang benar.

h. *Inform the admin to try again*
Setelah *login* ditolak, admin akan diinformasikan untuk mencoba lagi. Ini memberikan kesempatan bagi admin untuk memasukkan kembali kredensial mereka atau mengoreksi kesalahan yang mungkin terjadi.

1.2.2 *Fitur Answering Question*

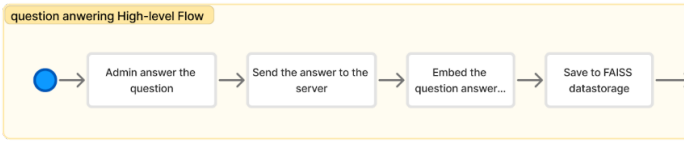
Setelah pengguna memasukkan pertanyaan, sistem *chatbot* akan memproses *input* tersebut dan memberikan jawaban. Proses ini bisa otomatis atau mungkin memerlukan intervensi manual oleh staf *Helpdesk* jika pertanyaan di luar cakupan pengetahuan *chatbot*.

Berikut *activity diagram* dari fitur *Answering Question Helpdesk* (Admin) pada Gbr 7:



Gbr 7. Activity Diagram Fitur Answering Question Helpdesk (Admin)

Adapun *high-level flow diagram* fitur *Answering Question Helpdesk* (Admin) pada gambar 3.12.



Gbr 8. High-Level Flow Diagram Fitur Answering Question Helpdesk (Admin)

a. Admin answer the question

Langkah pertama dalam alur ini adalah admin memberikan jawaban atas pertanyaan yang diajukan. Admin berperan sebagai pihak yang memiliki pengetahuan dan informasi yang diperlukan untuk menjawab pertanyaan dari pengguna.

b. Send the answer to the server

Setelah admin menjawab pertanyaan, jawaban tersebut dikirimkan ke server. Server akan bertindak sebagai pengelola dan penyimpan data.

c. Embed the question and answer

Langkah berikutnya adalah menyematkan *(embed)* pertanyaan dan jawaban tersebut. Proses ini melibatkan transformasi teks pertanyaan dan jawaban ke dalam bentuk vektor yang dapat diproses oleh sistem, sehingga memudahkan dalam pencarian dan pengambilan informasi di masa mendatang.

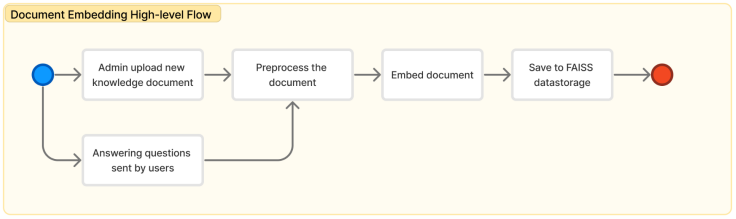
d. Save to FAISS datastore

Vektor yang dihasilkan kemudian disimpan ke dalam penyimpanan data FAISS (Facebook AI *Similarity Search*). FAISS adalah *library* yang memungkinkan pencarian yang cepat dan efisien terhadap vektor, sehingga memungkinkan sistem *chatbot* untuk menemukan jawaban yang paling relevan dan akurat berdasarkan pertanyaan pengguna.

1.2.3 *Fitur Input New Knowledge*

Admin atau *Helpdesk* dapat memasukkan pengetahuan atau informasi baru ke dalam sistem. Ini adalah cara untuk memperbarui dan memperluas basis pengetahuan yang *chatbot* gunakan untuk menjawab pertanyaan. Berikut *activity diagram* dari fitur *Input New Knowledge* (Admin):

Adapun *high-level* flow fitur *input new knowledge* terdapat pada Gbr 9.:



Gbr 9. High-Level Flow Fitur Input New Knowledge

a. Users give negative feedback in chatbot responses

Pengguna memberikan umpan balik negatif pada jawaban atau respon yang diberikan oleh *chatbot*. Ini menunjukkan bahwa pengguna tidak puas atau memiliki masalah dengan informasi yang diberikan oleh *chatbot*.

b. Chatbot creates a form to record questions

Setelah menerima umpan balik negatif, *chatbot* secara otomatis membuat dan menampilkan formulir kepada pengguna. Formulir ini dirancang untuk mencatat detail pertanyaan atau masalah yang dihadapi oleh pengguna. Langkah ini bertujuan untuk mengumpulkan informasi lebih lanjut mengenai umpan balik negatif yang diberikan.

c. Users fill out the form with their email and inquiry

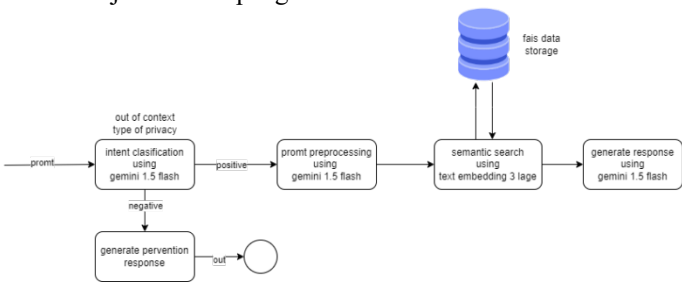
Pengguna kemudian mengisi formulir yang disediakan dengan alamat email mereka dan pertanyaan atau masalah yang mereka hadapi. Ini membantu dalam mengidentifikasi pengguna dan mendokumentasikan masalah mereka secara lebih rinci.

d. Forms are submitted and saved to the database

Setelah pengguna mengisi dan mengirimkan formulir, data dari formulir tersebut dikirimkan ke server dan disimpan ke dalam *database*. Penyimpanan ini memungkinkan sistem untuk melacak dan menganalisis umpan balik negatif yang diterima, serta memberikan tindak lanjut yang diperlukan berdasarkan informasi yang dikumpulkan

2) Architectural Spike

Architectural spike merupakan eksperimen yang dilakukan untuk memahami bagaimana aspek teknis tertentu dari sistem akan bekerja dan mempengaruhi *User stories*



Gbr 10. System Architecture

2.1 Input dari Pengguna *(prompt)*

Pengguna memulai interaksi dengan mengirimkan pertanyaan melalui *prompt input*.

2.2 Intent Clasificaton

Intent classification adalah proses untuk memahami maksud pengguna dari pertanyaan yang diajukan, dan dalam penelitian ini, model Gemini 1.5 Flash digunakan agar chatbot dapat mengidentifikasi maksud tersebut dengan lebih akurat. Klasifikasi maksud ini terbagi dalam tiga kategori utama: ACADEMIC_ADMINISTRATION, RESOURCE_SERVICE, dan SUPPORT. Dalam kategori ACADEMIC_ADMINISTRATION,

Chatbot mendukung pencarian informasi akademik dan administratif, seperti detail program akademik, jadwal mata kuliah, prosedur administrasi pendaftaran, dan bantuan terkait akun pribadi, misalnya saat pengguna lupa password portal. Kategori RESOURCE_SERVICE berfokus pada bantuan akses sumber daya kampus dan dukungan teknis, seperti menangani masalah jaringan, mencari lokasi kampus, serta layanan bimbingan karir dan konseling. Sementara itu, kategori SUPPORT mencakup bantuan darurat dan keluhan terkait fasilitas kampus, seperti pelaporan masalah keamanan atau kondisi fasilitas. Jika maksud tidak sesuai dengan ketiga kategori ini, intent OTHER akan dipilih.

Dengan klasifikasi ini, chatbot lebih efisien dalam pencarian dan pengambilan informasi dari basis data FAISS, karena data sudah dilabeli sesuai dengan maksud pengguna.

2.3 Prompt Preprocessing

Setelah proses klasifikasi *intent*, *prompt* yang dinilai relevan akan diproses lebih lanjut melalui tahap pra-pemrosesan. Tahap ini mencakup beberapa langkah penting, seperti pembersihan teks, perbaikan kesalahan ketik (*typo*), dan peningkatan kualitas keseluruhan *prompt*. Pra-pemrosesan ini bertujuan untuk memastikan bahwa *prompt* yang dihasilkan bebas dari kesalahan ketik dan siap digunakan dalam pencarian semantik. Pada penelitian ini, digunakan model *chatbot* Gemini 1.5 *Flash* untuk melaksanakan proses pra-pemrosesan tersebut.

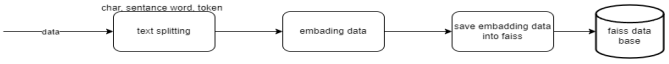
Penggunaan Gemini 1.5 *Flash* dalam prapemrosesan memungkinkan peningkatan akurasi dan efisiensi dalam menangani teks, sehingga *prompt* yang dihasilkan lebih berkualitas dan sesuai untuk digunakan dalam langkah-langkah selanjutnya dalam sistem *chatbot*.

2.4 Pencarian Semantik (Semantic Search)

Sistem pencarian semantik pada chatbot menggunakan model Retrieval-Augmented Generation (RAG) untuk memberikan respons yang relevan dan informatif. Model RAG menggabungkan kemampuan generatif dengan pengambilan informasi dari database eksternal, sehingga jawaban dapat diperbarui sesuai data terbaru. Data terkait layanan administratif kampus dikumpulkan, diubah menjadi representasi numerik dengan teknik NLP, dan disimpan dalam database FAISS untuk pencarian cepat. Saat pengguna mengajukan pertanyaan, sistem melakukan pencarian semantik di FAISS berdasarkan kesamaan dengan kueri pengguna. Informasi yang ditemukan diproses oleh RAG untuk menghasilkan jawaban yang kontekstual dan akurat, yang disampaikan melalui antarmuka chatbot.

2.5 FAISS data Storage (Knowledge Database):

Database ini menyimpan informasi yang dapat diakses oleh *chatbot* untuk menjawab pertanyaan. Data data akan di disimpan dalam bentuk FAISS data *Storage* untuk proses pencarian semantic data search.



Gbr 11. Data Preprocessing

Diagram ini mengilustrasikan proses penyiapan data teks untuk digunakan dalam *database* FAISS, yaitu pustaka untuk pencarian kesamaan yang efisien. Berikut penjelasan detail dari setiap langkah:

a. Masukan Data: Prosesnya dimulai dengan data teks mentah sebagai masukan. Data ini dapat dalam berbagai format, seperti dokumen, kalimat, atau kata-kata individual.
b. Pemisahan Teks: Data teks dipecah menjadi unit yang lebih kecil, seperti karakter, kata, kalimat, atau token. Unit pemisahan spesifik tergantung pada tingkat granularitas yang diinginkan untuk pengindeksan dan pencarian dalam *database* FAISS.
c. Penyematan Data: Setiap unit teks (karakter, kata, kalimat, atau token) diubah menjadi representasi numerik yang disebut penyematan. Penyematan menangkap makna semantik dan hubungan antar kata, memungkinkan perbandingan kesamaan.

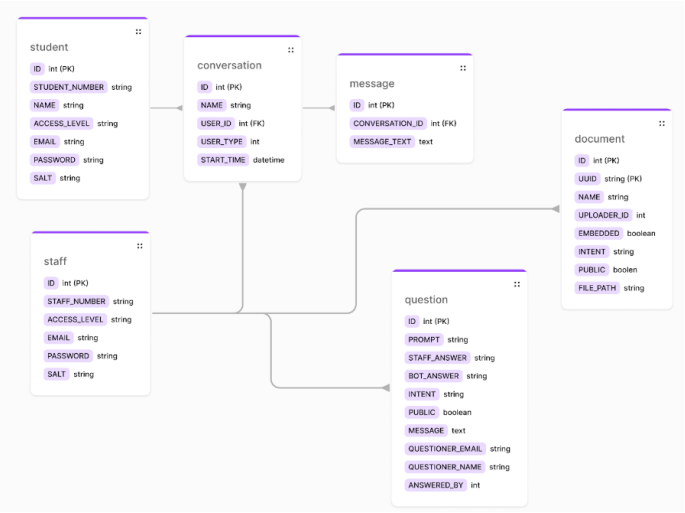
d. Menyimpan Data Penyematan ke FAISS: Penyematan yang dihasilkan disimpan dalam *database* FAISS. FAISS menyediakan struktur pengindeksan dan algoritma yang efisien untuk mencari dan mengambil penyematan serupa berdasarkan metrik jarak seperti kesamaan kosinus.
e. *Database* FAISS: *Database* FAISS sekarang berisi penyematan yang diindeks, memungkinkan pencarian kesamaan yang cepat dan efisien. Ini memungkinkan menemukan unit teks yang secara semantik mirip dengan kueri yang diberikan

2.6 *Respons Generatif (Generative Response):*

Berdasarkan informasi dari pencarian semantik, sistem menentukan apakah dapat menghasilkan jawaban secara otomatis.

2.7 *Desain Database*

Proses dalam *architectural spike* juga dapat dijelaskan melalui alur perencanaan dalam bentuk *database* seperti berikut:



Gbr 12. Desain ERD Database

ERD *(Entity Relationship Diagram)* pada desain basis data di atas menggambarkan struktur hubungan antara beberapa tabel yang digunakan dalam sistem *chatbot* untuk layanan *helpdesk* kampus. Tabel student dan staff masing-masing menyimpan informasi mengenai mahasiswa dan staf kampus, seperti ID, nomor unik, nama, tingkat akses, email, *password*, dan salt untuk keamanan. Data ini penting untuk mengidentifikasi dan mengelola akses pengguna ke sistem.

Selanjutnya, tabel *conversation* dan *message* mendokumentasikan percakapan yang terjadi antara pengguna (baik mahasiswa maupun staf) dengan *chatbot*. Setiap percakapan memiliki ID unik dan berisi informasi seperti ID pengguna, tipe pengguna (apakah mahasiswa atau staf), serta waktu mulai percakapan. Pesan-pesan yang terkait dengan percakapan tersebut disimpan dalam tabel *message*, yang mencakup teks dari setiap pesan yang dikirim. Tabel document berfungsi untuk menyimpan dokumen-dokumen yang diunggah oleh pengguna, dengan atribut seperti UUID, nama dokumen, dan *path* file. Sementara itu, tabel question menyimpan data pertanyaan yang diajukan kepada *chatbot*, mencakup detail seperti *prompt*, jawaban dari staf, jawaban dari bot, dan *intent* dari pertanyaan tersebut. Semua elemen ini berintegrasi untuk menciptakan sebuah sistem yang efisien dalam mengelola komunikasi antara pengguna dengan layanan *helpdesk* berbasis *chatbot*.

3) *Release Planning*

Setelah arsitektur dasar dan User stories dipahami, dilakukan perencanaan rilis (“release planning”). Pada tahap ini, User stories diperkirakan (biasanya dalam satuan poin) dan dijadwalkan untuk dirilis. Tahap ini bertujuan untuk menentukan urutan prioritas pengembangan fitur.

4) *Iteration*

Dengan rencana rilis sebagai panduan, tahap iterasi dimulai. Iterasi adalah siklus pengembangan singkat (biasanya 1-3 minggu) di mana User stories dipilih untuk dikembangkan menjadi fitur yang berfungsi. Setiap iterasi melibatkan coding, testing, dan rilis versi kecil dari sistem yang dapat diuji dan dinilai.

5) *Spike*

Jika selama iterasi ditemukan bahwa estimasi untuk User stories terlalu tidak pasti atau sulit, ‘spike’ dapat dilakukan. Spike adalah eksperimen kecil yang dirancang untuk mengurangi risiko dan membantu membuat estimasi yang lebih pasti untuk User stories tersebut di iterasi berikutnya.

6) *Acceptance*

Setelah iterasi selesai, sistem atau fitur yang dikembangkan diperiksa untuk memastikan bahwa mereka memenuhi persyaratan yang ditetapkan dalam User stories. Ini biasanya melibatkan pengujian penerimaan yang dilakukan oleh pengguna atau pemangku kepentingan untuk memverifikasi bahwa fitur bekerja seperti yang diinginkan.

7) *Small Releases*

Setelah fitur diterima, aplikasi dirilis dalam ‘rilis kecil’. Ini memungkinkan pengguna untuk mulai menggunakan fitur baru segera setelah siap, memungkinkan feedback yang cepat dan kemampuan untuk membuat penyesuaian lebih lanjut berdasarkan penggunaan nyata.

8) *Next Iteration*

Setelah rilis kecil, siklus berlanjut dengan iterasi berikutnya. Tim kembali melihat User stories, mungkin menambahkan yang baru atau mengubah yang lama berdasarkan feedback dari pengguna, dan prosesnya

diulangi dengan fokus pada perbaikan berkelanjutan dan peningkatan fitur sistem.

E. Analisis

Pengujian fungsionalitas pada chatbot bertujuan memastikan aplikasi berfungsi sesuai spesifikasi. Metode pengujian yang digunakan adalah blackbox, yang menguji fungsionalitas tanpa memeriksa kode sumber. Langkah awal adalah menentukan skenario pengujian berdasarkan spesifikasi fungsional, mencakup berbagai kasus penggunaan yang mungkin dihadapi pengguna. Setelah itu, data uji disiapkan, termasuk input valid dan invalid, untuk memastikan sistem menangani semua kemungkinan dengan baik.

Setiap skenario diuji menggunakan data uji, dengan mencatat output yang dihasilkan untuk dibandingkan dengan hasil yang diharapkan. Setiap perbedaan diidentifikasi sebagai potensi masalah. Analisis hasil pengujian membantu menentukan apakah aplikasi memenuhi semua persyaratan fungsional. Setelahnya, laporan pengujian disusun untuk merinci hasil, temuan, dan rekomendasi perbaikan. Berdasarkan laporan tersebut, perbaikan dilakukan, dan pengujian ulang memastikan masalah teratasi tanpa menimbulkan cacat baru. Pengujian blackbox ini meningkatkan kualitas dan keandalan chatbot sebelum dirilis untuk pengguna akhir.

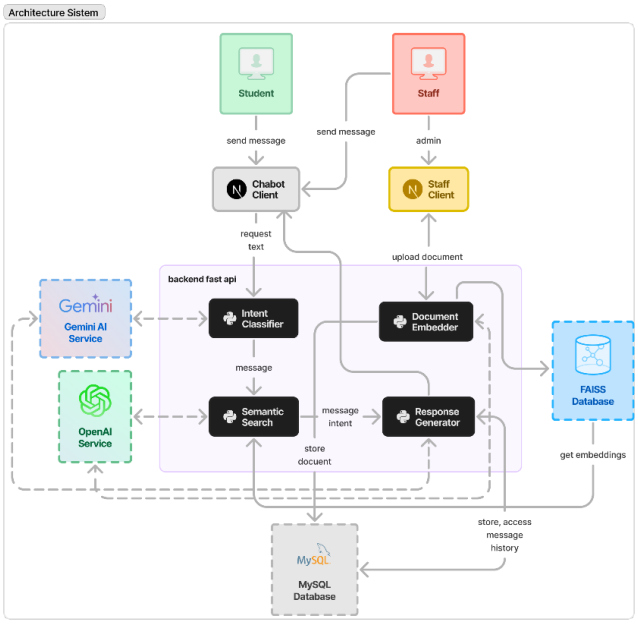
III. PEMBAHASAN

A. Implementasi Sistem chatbot

1) Sistem Backend

Server dikembangkan menggunakan FastAPI, sebuah framework Python yang dirancang untuk membangun aplikasi web dengan performa tinggi. FastAPI dipilih karena ringan dan cepat, menjadikannya pilihan ideal untuk server *chatbot* yang memerlukan waktu respons yang cepat. Keunggulan utama FastAPI adalah kemampuannya untuk mendukung pemrograman asinkron, yang memungkinkan server menangani banyak permintaan secara bersamaan tanpa menunggu setiap permintaan selesai satu per satu. Dengan kata lain, ketika satu pengguna mengirimkan permintaan yang memerlukan waktu lama untuk diproses, pengguna lain dapat mengirimkan permintaan mereka tanpa mengalami penundaan.

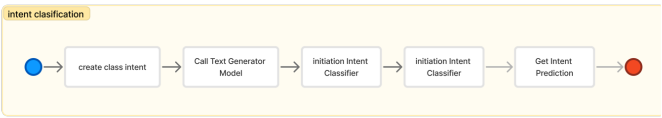
2) Chat Model



Gbr 13. Sistem Arsitektur

Chat model pada sistem ini terdiri dari beberapa komponen utama yang bekerja bersama untuk menghasilkan respons yang relevan dan akurat. Pertama, *Intent Classification* bertugas untuk mengidentifikasi tujuan atau maksud dari setiap permintaan pengguna. Di sini, terdapat empat kategori utama yang diidentifikasi: akademik, *resource, support,* dan *other*. Proses ini menggunakan metode *Retrieval-Augmented Generation* (RAG) dengan *prompt* templating untuk memastikan klasifikasi yang tepat berdasarkan template yang telah ditentukan.

1.1 Klasifikasi Intent



Gbr 14. System Intent Clasification

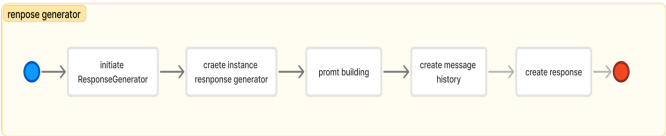
Sistem chatbot ini dirancang untuk mengklasifikasikan berbagai jenis permintaan pengguna menggunakan beberapa komponen utama yang terintegrasi dengan baik. Klasifikasi intent, atau tujuan pesan pengguna, berfungsi untuk mengelompokkan permintaan ke dalam kategori seperti *Academic Administration*, *Resource Service*, *Support*, dan *Other*, yang membantu sistem merespons lebih tepat sasaran.

a. Klasifikasi Intent: Chatbot mengenali intent berdasarkan deskripsi yang terstruktur dalam sebuah dictionary. Setiap intent, seperti *Academic Administration* atau *Resource Service*, mencakup

berbagai jenis pertanyaan, mulai dari pencarian informasi akademik hingga bantuan terkait akun. Dengan pendekatan ini, chatbot dapat mengidentifikasi konteks percakapan secara mendalam.

b. Pemanggilan Model Generatif: Dalam proses klasifikasi intent, sistem menginisialisasi konfigurasi melalui modul konfigurasi yang ditentukan, kemudian memuat model generatif melalui *ModelLoader*. Model ini menghasilkan prediksi intent yang paling sesuai berdasarkan masukan pengguna, memanfaatkan sejarah percakapan dan contoh pesan yang relevan untuk akurasi lebih tinggi.
c. Pencarian Informasi: Sistem ini dilengkapi dengan *Information Retriever* yang menggunakan *FAISS database* sebagai penyimpanan data vektor untuk mempercepat pencarian informasi yang relevan. Ketika sebuah pesan dikirim, model embedding menghasilkan representasi vektor untuk mencocokkan intent dengan data relevan yang disimpan dalam indeks FAISS. Pencarian ini memungkinkan chatbot mengembalikan informasi yang tepat dengan efisiensi tinggi.
d. Pencarian Kemiripan (Similarity Search): Fungsi pencarian kemiripan memanfaatkan indeks FAISS untuk mengukur relevansi antara query pengguna dengan data yang tersedia, mengembalikan hasil yang paling relevan. Sistem juga mendukung pencarian asinkron untuk meningkatkan efisiensi dan kinerja secara keseluruhan, terutama dalam skenario yang melibatkan banyak pengguna secara bersamaan.

1.2 Response Generator



Gbr 15. System Response Generator

Setelah chatbot menerima informasi dari pengguna, *Response Generator* bertugas untuk menghasilkan respons teks menggunakan model generatif, dalam hal ini model Gemini. Dengan sistem streaming, chatbot dapat memberikan jawaban secara real-time, sehingga pengguna mendapatkan respons secara instan.

a. Inisialisasi Response Generator: *Response Generator* disiapkan dengan menginisialisasi template *prompt* dan model generator teks. *Prompt* yang dibuat memungkinkan sistem merespons secara relevan sesuai konteks percakapan. Dalam prosesnya, *Response Generator* juga memungkinkan pengguna untuk mendapatkan jawaban yang tepat dan bersifat kontekstual.
b. Penyusunan Prompt dan Riwayat Pesan: Untuk menghasilkan respons yang relevan, *Response Generator* menyusun *prompt* dengan menyertakan contoh pesan atau riwayat interaksi. Ini membantu chatbot memahami konteks penuh dari percakapan dan memastikan jawaban yang diberikan sesuai dengan pesan sebelumnya. Sistem ini menyimpan sejarah percakapan untuk menyediakan informasi tambahan saat merespons, terutama dalam interaksi yang lebih panjang.
c. Pembuatan Respons Asinkron: Dengan pendekatan asinkron, chatbot dapat menangani beberapa permintaan secara bersamaan. Ketika ada permintaan untuk respons, sistem akan menghasilkan iterator asinkron yang memproses pesan dari pengguna dan riwayat percakapan, meningkatkan efisiensi dan mengurangi waktu tunggu dalam interaksi.

1.3 Pengolahan dan Penyimpanan Dokumen dengan Document Embedder

Document Embedder berperan dalam mengolah dokumen seperti file teks atau PDF menjadi representasi vektor, yang kemudian disimpan dalam penyimpanan FAISS untuk memungkinkan pencarian berbasis kesamaan. Hal ini memungkinkan chatbot untuk mengakses informasi yang relevan secara cepat saat dibutuhkan dalam percakapan.

a. Konversi Dokumen ke Vektor: *Document Embedder* mengubah dokumen menjadi vektor yang dapat digunakan untuk pencarian. Model *embedding* menghasilkan representasi vektor dari isi dokumen, sementara *splitter* teks membagi dokumen menjadi bagian-bagian kecil agar lebih mudah diolah dan disimpan.
b. Penyimpanan di FAISS Database: Setelah dokumen diubah menjadi vektor, setiap bagian disimpan dalam indeks FAISS sesuai kategori dokumen. Proses penyimpanan ini memungkinkan pencarian cepat berdasarkan kategori dokumen sehingga respons yang diberikan oleh chatbot selalu relevan dengan pertanyaan pengguna.
c. Memuat dan Memecah Dokumen: *Document Embedder* mampu memuat dokumen berdasarkan tipe seperti teks atau PDF, lalu membaginya ke dalam potongan kecil. Hal ini dilakukan untuk memungkinkan pemrosesan yang lebih efisien, di mana potongan-potongan dokumen dapat dicari dan diambil secara cepat berdasarkan relevansi dengan permintaan pengguna.

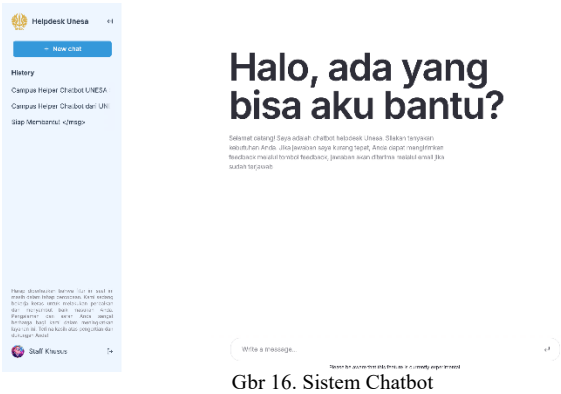
Dengan kombinasi ini, chatbot dapat dengan mudah menemukan informasi yang relevan dari dokumen yang

telah disimpan, menghasilkan respons yang akurat dan responsif bagi pengguna dalam konteks layanan helpdesk.

1.4 Tampilan Antarmuka Pengguna

Antarmuka pengguna sistem chatbot ini dibangun dengan framework Next.js yang memungkinkan performa optimal dan responsivitas tinggi, cocok untuk interaksi intensif dengan pengguna. Next.js mendukung *server-side rendering* (SSR) dan *static site generation* (SSG), yang mempercepat waktu muat dan meningkatkan SEO. Fitur seperti *pre-fetching* dan *code-splitting* memastikan halaman hanya memuat elemen yang diperlukan, memberikan pengalaman pengguna yang lancar baik di desktop maupun perangkat mobile.

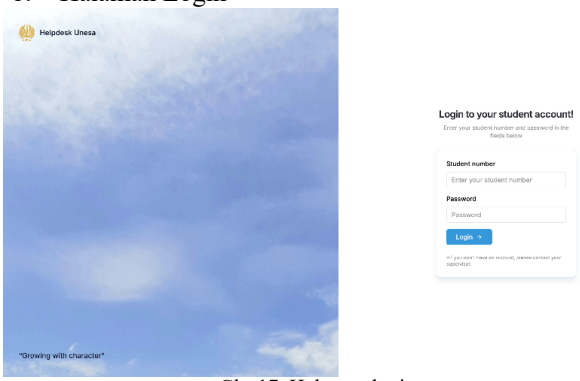
a. Antarmuka Sistem Chatbot



Gbr 16. Sistem Chatbot

Pengguna dapat memulai obrolan baru, melihat riwayat percakapan, atau keluar (logout). Tampilan awal memberikan instruksi singkat, serta opsi untuk memberikan umpan balik jika respons chatbot dirasa kurang tepat. Umpan balik ini akan direspon melalui email dan membantu peningkatan layanan.

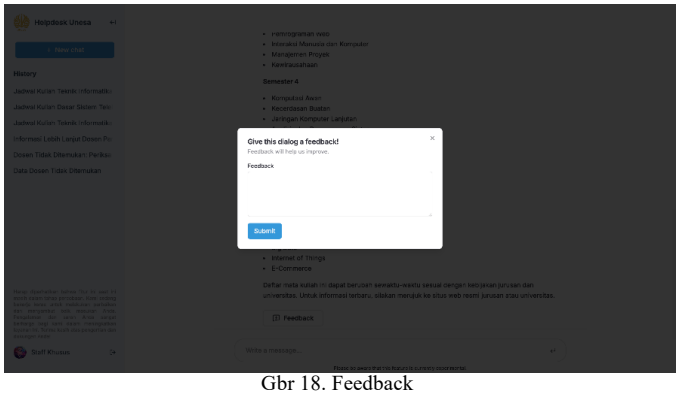
b. Halaman Login



Gbr 17. Halaman login

Pengguna memasukkan nomor mahasiswa dan kata sandi untuk mengakses akun. Jika belum memiliki akun, pengguna diarahkan untuk menghubungi supervisor.

c. Antarmuka Feedback



Gbr 18. Feedback

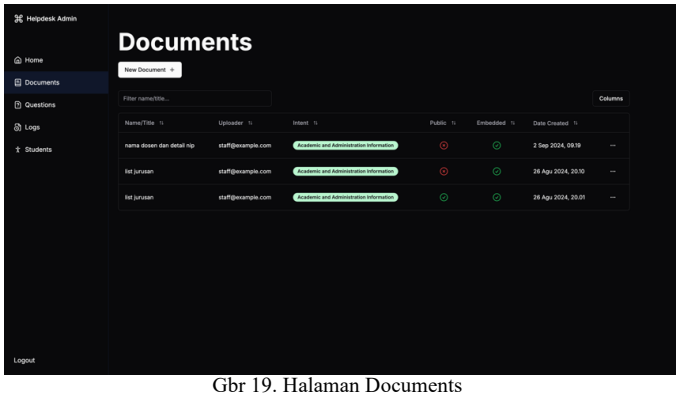
Jendela pop-up memungkinkan pengguna memberikan umpan balik jika tidak puas dengan jawaban chatbot. Feedback ini penting untuk pengembangan lebih lanjut dan meningkatkan kualitas respons chatbot.

1.5 Tampilan Antarmuka Staff

Sistem staff dan admin dibangun menggunakan Next.js dengan repositori terpisah, memastikan akses dan fungsionalitas user dan admin dikelola secara mandiri. Pemisahan ini memungkinkan pengembangan yang fleksibel dan efisien untuk setiap sistem.

Pada Admin Dashboard, menggunakan template *shadcn* yang menyediakan komponen modern untuk antarmuka yang responsif, mempermudah admin dalam mengelola data, analisis, dan akses informasi penting.

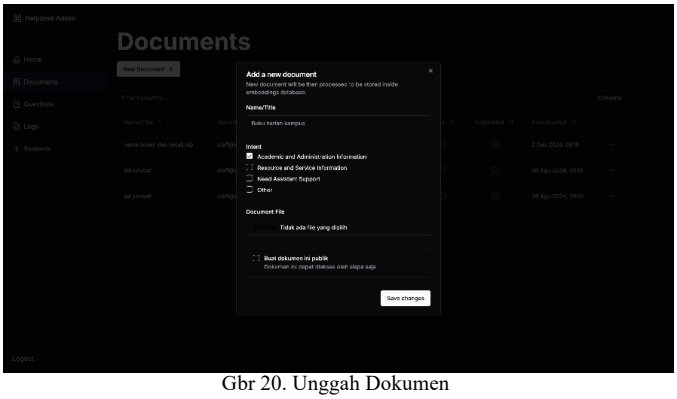
a. Antarmuka Halaman Dokumen



Gbr 19. Halaman Documents

Menyediakan fitur "New Document" untuk menambah dokumen, kolom pencarian, serta tabel yang menampilkan dokumen dengan informasi seperti pengunggah, status, dan tanggal pembuatan.

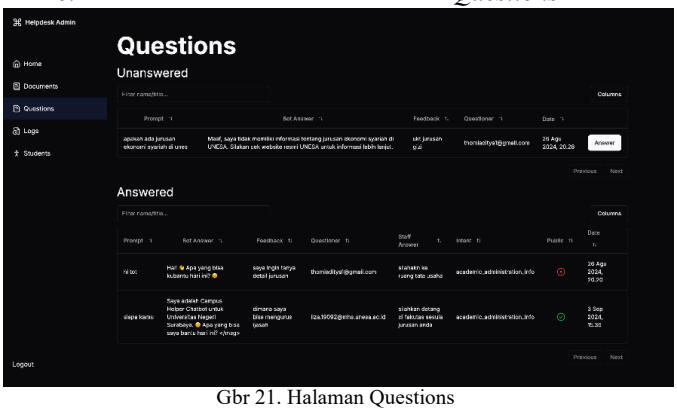
b. Antarmuka Unggah Dokumen



Gbr 20. Unggah Dokumen

Pop-up "Add a new document" memungkinkan admin menambahkan dokumen baru dengan judul, tujuan (*intent*), file, dan opsi akses publik.

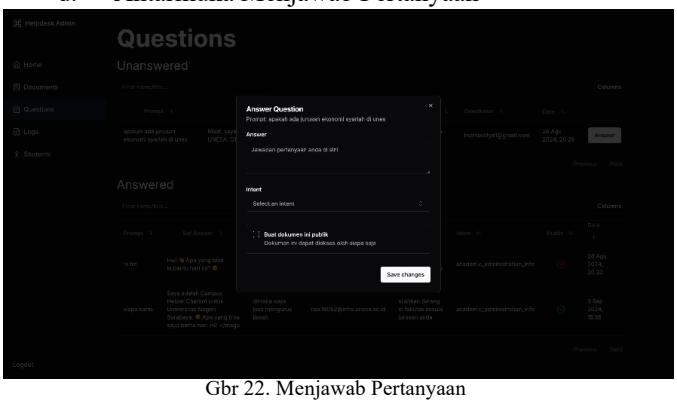
c. Antarmuka Halaman Feedback *Questions*



Gbr 21. Halaman Questions

Menampilkan pertanyaan dalam dua kategori, "Unanswered" dan "Answered," dengan detail seperti jawaban bot, umpan balik, dan nama penanya.

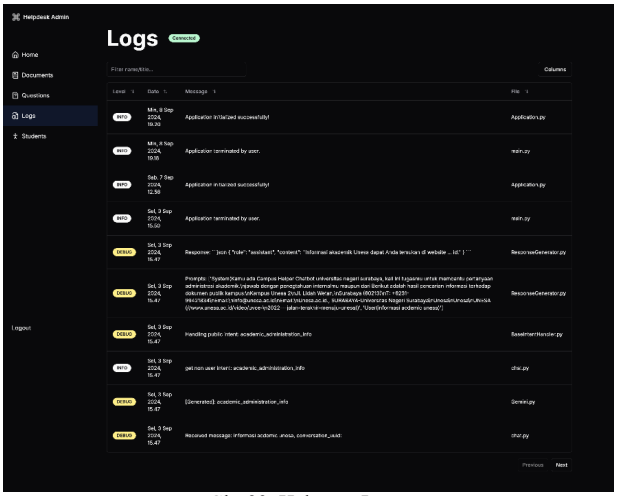
d. Antarmuka Menjawab Pertanyaan



Gbr 22. Menjawab Pertanyaan

Modal "Answer Question" untuk memasukkan jawaban atas pertanyaan belum terjawab, dengan pilihan untuk memilih tujuan dan akses publik.

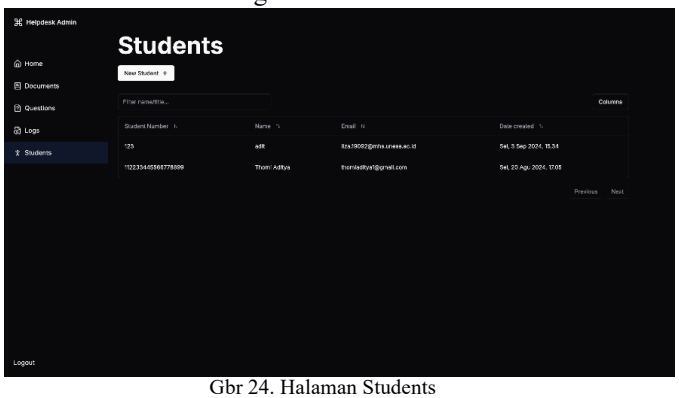
e. Antarmuka Logs



Gbr 23. Halaman Logs

Berisi tabel log aktivitas sistem dengan level log, tanggal, waktu, dan pesan log, membantu dalam pemantauan operasional.

f. Antarmuka Pengelola Siswa



Gbr 24. Halaman Students

Memungkinkan admin menambah mahasiswa baru, melakukan pencarian, dan melihat daftar mahasiswa dengan nomor mahasiswa, nama, email, dan tanggal pembuatan akun.

B. Pengujian Sistem

1) Pengujian Sistem

1.1 Unit Test

Unit testing adalah proses pengujian kode program secara terpisah pada unit terkecil *Unit testing* digunakan untuk mengidentifikasi dan memperbaiki kesalahan pada tingkat paling awal dalam siklus pengembangan,

Adapun tahapan-tahapan unit testing adalah sebagai berikut:

a. Test Gemini

TestGemini digunakan untuk menguji berbagai fungsi dari model Gemini. Tes ini mencakup beberapa skenario pengujian, termasuk pengujian terhadap kemampuan model dalam menghasilkan teks, memeriksa

apakah metode streaming teks mengembalikan generator yang sesuai, dan memastikan bahwa pesan dalam Gemini dapat di-cast menjadi string.

TABLE I
HASIL TEST GEMINI

No	Skenario Pengujian	Hasil
1	*Generate Text Integration*	*Success*
2	*Streaming Text Returns Generator Integration*	*Success*
3	*Message Template Casts to Str*	*Success*

Hasil tes menunjukkan bahwa fungsi generate text pada Gemini dapat berjalan dengan baik dengan menghasilkan teks yang valid dan dapat di-cast menjadi string, serta metode stream juga berfungsi dengan baik dengan mengembalikan potongan teks sebagai generator yang diharapkan.

b. Test Document Embedder

Test Document Embedder bertujuan untuk menguji berbagai aspek dari *Document Embedder.* serta pengujian penyimpanan dokumen ke dalam *vectorstore,* termasuk dokumen publik.

TABLE II
HASIL TEST DOCUMENT EMBEDDER

No	Skenario Pengujian	Hasil
1	*Instance Initialization*	*Success*
2	*Instance Singleton*	*Success*
3	*Save Document to Vectorstore*	*Success*
4	*Save Public Document to Vectorstore*	*Success*

Hasil tes menunjukkan bahwa DocumentEmbedder dapat berjalan dengan baik dengan berhasil menginisialisasi instance yang benar, berfungsi sebagai singleton, dan menyimpan dokumen ke dalam vectorstore dengan struktur direktori FAISS yang valid.

c. Information Retriever

Information Retriever digunakan untuk memastikan bahwa *instance* dari *Information Retriever* berfungsi dengan benar dan dapat mengembalikan respons yang valid. Tes ini mencakup beberapa skenario pengujian, seperti inisialisasi *instance*, serta pengujian terhadap metode *retrieve_async* dan *retrieve_public_async* untuk memastikan bahwa respons yang dikembalikan sesuai dengan *input* yang diberikan, baik dalam hal format maupun konten.

TABLE III
HASIL INFORMATION RETRIEVER

No	Skenario Pengujian	Hasil
1	*Instance Information Retriever*	*Success*
2	*Retrieve Returns Valid Response*	*Success*
3	*Retrieve Public Returns Valid Response*	*Success*

Hasil tes menunjukkan bahwa InformationRetriever dapat berjalan dengan baik dengan menginisialisasi instance yang benar dan mengembalikan respons yang valid dari metode retrieve_async dan retrieve_public_async.

d. Intent Classifier

Intent Classifier bertujuan untuk menguji fungsi dari *Intent* Classifier dalam mengklasifikasikan niat (*intent*). Pengujian ini mencakup inisialisasi *instance Intent Classifier*, memastikan bahwa metode list mengembalikan daftar, dan menguji apakah metode *classify* dapat mengembalikan *intent* yang sesuai dengan *input* yang diberikan.

TABLE IV
HASIL INTENT CLASSIFIER

No	Skenario Pengujian	Hasil
1	*Intent Classifier Instance*	*Success*
2	*Intent List Returns List*	*Success*
3	*Classify Returns Intent*	*Success*

Hasil tes menunjukkan bahwa IntentClassifier dapat berjalan dengan baik dengan memastikan instance terinisialisasi dengan benar, mengembalikan list intents yang sesuai, dan mengklasifikasikan intent berdasarkan input dengan benar.

e. Model Loader

Model Loader digunakan untuk menguji proses pemuatan model. Skenario pengujian mencakup apakah metode *load_model* mengembalikan *instance* yang sesuai dari *Text Generator* dan Gemini, serta memastikan bahwa kesalahan yang tepat dilempar jika model yang tidak ada dicoba untuk dimuat.

TABLE V
HASIL MODEL LOADER

No	Skenario Pengujian	Hasil
1	*Load Model Returns Text Generator*	*Success*
2	*Load Nonexistent Model*	*Success*

Hasil tes menunjukkan bahwa ModelLoader dapat berjalan dengan baik dengan berhasil mengembalikan instance dari TextGenerator saat model yang valid dimuat dan menangani kesalahan dengan tepat ketika model yang tidak ada dimuat.

f. Prompt Manager

Prompt Manager bertujuan untuk menguji fungsi manajemen *prompt*, khususnya memastikan bahwa metode *get_prompt* mengembalikan string yang sesuai ketika dipanggil dengan parameter yang valid.

TABLE VI
HASIL PROMPT MANAGER

No	Skenario Pengujian	Hasil
1	*Prompt Manager Get Prompt*	*Success*

Hasil tes menunjukkan bahwa PromptManager dapat berjalan dengan baik dengan memastikan metode get_prompt mengembalikan string yang valid sesuai dengan parameter yang diberikan.

g. Response Generator

Response Generator digunakan untuk menguji apakah metode response_async dapat mengembalikan potongan teks yang relevan. Tes ini memastikan bahwa hasil yang dikembalikan berupa potongan teks yang berisi kata kunci yang sesuai dengan konteks *input*.

TABLE VII
HASIL RESPONSE GENERATOR

No	Skenario Pengujian	Hasil
1	*Response Generator Returns Chunks*	*Success*

Hasil tes menunjukkan bahwa ResponseGenerator dapat berjalan dengan baik dengan mengembalikan potongan teks yang berisi kata kunci yang relevan melalui metode *response_async.*

h. Title Generator

Title Generator bertujuan untuk menguji fungsi pembuatan judul. Pengujian mencakup memastikan bahwa Title Generator adalah singleton dan bahwa metode generate_title mengembalikan string yang relevan dengan *input* yang diberikan.

TABLE VIII
HASIL TITLE GENERATOR

No	Skenario Pengujian	Hasil
1	*Title Generator Singleton*	*Success*
2	*Generate Title*	*Success*

Hasil tes menunjukkan bahwa TitleGenerator dapat berjalan dengan baik dengan berfungsi sebagai singleton dan mengembalikan judul yang relevan berdasarkan input melalui metode *generate_title.*

i. Test Utils

Test Utils berfungsi untuk menguji utilitas proyek, khususnya fungsi *project_path*. Tes ini memastikan bahwa metode *project_path* mengembalikan jalur yang benar sesuai dengan parameter yang diberikan.

TABLE IX
HASIL TEST UTILS

No	Skenario Pengujian	Hasil
1	*Project Path Util*	*Success*

Hasil tes menunjukkan bahwa TestUtils dapat berjalan dengan baik dengan fungsi project_path yang mengembalikan path yang sesuai dengan yang diharapkan.

1.2 Integration Test

Integration testing adalah proses pengujian yang dilakukan setelah unit testing, di mana berbagai komponen atau unit yang telah diuji secara individual diintegrasikan dan diuji secara bersama-sama. Dalam skripsi ini

TABLE X
HASIL INTEGRATION TEST

No	Skenario Pengujian	Hasil
1	Pengujian Login *Helpdesk*	*Success*
2	Pengujian Respons *Chatbot* untuk Pertanyaan Umum	*Success*
3	Pengujian Respons *Chatbot* untuk Pertanyaan Khusus	*Success*
4	Pengujian Formulir Ketidakpuasan	*Success*
5	Pengujian Notifikasi Email	*Success*
6	Pengujian Kinerja *Chatbot*	*Success*
7	Pengujian Integrasi *Knowledge Base*	*Success*
8	Pengujian Pembaruan *Knowledge Base* oleh Staf *Helpdesk*	*Success*

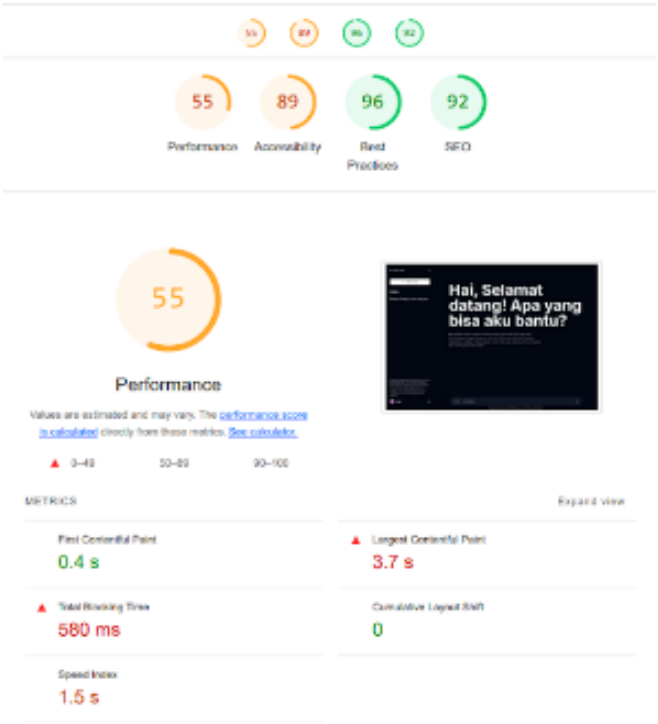
Hasil pengujian menunjukkan bahwa berbagai komponen dalam sistem *helpdesk* berfungsi dengan baik. Staf *helpdesk* dapat login ke sistem dengan sukses dan diarahkan ke dashboard, sementara *chatbot* mampu memberikan jawaban yang relevan dan akurat baik untuk

pertanyaan umum maupun khusus berdasarkan data di *knowledge base.*

2) Pengujian Performa Aplikasi

2.1 Light House

Lighthouse adalah alat otomatis yang digunakan untuk meningkatkan kualitas halaman



Gbr 25. Hasil Test Lighthouse

Hasil pengujian Lighthouse menunjukkan skor kinerja situs web sebesar 55, yang menandakan perlunya optimasi pada waktu eksekusi JavaScript, pekerjaan di main-thread, dan sumber daya yang menghalangi rendering. Untuk aksesibilitas, situs memperoleh skor tinggi yaitu 89, menandakan akses yang baik bagi berbagai pengguna. Praktik terbaik meraih skor 96, mencerminkan kepatuhan terhadap standar keamanan dan teknologi terbaru. SEO juga cukup baik dengan skor 92, menunjukkan elemen optimisasi yang sudah diterapkan. Fokus perbaikan perlu diarahkan pada kinerja situs agar waktu muat lebih optimal, meningkatkan pengalaman dan kepuasan pengguna.

C. Analisis Hasil Pengujian

1) Analisis Hasil Unit Test

Hasil unit test menunjukkan semua fungsi dan komponen bekerja sesuai skenario yang dirancang, tanpa ditemukan bug atau kesalahan. Hal ini menandakan bahwa komponen-komponen sistem berfungsi baik dan sesuai spesifikasi, menunjukkan kualitas kode yang modular dan terstruktur.

2) Analisis hasil integration test

Integration test menunjukkan bahwa semua modul terintegrasi dengan baik dan saling berinteraksi tanpa konflik. Setiap skenario pengujian berjalan lancar, menunjukkan bahwa sistem telah siap untuk tahap pengujian atau deployment berikutnya..

3) Kesimpulan hasil penelitian

Pengujian menunjukkan bahwa sistem helpdesk berbasis chatbot ini memenuhi tujuan penelitian, yaitu mendukung layanan administratif kampus. Unit test dan integration test mengonfirmasi bahwa setiap komponen bekerja baik secara individual dan dalam integrasi, sesuai harapan dan spesifikasi awal.

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Penelitian ini berhasil mengembangkan sistem chatbot berbasis Retrieval-Augmented Generation (RAG) untuk mendukung layanan administratif kampus. Sistem ini memudahkan staf dan mahasiswa dalam mengakses informasi serta memperoleh bantuan yang relevan dengan cepat. Pengujian menunjukkan bahwa chatbot memenuhi kriteria fungsional dengan kinerja yang stabil dan responsif di lingkungan nyata. Dengan demikian, sistem ini efektif dan efisien dalam meningkatkan layanan administratif kampus, memenuhi tujuan utama penelitian.

B. Saran

Untuk pengembangan lebih lanjut, disarankan agar sistem chatbot segera di-deploy ke server agar dapat diuji dalam kondisi operasional sebenarnya, memungkinkan evaluasi performa dan identifikasi potensi perbaikan yang mungkin terlewatkan saat pengujian lokal. Selain itu, peningkatan model chatbot dengan versi yang lebih canggih dapat meningkatkan keakuratan respons. Penyesuaian konfigurasi, terutama pada prompt, juga diusulkan untuk meningkatkan kualitas interaksi. Langkah-langkah ini diharapkan membuat chatbot lebih responsif dan mampu memenuhi kebutuhan pengguna dengan lebih baik.

REFERENSI

[1] Ardiansyah, R., Marya, D., & Novianti, A. (2023). Penggunaan metode string matching pada sistem informasi mahasiswa Polinema dengan chatbot.

[2] Ichsani, Y. (2017). Evaluasi performa usability dan kepuasan penggunaan situs-situs web perguruan tinggi negeri di Indonesia.

[3] Mendoza, S., Hernández-León, M., Sánchez-Adame, L. M., Rodríguez, J., Decouchant, D., & Meneses-Viveros, A. (2020). Supporting Student-Teacher Interaction Through a Chatbot. In Proceedings of the 2020 International Conference on Artificial Intelligence and Education Technology.

[4] Hardi, R., Pee, A. N. C., Abdullah, M., Pitogo, V. P., Pribadi, A. S., & Rusdi, J. F. (2022). Academic Smart Chatbot Support: An Emerging Artificial Intelligence. In Proceedings of the 2022 International Conference on Educational Technology.
[5] Al-Jedaie, R., Al-Hindy, R., Al-Onazi, H., Kariri, E., & Masmoudi, F. (2022). Chatbot for Academic Advising. In Proceedings of the 2022 International Conference on Advanced Learning Technologies.
[6] Ghadge, M., Dhumale, A., Daki, G., Kolekar, U., & Shaikh, N. (2020). Chatbot for Efficient Utilization of College Laboratories. In Proceedings of the 2020 International Conference on Information Technology.
[7] Setiyani, L. (2023). Increasing the Effectiveness of Higher Education Academic Services Using a Chatbot. In Proceedings of the 2023 International Conference on Educational Technology.
[8] Alqaidi, S., Alharbi, W. S., & Almatrafi, O. (2021). A Support System for College Students: A Case Study of a Chatbot Application. In Proceedings of the 2021 International Conference on Computer Science and Education Technology.
[9] Fang, Z. (2021). *Writing a Literature Review*. Demystifying Academic Writing.
[10] Alfantoukh, L., & Durresi, A. (2014). *Techniques for Collecting Data in Social Networks*. 2014 17th International Conference on Network-Based Information Systems, 336-341.