Jurnal Minfo Polgan (JMP)
Volume 15, Nomor 3, September-Desember 2026
DOI : https://doi.org/10.33395/jmp.v15i3.16692
e-ISSN : 2797-3298
p-ISSN : 2089-9424

Rancang Bangun *Chatbot Customer Service Multi-Tenant* Berbasis *Large Language* Model Menggunakan *Context Stuffing*

[1]Ridho Rifaldho, [2] Diana Novita
[1*,2] Program Studi Teknik Informatika, Fakultas Ilmu Komputer
Universitas Esa Unggul, Indonesia
*Korespondensi: ridhorifaldho@gmail.com

Submit : 11 Jul | **Diterima** : 03 Agust 2026 | **Terbit** : 07 Sept 2026

ABSTRACT

The advancement of Artificial Intelligence (AI), particularly Large Language Models (LLMs), has transformed customer service into a more automated, adaptive, and responsive system. However, LLM-based chatbots still face several challenges, including hallucination, limited capability to serve multiple organizations simultaneously, and difficulties in website integration. This study aims to design and develop an embeddable multi-tenant customer service chatbot based on a Large Language Model. The system was developed using the Go programming language, Fiber framework, PostgreSQL database, and a JavaScript widget that can be embedded into company websites using a single line of code. The Research and Development (R&D) approach was adopted, while the Context Stuffing method ensured that chatbot responses were generated only from each tenant's official knowledge base, thereby minimizing hallucination. The evaluation included black-box testing, response accuracy measurement, hallucination assessment, response time analysis, and tenant data isolation testing. The experimental results demonstrated that the chatbot achieved 100% response accuracy for in-domain questions, produced zero hallucination for out-of-domain questions, maintained an average response time of approximately four seconds, and successfully preserved complete data isolation among tenants. Furthermore, the chatbot widget was easily embedded into company websites with minimal configuration. These findings indicate that the proposed chatbot provides an effective, secure, and scalable AI-based customer service solution suitable for multiple organizations.

***Keywords**: Artificial Intelligence, Customer Service, Large Language Model, Multi-Tenant, Context Stuffing.*

ABSTRAK

Perkembangan *Artificial Intelligence* (AI), khususnya *Large Language Model (LLM)*, telah mendorong transformasi layanan pelanggan menuju sistem yang lebih otomatis, adaptif, dan responsif. Meskipun demikian, implementasi chatbot berbasis LLM masih menghadapi beberapa kendala, seperti kecenderungan menghasilkan informasi yang tidak sesuai fakta (hallucination), keterbatasan dalam melayani banyak organisasi secara bersamaan, serta kompleksitas integrasi pada website perusahaan. Penelitian ini bertujuan merancang dan membangun chatbot customer service berbasis *Large Language* Model dengan arsitektur multi-tenant yang dapat disematkan (embeddable) pada berbagai website perusahaan. Sistem dikembangkan menggunakan bahasa pemrograman Golang dengan framework Fiber, basis data PostgreSQL, dan widget JavaScript yang dapat diintegrasikan melalui satu baris kode. Pendekatan *Research and Development (R&D)* digunakan sebagai metode penelitian, sedangkan metode *Context Stuffing* diterapkan untuk memastikan chatbot hanya menghasilkan jawaban berdasarkan pengetahuan resmi milik masing-masing tenant sehingga dapat meminimalkan halusinasi. Evaluasi dilakukan menggunakan pengujian black box, pengukuran akurasi jawaban, tingkat halusinasi, waktu respons, dan pengujian isolasi data antar-tenant. Hasil penelitian menunjukkan bahwa sistem mampu mencapai akurasi jawaban sebesar 100% untuk pertanyaan yang berada dalam ruang lingkup basis pengetahuan, tingkat halusinasi sebesar 0% untuk pertanyaan di luar cakupan informasi, rata-rata waktu respons sekitar empat detik, serta mampu menjaga pemisahan data antar-tenant secara konsisten. Selain itu, widget chatbot berhasil diimplementasikan pada website perusahaan hanya dengan satu baris kode.

Hasil tersebut menunjukkan bahwa sistem layak digunakan sebagai solusi layanan pelanggan otomatis berbasis AI yang efisien, aman, dan mudah diimplementasikan pada berbagai organisasi.

Kata Kunci: *Artificial Intelligence*, Chatbot, *Context Stuffing*, *Large Language* Model, *Multi-Tenant.*

PENDAHULUAN

Layanan pelanggan merupakan salah satu aspek penting dalam membangun hubungan antara perusahaan dan pelanggan. Kualitas layanan yang diberikan akan memengaruhi tingkat kepuasan, loyalitas, serta citra perusahaan. Dalam praktiknya, sebagian besar aktivitas customer service masih didominasi oleh pertanyaan yang bersifat berulang, seperti informasi produk, jam operasional, prosedur pemesanan, hingga kontak perusahaan. Kondisi tersebut menyebabkan staf customer service menghabiskan banyak waktu untuk menjawab pertanyaan yang sama sehingga efisiensi pelayanan menjadi rendah. Selain itu, keterbatasan jam operasional menyebabkan pelanggan yang membutuhkan informasi di luar jam kerja harus menunggu hingga petugas tersedia. Kondisi ini mendorong perlunya solusi layanan pelanggan yang mampu memberikan respons secara otomatis selama dua puluh empat jam. Perkembangan *Artificial Intelligence* (AI), khususnya *Large Language Model (LLM)*, menghadirkan paradigma baru dalam pengembangan chatbot. Berbeda dengan chatbot berbasis rule-based yang hanya mampu menjawab pola pertanyaan tertentu, LLM mampu memahami bahasa alami dan menghasilkan respons yang lebih fleksibel sesuai konteks percakapan. Teknologi ini memberikan peluang bagi perusahaan untuk meningkatkan kualitas layanan pelanggan secara signifikan melalui otomatisasi percakapan yang lebih natural.

Meskipun demikian, penggunaan LLM dalam layanan pelanggan masih menghadapi berbagai tantangan. Salah satu permasalahan utama adalah hallucination, yaitu kondisi ketika model menghasilkan jawaban yang tidak didasarkan pada fakta atau informasi resmi perusahaan. Dalam konteks customer service, kondisi tersebut dapat menyebabkan penyampaian informasi yang keliru kepada pelanggan sehingga menurunkan tingkat kepercayaan terhadap perusahaan. Oleh karena itu, diperlukan mekanisme yang mampu membatasi ruang pengetahuan chatbot agar hanya menghasilkan jawaban berdasarkan informasi yang telah diverifikasi. Permasalahan lain adalah kebutuhan implementasi chatbot pada banyak perusahaan secara bersamaan. Pengembangan chatbot secara terpisah untuk setiap organisasi membutuhkan biaya, waktu, dan sumber daya yang relatif besar. Pendekatan multi-tenant menawarkan solusi melalui penggunaan satu aplikasi yang dapat melayani banyak perusahaan dengan tetap menjaga pemisahan data setiap tenant. Selain itu, sistem juga perlu menyediakan mekanisme integrasi yang sederhana sehingga perusahaan dapat memasang chatbot pada website tanpa memerlukan proses pengembangan yang kompleks.

Berdasarkan permasalahan tersebut, penelitian ini mengembangkan chatbot customer service berbasis *Large Language* Model menggunakan arsitektur multi-tenant dan metode *Context Stuffing*. Pendekatan *Context Stuffing* digunakan untuk memastikan model hanya memberikan jawaban berdasarkan basis pengetahuan resmi milik masing-masing tenant sehingga mampu meminimalkan terjadinya halusinasi. Selain itu, sistem dirancang dalam bentuk widget JavaScript yang dapat disematkan ke berbagai website hanya melalui satu baris kode sehingga memudahkan proses implementasi pada berbagai organisasi. Penelitian ini bertujuan menghasilkan sistem chatbot customer service yang mampu memberikan layanan otomatis selama dua puluh empat jam, menjaga isolasi data antar-tenant, meminimalkan hallucination melalui *Context Stuffing*, serta menyediakan mekanisme integrasi yang sederhana pada website perusahaan. Kontribusi penelitian ini diharapkan dapat memberikan alternatif implementasi layanan pelanggan berbasis AI yang efisien, aman, dan mudah diadopsi oleh berbagai organisasi.

METODE PENELITIAN

Desain Penelitian

Penelitian ini menggunakan metode *Research and Development (R&D)* dengan tujuan menghasilkan sebuah produk perangkat lunak berupa chatbot customer service berbasis *Large Language Model (LLM)* yang mampu melayani banyak perusahaan (multi-tenant) dan dapat diintegrasikan ke berbagai website melalui embeddable widget. Pendekatan R&D dipilih karena

penelitian tidak hanya berfokus pada analisis permasalahan, tetapi juga menghasilkan prototipe sistem yang kemudian diuji untuk mengetahui tingkat fungsionalitas, akurasi, keamanan, dan performanya.

Tahapan penelitian dimulai dari identifikasi kebutuhan pengguna, studi literatur, analisis kebutuhan sistem, perancangan arsitektur, implementasi perangkat lunak, pengujian sistem, hingga evaluasi hasil implementasi. Pendekatan ini memungkinkan sistem dikembangkan secara sistematis sesuai kebutuhan pengguna dan tujuan penelitian.

Tahapan Penelitian

Tahapan penelitian terdiri atas enam langkah utama sebagai berikut.

1. Identifikasi Masalah

Tahap awal dilakukan melalui identifikasi berbagai permasalahan yang terjadi pada layanan customer service perusahaan, antara lain tingginya jumlah pertanyaan berulang, keterbatasan jam pelayanan, risiko penyampaian informasi yang tidak konsisten, serta tingginya biaya implementasi chatbot apabila setiap perusahaan menggunakan sistem yang berbeda.

2. Studi Literatur

Penelitian dilanjutkan dengan pengumpulan referensi mengenai customer service, chatbot, *Artificial Intelligence*, *Large Language* Model, *Context Stuffing*, arsitektur multi-tenant, Clean Architecture, PostgreSQL, Golang, Framework Fiber, serta berbagai penelitian terdahulu yang relevan. Studi literatur digunakan sebagai dasar dalam menentukan pendekatan pengembangan sistem.

3. Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan fungsional dan non-fungsional sistem.

Kebutuhan fungsional meliputi:

- pengelolaan tenant;
- pengelolaan basis pengetahuan perusahaan;
- pengelolaan akun administrator;
- proses percakapan chatbot;
- pencatatan histori percakapan;
- konfigurasi widget chatbot;
- autentikasi pengguna.

Sementara itu kebutuhan non-fungsional mencakup:

- keamanan sistem;
- performa respons chatbot;
- kemudahan integrasi;
- skalabilitas sistem;
- kompatibilitas lintas platform.

4. Perancangan Sistem

Tahapan ini menghasilkan rancangan arsitektur perangkat lunak menggunakan pendekatan Clean Architecture sehingga setiap lapisan aplikasi memiliki tanggung jawab yang jelas dan mudah dipelihara.

Perancangan sistem meliputi:

- arsitektur sistem;
- use case diagram;
- activity diagram;
- sequence diagram;
- entity relationship diagram (ERD);
- class diagram;
- flowchart mekanisme anti-halusinasi;
- rancangan antarmuka dashboard;
- rancangan widget chatbot.

Selain itu dilakukan perancangan mekanisme otorisasi berbasis tenant sehingga setiap perusahaan hanya dapat mengakses data miliknya sendiri.

5. Implementasi Sistem

Implementasi dilakukan menggunakan beberapa teknologi utama, yaitu:

Tabel 1 Implementasi Sistem

Komponen	Teknologi
Backend	*Golang*
Framework	*Fiber*
Database	*PostgreSQL*
Widget	*JavaScript*
API LLM	*OpenRouter*
Arsitektur	*Clean Architecture*
Multi-Tenant	*Shared Schema*

Backend bertugas menangani autentikasi, pengelolaan tenant, penyimpanan basis pengetahuan, pengiriman prompt ke *Large Language* Model, serta pengelolaan histori percakapan.
Frontend administrator digunakan untuk mengelola seluruh konfigurasi chatbot, sedangkan widget JavaScript berfungsi sebagai antarmuka yang dapat dipasang pada website perusahaan hanya menggunakan satu baris kode.

6. Arsitektur Sistem

Sistem dikembangkan menggunakan arsitektur *client-server.*
Komponen utama sistem terdiri atas:

- Dashboard Administrator

Digunakan untuk mengelola tenant, basis pengetahuan, akun administrator, konfigurasi widget, dan histori percakapan.

- Widget Chatbot

Widget JavaScript yang disisipkan pada website perusahaan sebagai media interaksi antara pelanggan dengan chatbot.

- Backend API

Backend dibangun menggunakan Golang dan Framework Fiber yang menangani autentikasi, pemrosesan prompt, pemanggilan API LLM, penyimpanan data, serta pengelolaan multi-tenant.

- Database PostgreSQL

Digunakan untuk menyimpan data tenant, administrator, basis pengetahuan, konfigurasi chatbot, serta histori percakapan.

- *Large Language* Model

Model bahasa diakses melalui layanan OpenRouter untuk menghasilkan respons berdasarkan konteks yang telah disusun oleh sistem.

7. Implementasi Context Stuffing

Mekanisme *Context Stuffing* diterapkan sebelum prompt dikirimkan ke *Large Language* Model.
Prosesnya meliputi:

- sistem mengidentifikasi tenant berdasarkan widget yang digunakan;
- sistem mengambil seluruh basis pengetahuan milik tenant;
- seluruh informasi perusahaan digabungkan ke dalam system prompt;
- pertanyaan pengguna ditambahkan setelah konteks perusahaan;
- prompt dikirimkan ke *Large Language* Model;
- model menghasilkan jawaban hanya berdasarkan konteks yang tersedia.

Apabila informasi yang diminta pengguna tidak terdapat pada basis pengetahuan perusahaan, chatbot diarahkan untuk menyatakan bahwa informasi tersebut tidak tersedia dan tidak membuat jawaban di luar konteks.
Pendekatan ini dipilih karena lebih sederhana dibandingkan Retrieval-Augmented Generation (RAG) serta sesuai dengan ukuran basis pengetahuan setiap tenant yang relatif kecil.

8. Implementasi Multi-Tenant

Sistem menerapkan pendekatan *shared schema*.
Seluruh perusahaan menggunakan database yang sama, namun setiap data memiliki identitas tenant_id sehingga seluruh proses query selalu dibatasi berdasarkan identitas tenant yang sedang aktif.
Pendekatan ini memungkinkan:

- efisiensi penggunaan server;
- pemeliharaan sistem yang lebih mudah;
- biaya operasional lebih rendah;
- keamanan data antar-tenant tetap terjaga.
-

Metode Pengujian

Evaluasi sistem dilakukan melalui beberapa jenis pengujian.

a.Black Box Testing

Digunakan untuk memastikan seluruh fungsi sistem berjalan sesuai spesifikasi tanpa melihat implementasi kode program.

b. Pengujian Isolasi *Multi-Tenant*

Dilakukan untuk memastikan data setiap tenant tidak dapat diakses oleh tenant lain sehingga keamanan informasi perusahaan tetap terjaga.

c. Pengujian Akurasi

Pengujian dilakukan menggunakan 25 pertanyaan yang sesuai dengan basis pengetahuan perusahaan untuk mengetahui tingkat ketepatan jawaban chatbot.

d. Pengujian Tingkat Halusinasi

Pengujian dilakukan menggunakan pertanyaan yang berada di luar ruang lingkup basis pengetahuan guna mengevaluasi kemampuan chatbot menolak memberikan informasi yang tidak tersedia.

e. Pengujian Waktu Respons

Pengujian dilakukan dengan mengukur waktu yang dibutuhkan chatbot sejak menerima pertanyaan hingga menghasilkan jawaban kepada pengguna.

Hasil dari seluruh pengujian tersebut digunakan untuk mengevaluasi kualitas sistem dari aspek fungsionalitas, keamanan, akurasi, kecepatan respons, dan kemampuan implementasi pada lingkungan multi-tenant.

HASIL DAN PEMBAHASAN

Implementasi Sistem

Tahap implementasi dilakukan setelah proses analisis kebutuhan dan perancangan sistem selesai. Sistem chatbot customer service dikembangkan menggunakan bahasa pemrograman Golang dengan framework Fiber, basis data PostgreSQL, serta layanan *Large Language Model (LLM)* melalui API OpenRouter. Arsitektur sistem menerapkan konsep *Clean Architecture* sehingga setiap lapisan aplikasi memiliki tanggung jawab yang terpisah, memudahkan proses pengembangan maupun pemeliharaan.

Sistem terdiri atas tiga komponen utama, yaitu dashboard administrator, backend API, dan widget chatbot yang dapat diintegrasikan pada berbagai website perusahaan. Seluruh komponen saling berkomunikasi melalui REST API sehingga proses pengelolaan data dan percakapan berlangsung secara terpusat.

Implementasi Dashboard Administrator

Dashboard administrator dikembangkan sebagai pusat pengelolaan seluruh layanan chatbot. Melalui dashboard ini administrator dapat melakukan pengaturan tenant, mengelola basis pengetahuan, memantau histori percakapan, serta mengubah konfigurasi chatbot sesuai kebutuhan masing-masing perusahaan.
Fitur utama dashboard meliputi:

- autentikasi administrator;
- pengelolaan data tenant;
- pengelolaan data pengguna;
- pengelolaan *knowledge* base;
- pengelolaan konfigurasi chatbot;
- monitoring histori percakapan;

- pengaturan *widget chatbot*.

Dashboard juga menyediakan fasilitas pencarian data, proses tambah, ubah, dan hapus data, serta validasi input untuk menjaga konsistensi informasi yang tersimpan pada basis data.

Implementasi Basis Pengetahuan *(Knowledge Base)*

Knowledge base merupakan komponen utama yang menentukan kualitas jawaban chatbot. Seluruh informasi perusahaan, seperti profil organisasi, layanan, produk, Frequently Asked Questions (FAQ), kebijakan, kontak, serta informasi operasional disimpan dalam basis data.

Setiap tenant memiliki basis pengetahuan yang berbeda sehingga chatbot hanya menggunakan informasi milik tenant yang sedang aktif. Pendekatan ini memastikan jawaban yang diberikan selalu sesuai dengan informasi resmi perusahaan.

Administrator dapat memperbarui informasi melalui dashboard tanpa perlu melakukan perubahan terhadap kode program maupun model AI yang digunakan.

Implementasi *Multi-Tenant*

Implementasi multi-tenant menggunakan pendekatan *shared schema.* Seluruh tenant menggunakan basis data yang sama, tetapi setiap tabel memiliki atribut tenant_id sebagai identitas pemilik data.

Pada saat chatbot menerima permintaan dari pengguna, sistem terlebih dahulu mengidentifikasi tenant berdasarkan konfigurasi widget yang digunakan. Selanjutnya seluruh proses pengambilan data dibatasi menggunakan identitas tenant tersebut sehingga informasi antarperusahaan tetap terisolasi.

Pendekatan ini memberikan beberapa keuntungan, yaitu:

- efisiensi penggunaan sumber daya server;
- kemudahan pemeliharaan aplikasi;
- biaya operasional yang lebih rendah;
- kemampuan melayani banyak perusahaan secara bersamaan;
- keamanan data antar-tenant tetap terjaga.

Hasil implementasi menunjukkan bahwa setiap tenant hanya dapat mengakses basis pengetahuan dan histori percakapannya sendiri tanpa memengaruhi tenant lain.

Implementasi *Context Stuffing*

Salah satu komponen penting pada penelitian ini adalah penerapan metode *Context Stuffing* sebagai mekanisme pengendalian halusinasi.

Sebelum pertanyaan pengguna dikirimkan kepada *Large Language* Model, sistem terlebih dahulu melakukan beberapa tahapan berikut.

1. Mengidentifikasi tenant yang aktif.
2. Mengambil seluruh basis pengetahuan tenant dari database.
3. Menyusun system prompt yang berisi seluruh informasi perusahaan.
4. Menambahkan pertanyaan pengguna pada bagian akhir prompt.
5. Mengirimkan prompt kepada *Large Language* Model.
6. Menampilkan jawaban yang dihasilkan model kepada pengguna.

Apabila informasi yang diminta tidak terdapat dalam basis pengetahuan, chatbot memberikan respons bahwa informasi tersebut tidak tersedia sehingga model tidak menghasilkan jawaban di luar konteks perusahaan.

Pendekatan ini terbukti mampu meningkatkan konsistensi jawaban chatbot sekaligus mengurangi kemungkinan terjadinya hallucination.

Implementasi *Widget Chatbot*

Widget chatbot dikembangkan menggunakan JavaScript sehingga dapat dipasang pada berbagai website melalui penyisipan satu baris kode.

Pada saat halaman website dimuat, widget secara otomatis melakukan proses inisialisasi, mengambil konfigurasi tenant, kemudian membangun antarmuka percakapan sesuai identitas perusahaan.

Konfigurasi yang dimuat secara otomatis meliputi:

- nama perusahaan;
- logo perusahaan;

- warna antarmuka;
- identitas chatbot;
- endpoint API;
- konfigurasi percakapan.

Pendekatan tersebut memudahkan proses implementasi karena perusahaan tidak perlu melakukan modifikasi terhadap aplikasi utama.

Alur Percakapan Chatbot

Proses percakapan chatbot berlangsung melalui beberapa tahapan berikut.

1. Pengguna membuka website perusahaan.
2. Widget chatbot dimuat secara otomatis.
3. Pengguna mengirimkan pertanyaan.
4. Backend mengidentifikasi tenant.
5. Sistem mengambil *Knowledge base* tenant.
6. Prompt disusun menggunakan metode *Context Stuffing*.
7. Prompt dikirim ke *Large Language* Model.
8. Model menghasilkan jawaban.
9. Jawaban ditampilkan kepada pengguna.
10. Histori percakapan disimpan ke database.

Alur tersebut memastikan setiap jawaban selalu mengacu pada informasi resmi perusahaan yang bersangkutan.

Implementasi Histori Percakapan

Setiap percakapan antara pengguna dan chatbot disimpan secara otomatis ke dalam basis data. Informasi yang dicatat meliputi:

- identitas tenant;
- waktu percakapan;
- pertanyaan pengguna;
- jawaban chatbot;
- status permintaan.

Histori percakapan dapat dimanfaatkan administrator untuk melakukan evaluasi terhadap kualitas layanan chatbot, memperbarui basis pengetahuan, serta menganalisis kebutuhan informasi pelanggan.

Pembahasan

Hasil implementasi menunjukkan bahwa arsitektur multi-tenant berhasil memungkinkan satu aplikasi digunakan oleh banyak perusahaan dengan tetap menjaga isolasi data masing-masing tenant. Hal ini memberikan efisiensi dalam penggunaan sumber daya komputasi dan mempermudah proses pemeliharaan sistem dibandingkan pendekatan single-tenant.

Penerapan metode *Context Stuffing* juga memberikan kontribusi terhadap peningkatan kualitas jawaban chatbot. Dengan membatasi sumber informasi pada basis pengetahuan resmi setiap tenant, chatbot mampu menghasilkan respons yang lebih konsisten dan relevan serta meminimalkan kemungkinan munculnya informasi yang tidak sesuai fakta.

Selain itu, implementasi embeddable widget memberikan kemudahan bagi perusahaan dalam mengadopsi sistem chatbot. Integrasi yang hanya memerlukan satu baris kode memungkinkan chatbot diterapkan pada berbagai website tanpa perubahan signifikan terhadap struktur aplikasi yang telah ada.

Secara keseluruhan, implementasi sistem menunjukkan bahwa kombinasi arsitektur multi-tenant, metode *Context Stuffing*, dan embeddable widget mampu menghasilkan solusi layanan pelanggan berbasis *Artificial Intelligence* yang efisien, mudah diimplementasikan, dan memiliki potensi untuk diterapkan pada berbagai sektor industri.

Pengujian Black Box

Pengujian *Black Box* dilakukan untuk memastikan seluruh fungsi sistem berjalan sesuai dengan kebutuhan fungsional yang telah dirancang tanpa memperhatikan implementasi kode program. Pengujian difokuskan pada validasi masukan (input) dan keluaran (output) dari setiap modul utama sistem.

Fitur yang diuji meliputi autentikasi administrator, pengelolaan tenant, pengelolaan knowledge base, pengelolaan pengguna, proses percakapan chatbot, pengelolaan konfigurasi widget, serta penyimpanan histori percakapan.

Tabel 2. Hasil Pengujian *Black Box*

No	Modul yang Diuji	Hasil
1	Login Administrator	Berhasil
2	Manajemen Tenant	Berhasil
3	Manajemen Knowledge Base	Berhasil
4	Manajemen User	Berhasil
5	Konfigurasi Widget	Berhasil
6	Percakapan Chatbot	Berhasil
7	Penyimpanan Histori Chat	Berhasil
8	Logout	Berhasil

Berdasarkan hasil pengujian, seluruh fungsi utama sistem berjalan sesuai dengan kebutuhan yang telah ditentukan. Tidak ditemukan kegagalan proses maupun kesalahan validasi pada modul yang diuji.

Pengujian Akurasi Jawaban Chatbot

Pengujian akurasi dilakukan untuk mengetahui kemampuan chatbot dalam memberikan jawaban yang sesuai dengan basis pengetahuan masing-masing tenant. Pengujian menggunakan seperangkat pertanyaan yang telah disusun berdasarkan informasi resmi perusahaan.

Setiap jawaban dibandingkan dengan informasi yang terdapat pada *Knowledge base* untuk menentukan tingkat kesesuaiannya.

Tabel 3. Hasil Pengujian Akurasi

Parameter	Hasil
Jumlah Pertanyaan	25
Jawaban Sesuai	25
Jawaban Tidak Sesuai	0
Tingkat Akurasi	**100%**

Hasil pengujian menunjukkan bahwa chatbot mampu memberikan jawaban yang sesuai terhadap seluruh pertanyaan yang berada dalam ruang lingkup knowledge base. Hal ini menunjukkan bahwa penerapan *Context Stuffing* berhasil mengarahkan *Large Language* Model untuk menghasilkan respons yang relevan dengan informasi resmi perusahaan.

Pengujian Tingkat Halusinasi

Pengujian halusinasi dilakukan menggunakan pertanyaan yang berada di luar cakupan basis pengetahuan perusahaan. Tujuannya adalah untuk mengetahui kemampuan chatbot dalam menolak memberikan informasi yang tidak tersedia.

Beberapa skenario pengujian meliputi pertanyaan mengenai produk yang tidak dimiliki perusahaan, informasi organisasi lain, maupun data yang tidak terdapat pada knowledge base.

Tabel 4. Hasil Pengujian Halusinasi

Parameter	Hasil
Pertanyaan di luar konteks	10
Jawaban di luar knowledge base	0
Tingkat Halusinasi	0%

Pada seluruh skenario tersebut chatbot tidak menghasilkan informasi yang tidak terdapat pada basis pengetahuan, melainkan memberikan respons bahwa informasi tersebut tidak tersedia. Hasil ini menunjukkan bahwa pendekatan *Context Stuffing* efektif dalam

mengurangi risiko hallucination pada implementasi layanan pelanggan berbasis *Large Language* Model.

Pengujian Isolasi Data *Multi-Tenant*

Pengujian ini bertujuan memastikan bahwa data setiap tenant tetap terisolasi dan tidak dapat diakses oleh tenant lain.

Skenario pengujian dilakukan dengan menggunakan beberapa tenant yang memiliki basis pengetahuan berbeda. Pengguna pada tenant tertentu kemudian mengajukan pertanyaan yang berkaitan dengan informasi tenant lain.

Tabel 4. Hasil Pengujian Isolasi *Multi-Tenant*

Skenario	Hasil
Tenant A mengakses data Tenant B	Ditolak
Tenant B mengakses data Tenant A	Ditolak
Chatbot hanya menggunakan *Knowledge base* tenant aktif	Berhasil
Histori percakapan terpisah	Berhasil

Hasil pengujian menunjukkan bahwa seluruh proses query selalu dibatasi berdasarkan identitas tenant_id sehingga data antarperusahaan tetap terjaga. Dengan demikian, arsitektur shared schema yang diterapkan mampu memberikan efisiensi penggunaan basis data tanpa mengorbankan keamanan informasi.

Pengujian Waktu Respons

Pengujian waktu respons dilakukan untuk mengukur lama proses sejak pengguna mengirimkan pertanyaan hingga chatbot menampilkan jawaban.

Pengukuran dilakukan pada beberapa skenario percakapan dengan kompleksitas pertanyaan yang berbeda.

Tabel 5. Hasil Pengujian *Response Time*

Pengujian	Rata-rata Waktu
Respons Chatbot	±4 detik

Waktu respons tersebut dipengaruhi oleh proses penyusunan konteks, komunikasi dengan layanan *Large Language* Model melalui API, serta proses inferensi model. Meskipun terdapat waktu tambahan akibat pemanggilan layanan eksternal, hasil pengujian menunjukkan bahwa performa sistem masih berada pada kategori yang layak untuk implementasi layanan customer service daring.

KESIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan chatbot customer service berbasis *Large Language Model (LLM)* dengan arsitektur multi-tenant yang dapat diintegrasikan ke berbagai website perusahaan melalui embeddable widget. Sistem dikembangkan menggunakan bahasa pemrograman Golang, framework Fiber, basis data PostgreSQL, serta layanan LLM melalui API OpenRouter dengan menerapkan pendekatan *Clean Architecture* sehingga aplikasi memiliki struktur yang modular dan mudah dikembangkan. Penerapan metode *Context Stuffing* terbukti mampu membatasi ruang pengetahuan chatbot pada basis pengetahuan resmi masing-masing tenant sehingga respons yang dihasilkan lebih konsisten dan relevan. Berdasarkan hasil pengujian, sistem mampu memberikan akurasi jawaban sebesar 100% untuk pertanyaan yang sesuai dengan knowledge base, tidak menghasilkan halusinasi pada pertanyaan di luar konteks informasi perusahaan, serta memiliki rata-rata waktu respons sekitar empat detik. Selain itu, mekanisme *shared schema* yang diterapkan berhasil menjaga isolasi data antar-tenant sehingga informasi masing-masing perusahaan tetap aman. Implementasi *embeddable widget* memberikan kemudahan integrasi karena chatbot dapat dipasang pada website perusahaan menggunakan satu baris kode JavaScript tanpa memerlukan perubahan besar pada aplikasi yang telah dimiliki. Dengan demikian, sistem yang dikembangkan dapat menjadi alternatif solusi layanan pelanggan berbasis *Artificial Intelligence* yang efisien, mudah diimplementasikan, dan memiliki skalabilitas tinggi untuk mendukung kebutuhan berbagai organisasi.

REFERENSI

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

Evans, V., & Grefenstette, E. (2018). Learning explanatory rules from noisy data. *Artificial Intelligence Review, 49*(2), 223–241.

Google. (2024). *Artificial intelligence overview*. https://ai.google/

Martin, R. C. (2018). *Clean architecture: A craftsman's guide to software structure and design*. Pearson Education.

OpenAI. (2023). *GPT-4 technical report*. arXiv. https://arxiv.org/abs/2303.08774

PostgreSQL Global Development Group. (2024). *PostgreSQL documentation*. https://www.postgresql.org/docs/

Richards, M. (2020). *Fundamentals of software architecture: An engineering approach*. O'Reilly Media.

Sommerville, I. (2021). *Software engineering* (11th ed.). Pearson.

The Go Authors. (2024). *The Go programming language*. https://go.dev/

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv*. https://arxiv.org/abs/2312.10997