ISSN 2085-4552

Customer Service Chat Application Design Raden Inten II Airport, Lampung

M. Alif Ridho Setiawan[1], Indra Gunawan[2], Mezan el-Khaeri Kesuma[3], Fiqih Satria[4]
[1234]Prodi Sistem Informasi, Universitas Negeri Islam Raden Intan Lampung, Bandar Lampung, Indonesia
mezan@radenintan.ac.id

Accepted 04 January 2026
Approved 22 January 2026

***Abstract*— This study aims to design a web-based customer service application with live chat and chatbot features that implement Retrieval-Augmented Generation (RAG) technology at Raden Inten II Airport, Lampung. The main problems encountered are the slow response of conventional services and limited access to information by users located far from the airport. The system development uses the Waterfall model which includes requirements analysis, system design, implementation, testing, and maintenance. The main features in the application include user authentication, live chat, RAG chatbot, and data management through the admin dashboard. System testing was conducted using the black-box method, indicating that all features function according to specifications. The results of the study indicate that this system can significantly improve the efficiency and quality of customer service. For further development, it is recommended to use WebSocket to improve real-time communication performance.**

***Index Terms*— Customer Service; Live Chat; Chatbot; RAG; Web-based Application.**

I. INTRODUCTION

The development of artificial intelligence (AI) technology has had a significant impact on increasing the efficiency of customer service, including in the air transportation sector.[1] Artificial Intelligence (AI) is a computer or machine technology that possesses human-like intelligence. One form of AI implementation in industry is the use of chatbots as a customer relations tool.[2] One rapidly developing technology is chatbots, AI-based software capable of interacting with users in real-time.[3] One modern approach to chatbot development is Retrieval-Augmented Generation (RAG)[4], the chatbot in this study is built using a RAG-based model.[5] Retrieval-Augmented Generation technology is expected to be integrated into chatbots,[6] which combines relevant data retrieval and generative capabilities to generate contextual and accurate responses.[7]

Live chat has become an increasingly popular way to provide real-time service in today's customer service environment.[8] Live chat enhances communication between customers and agents. Live chat is crucial for creating a positive customer experience. Good communication can enable information exchange, problem-solving, and relationship building.[9]

Public service organizations are experiencing an increase in digital requests from citizens (e.g., web pages, social media, or service apps), which has increased during times of social distancing. To effectively address this surge, we need to combine existing resources with new ways that go beyond existing service models. Chatbots are an example of an Artificial Intelligence application that has been widely used to address the surge in service demand, performing communication tasks previously performed by humans.

As one of the main transportation hubs in Lampung Province, Raden Inten II Airport in Lampung faces challenges in providing responsive and informative customer service. Conventional systems that rely on direct interaction often result in delays in providing information and services. Web-based applications offer numerous benefits and advantages for supporting business and service operations. They provide cross-platform accessibility, allowing users to access services anytime and anywhere via browsers without the need for installation. They are also easier to update and maintain, ensure consistent performance across devices, and enable seamless integration with various digital services such as databases, payment systems, and customer support tools[10], [11].

To address these challenges, developing customer service chat applications is a potential solution. Many companies are also starting to provide chatbot features to automate communications with people who use computers as a tool to interact with customers.[12] RAG technology enables chatbots to provide more accurate and relevant responses by leveraging existing databases and dynamically generating answers[13], [14], [15]. With this technology, customers can obtain information and assistance quickly and conveniently, accessible wherever they are. Good service with real-time feedback can improve customer satisfaction. The novelty of this study lies in three main aspects: (1) integration of a web-based live chat system with an AI chatbot into a single seamless platform; (2) the use of

the Gemini + RAG model to develop a domain-specific chatbot tailored to the operational and informational context of an airport; and (3) the implementation of a knowledge base system directly managed by administrators via a web interface, enabling real-time updates of information and responses.

The technical integration of RAG in this study is designed not merely for computational accuracy, but to address the operational challenges of customer service. Through text normalization and semantic search (FAISS/ChromaDB), the system is capable of comprehending informal customer complaints and mapping technical issues to the appropriate SOP solutions. Customer data security is ensured through anonymization mechanisms and local LLM execution (Ollama), while fallback and feedback loop features ensure that any complaints unresolved by the bot are escalated to human agents, thereby maintaining high customer satisfaction standards.

II. METHOD

This research uses the Waterfall Model approach, namely a structured and sequential software development method. The Waterfall Model is the oldest and the most wellknown SDLC model. This model is widely used in government projects and in many major companies.[16] This model ensures that design errors can be detected before product development begins. It is well-suited for projects where quality control is a primary concern, as it emphasizes intensive documentation and planning. The stages include:

This study adopts the Waterfall model rather than iterative frameworks like Agile or Scrum. This choice is justified by the strict sequential dependency inherent in developing Retrieval-Augmented Generation (RAG) pipelines. The output quality of the Generator module (LLM) is strictly dependent on the performance of the Retriever module, which in turn relies on the integrity of the Data Preprocessing and Embedding phases.

Unlike Agile, which is optimized for volatile requirements, this project operates under fixed, critical constraints regarding data privacy and local execution. Consequently, the intensive System and Software Design phase provided by the Waterfall model is imperative to finalize the security architecture and vectorization schema before implementation begins, thereby minimizing the risk of costly architectural revisions later in the development cycle.

Although Waterfall is linear, we incorporated a feedback mechanism during the 'Operation and Maintenance' phase to allow for minor refinements based on user escalation data, providing a balance between structural rigidity and operational adaptability[17], [18], [19].

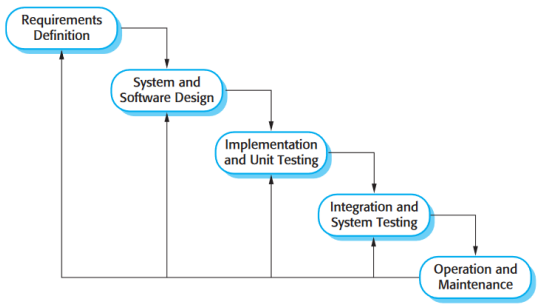

Figure 1: Waterfall Method

Figure 1 explain that:

- Requirements Definition: This stage encompasses the identification of customer complaint challenges (such as slang and typos) and data privacy requirements, which served as the foundation for selecting Local LLM technology.
- System and Software Design: This phase involves designing the technical RAG architecture, including the selection of PDF document chunking schemes and the vector database design (FAISS/ChromaDB).
- Implementation and Unit Testing: The coding stage (using FastAPI/Next.js backend) and isolated unit testing (e.g., verifying the correct functionality of text normalization).
- Integration and System Testing: A critical phase where the retriever module is integrated with the generator (LLM) and evaluated for accuracy (utilizing metrics such as BLEU/ROUGE)
- Operation and Maintenance: The deployment phase where the system actively serves customers. The feedback arrow indicates that data from real-world interactions (e.g., escalation tickets to the Admin) is utilized to update and refine the system in the initial phases.

III. RESULT AND DISCUSSION

A. Requirement Definition

Data was collected through observations and interviews with customer service personnel at Raden Inten II Airport, Lampung. It was found that users needed the following services:

1. User authentication
2. Live chat with admins
3. RAG-based chatbot
4. Admin dashboard for data management.

The purpose of this identification process was to ensure that the designed system truly met the needs of

end users, both customers and customer service officers.

B. System and Software Design

System design is done through use case diagrams, class diagrams, system architecture and system usage flow.

1. Use Case Diagram

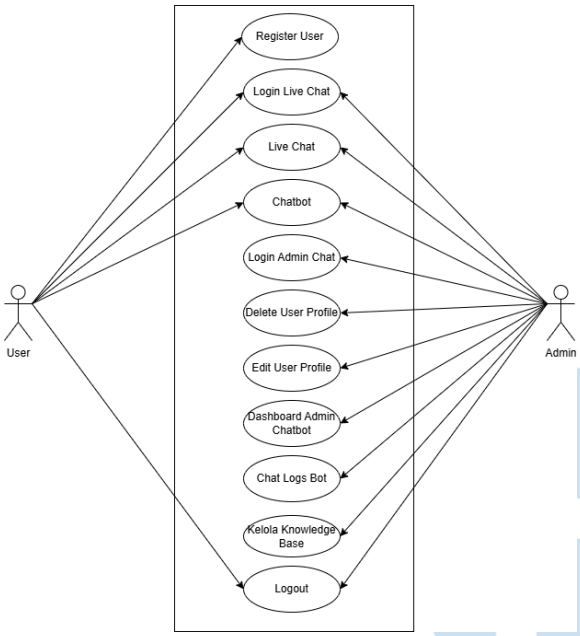

Figure 2: Usecase Diagram

Based on the use case diagram in the figure 2 above, the design of this customer service web application system involves two main actors: the User and the Admin. From the User perspective, the system provides essential functionality, including User Registration to create a new account and Live Chat Login to access the system. Once logged in, Users can choose to interact directly with staff via Live Chat or receive a quick, automated response from a Chatbot. The User session ends with a Logout function.

Admins, on the other hand, have more comprehensive access rights to manage the entire system. In addition to interacting with Users via Live Chat, Admins have a dedicated panel (Login Admin Chat) that grants them the authority to perform user management functions, such as Editing User Profiles and Deleting User Profiles. The Admin's core functionality is chatbot management, which includes access to the Chatbot Admin Dashboard to monitor activity, view conversation history (Chat Logs Bot), and most importantly, manage the knowledge base (Manage Knowledge Base), which serves as a source of information and answers for the chatbot. Like Users, Admins also have a Logout function to exit the system. Overall, this workflow is designed to create efficient and interactive customer service.

2. Class Diagram

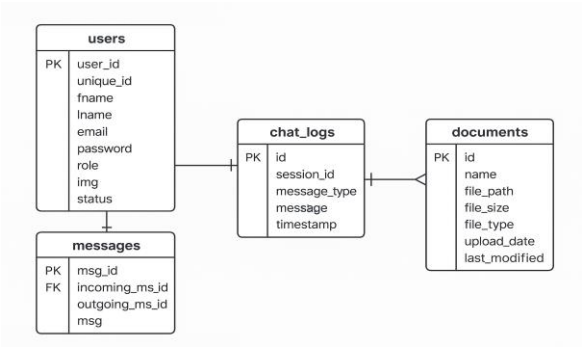

Figure 3: Class Diagram

The database structure used in designing this system consists of four main entities: users, messages, chat_logs, and documents. These four entities represent two main types of interactions in the system: live chat communication between users and admins, and Retrieval-Augmented Generation (RAG)-based chatbot interactions.[20], [21], [22], [23] The following explains the relationships between these entities:

a. Relationship between users and messages

The users table stores both user and system admin data. This table has a one-to-many relationship with the messages table, where a single user can send and receive multiple messages. This is represented by two foreign key attributes in messages: incoming_msg_id and outgoing_msg_id, each of which refers to the unique_id attribute in the users table. This relationship is necessary to support the two-way live chat feature between users and admins.

b. Relationship between users and chat_logs

The chat_logs table stores the history of interactions between users and the chatbot. Although there is no explicit foreign key to the users table, logically, each session_id in chat_logs is generated based on the identity of the currently active user. Therefore, this relationship is implicitly one-to-many, where a single user can have multiple chat sessions recorded in the system log.

c. Relationship between chat_logs and documents

In the RAG approach, chatbot responses are generated not only from the generative model but also from relevant documents retrieved from the documents table. Therefore, there is a many-to-one relationship from chat_logs to documents, reflecting that a single chatbot log entry can reference a single source document. Advanced implementations can extend this relationship to many-to-many, if a single chatbot response references more than one document.

d. The documents table as a knowledge base

The documents table stores the entire knowledge base that can be used by the chatbot. This entity does not have direct inbound or outbound foreign keys, but it is a crucial component in the RAG system's retrieval pipeline. Document files are stored along with metadata such as file name, size, type, and upload and update times.

With this ERD structure, the system is able to clearly separate user interactions (live chat) from interactions with the chatbot (chat_logs), and supports factual information retrieval through integration with the knowledge base.

3. System Architecture

The system architecture describes how the system is built from a technical and infrastructure perspective. This application is a web application consisting of several main components:

a. Client Side: The user and admin browsers serve as the main interface.
b. Application Server: The backend web server that processes requests from users/admins.
c. Live Chat Engine: Uses XMLHttpRequest for its live chat feature.
d. Chatbot Engine (LLM/RAG): The chatbot engine that generates automated responses using the integration of LLM models such as Gemini and the Retrieval-Augmented Generation approach.
e. Database: Stores user data, conversation history, and the chatbot's knowledge base.

4. System Usage Flow

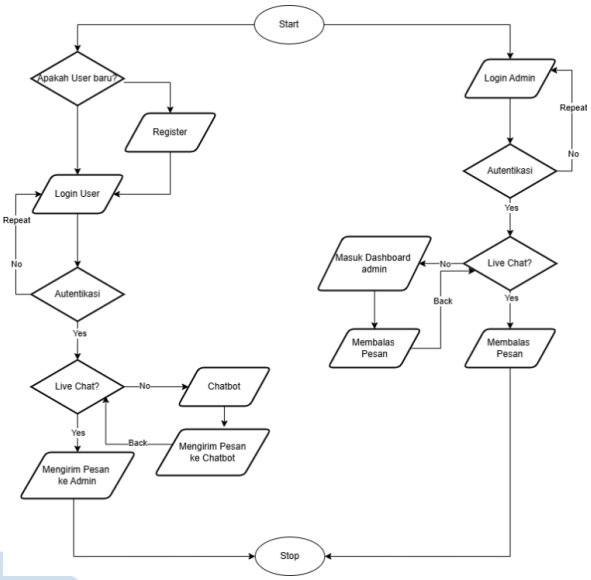

Figure 4: System Usage Flow

On figure 4 shows that System flowchart demonstrating the hybrid interaction flow with fallback mechanism to human agents.

C. Implementation and Unit Testing

After the system analysis and design are complete, the next stage is the implementation and testing of the customer service chat application. Implementation includes the design and coding phase, aligning with the business flow and integrating it with the database. The goal of this phase is to ensure that the customer service application functions properly according to specifications in a real-world environment. The implementation and testing of the customer service chat application are explained in the following page:

1. Register User Page

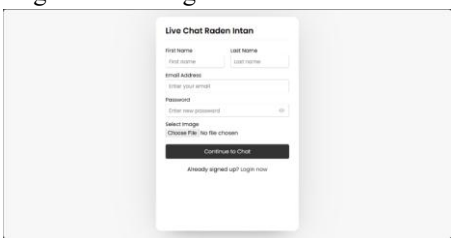

Figure 5: Register User page

The Figure 5 shows the registration page is used by new users to register before accessing the live chat or chatbot features. On this page, users are asked to enter their full name, email address, and

password, which will be used to log in to the system.

Once all the data has been entered, the user can click the Register button to save the data to the system. Validation is performed to ensure the email address is not duplicated and the password meets security standards.

2. Login User and Admin Page

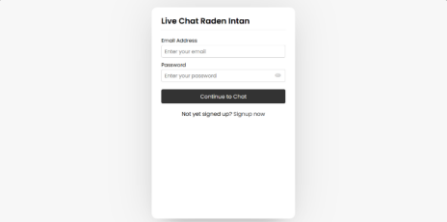

Figure 6: Login User and Admin

The figure shows After successfully registering, the user will be directed to the login page. This page is used for authentication by entering the email and password.

If the login data is valid, then the user will enter the main dashboard. If it is not valid, the system will display an error message.

3. Dashboard Chat User Page

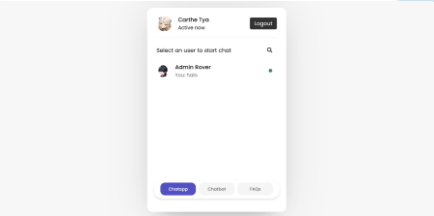

Figure 7: Dashboard Chat User Page

The figure shows The dashboard is the first page that appears after successfully logging in. It features a menu with options for live chat or a chatbot.

Users can choose the communication features they need. The interface is simple and intuitive for easy access.

4. Dashboard Chat Admin Page

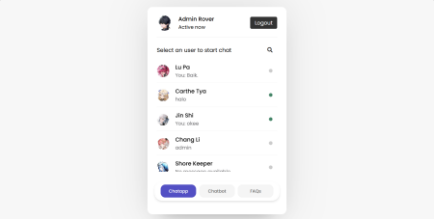

Figure 8: Dashboard Chat Admin Page

The figure shows The live chat admin dashboard page is the main interface for admins responsible for handling user messages in real-time.

Admins can view a list of recent conversations and access the live chat menu to respond to user messages directly.

5. Chat Interface Page

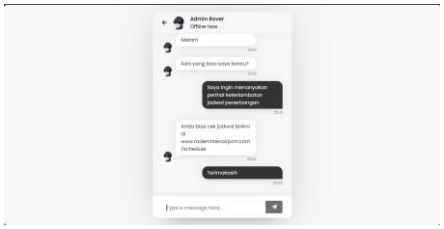

Figure 9: Chat Interface Page

The figure shows the page displays a conversation interface between the admin and the user. The admin can reply to user messages directly through the input box.

Messages are displayed chronologically and arranged like a chat window, making it easy for admins to monitor conversations in real-time.

6. Chatbot Page

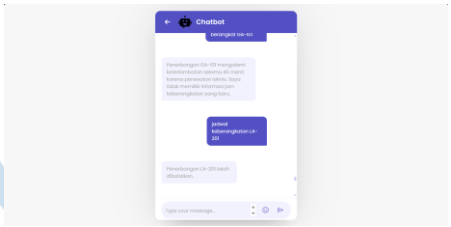

Figure 10: Chatbot Page

The figure shows The chatbot page allows users to interact with the system automatically. The chatbot works using a Retrieval-Augmented Generation (RAG) approach that can retrieve relevant documents and generate LLM-based answers.

Users simply type a question, and the system will respond quickly with accurate and contextual information.

7. Login Admin Chatbot Page

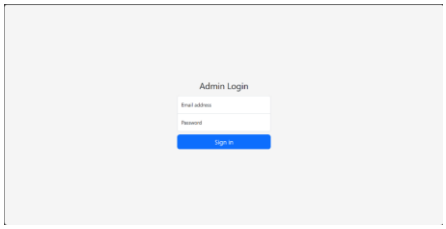

Figure 11: Login Admin Chatbot Page

The figure shows The login page is used by admins who manage the chatbot system. The login process requires valid credentials to access the chatbot dashboard.

8. Dashboard Admin Bot Page

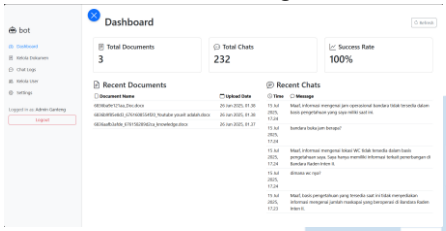

Figure 12: Dashboard Admin Bot Page

The figure shows The dashboard presents information such as API status, number of user interactions with the chatbot, and performance logs.

Admins can monitor the efficiency of chatbot responses and perform troubleshooting if necessary.

9. Management Knowledge Base Bot Page

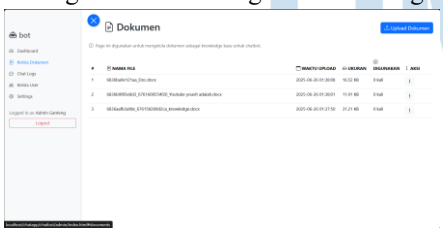

Figure 13: Management Knowledge Base Bot Page

The figure shows page allows you to upload, view, delete, or update documents that serve as sources of information for the chatbot during the retrieval process.

The managed documents will be used by the RAG system to factually answer user questions.

10. Management Chat Logs Bot Page

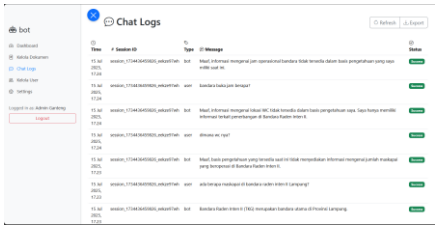

Figure 14: Management Chat Logs Bot Page

The figure shows Admins can monitor the entire history of user interactions with the chatbot through this page. Log data includes user input, chatbot output, interaction time, and success status.

This feature helps admins evaluate chatbot performance and identify responses that need to be improved or followed up.

11. Management User Page

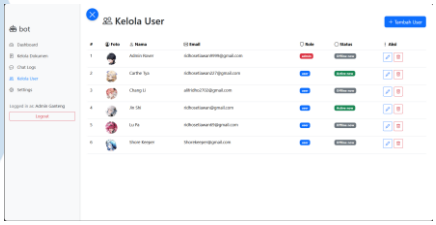

Figure 15: Management User Page

The figure shows Admin can manage available users.

12. API Management

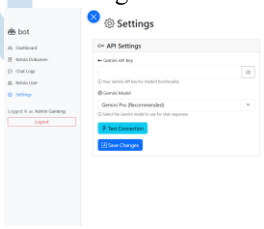

Figure 16: API Management

The figure shows Admin can manage the Gemini API if an error occurs.

D. Integration and System Testing

Testing is done using the black-box method. System testing is crucial because everyone makes mistakes when creating software. The errors in each piece of software will be different.[24] Therefore, we use blackbox testing to verify the functionality of the:

1. Table 1: Testing of Login Page

No	Skenario Test	Input	Test Step	Expected Output	Test Result	Status
1	Login with valid data	Valid Username & Password	Enter input and click login	User is redirected to the chat dashboard page	Output is correct	Successful
2	Login does not match valid data	Valid Username/Password	Enter input and click login	Login failed error alert appears	Output is correct	Successful
3	Leave login input blank	Empty	Click login without entering input	Please fill in all fields	Output is correct	Successful

2. Table 2: Testing of Live Chat

No	Skenario Test	Input	Test Step	Expected Output	Test Result	Status
1	User sends message to admin	Message text	Type and send	Incoming message to admin	Output is correct	Successful
2	Admin replies to message	Message text	Type and send	Incoming message to user	Output is correct	Successful
3	Send empty message	Empty	Click send	Shown please fill in all fields	Output is correct	Successful

3. Table 3: Testing of Chatbot (RAG)

No	Skenario Test	Input	Test Step	Expected Output	Test Result	Status
1	User asks for flight schedule information	Jadwal hari ini?	Type and send	Chatbot responds with the schedule information provided in the link.	Output is correct	Successful
2	User asks about airports	Dimanakah bandara terletak?	Type and send	Chatbot responds with a relevant answer	Output is correct	Successful
3	Random question other than airports	Siapa presiden Indonesia?	Type and send	“Maaf, saya adalah chatbot layanan pelanggan yang menyediakan informasi penerbangan di Bandara Raden Inten II (TKG). Saya tidak memiliki informasi mengenai presiden Indonesia.”	Output is correct	Successful

E. Operation and Maintenance

The system is monitored and evaluated regularly. User feedback serves as the basis for continuous improvement and development.

F. System Features

1. User Authentication: Users can register and log in to access the system.
2. Live Chat: Direct communication with a customer service representative for complex questions.
3. RAG Chatbot: An automated chatbot system that responds to questions based on relevant documents using a retrieval and generation approach.
4. Admin Dashboard: Provides user management, document management, chatbot settings, and conversation log monitoring.

G. System Testing

Table 4: The result of features tested

No	Features tested	Result
1	Login	Valid
2	Registration	Valid
3	Live Chat	Valid
4	Chatbot RAG	Valid
5	Dashboard Admin	Valid

All features functioned as intended. The chatbot successfully responded to questions with a high level of relevance, while live chat provided personalized assistance options when needed.

To complement functional (black-box) testing, this study implements system performance evaluation and human assessment to ensure the chatbot's operational readiness. Latency measurement is adopted from Albert & Voutama [13] to guarantee real-time responsiveness, which is crucial for customer satisfaction, while interface quality is evaluated using Lighthouse, as in the study by Pratama & Sisephaputra [14]. Furthermore, to mitigate the risk of fatal information hallucination in public services, human evaluation is conducted involving expert staff using Accuracy, Completeness, and Clarity parameters, referring to the methodology of Putro et al[15].

While initial unit testing confirmed the functional validity of all system modules (success/fail), this study further employed quantitative metrics to rigorously assess system performance. Response latency was measured to evaluate efficiency[13], ROUGE-L scores were calculated to quantify answer relevance[15], and Lighthouse metrics were utilized to assess interface quality[14], thereby providing a comprehensive, data-driven evaluation beyond simple binary validation.

H. Finding

The system architecture is engineered with a primary emphasis on operational resilience and contextual accuracy. The core mechanism initiates when user input undergoes processing via a normalization module to mitigate linguistic noise. Subsequently, the system activates a Dual-Path Routing mechanism (as illustrated in the Flowchart): the primary path leverages Retrieval-Augmented Generation to extract technical solutions from the vector knowledge base, while the secondary path functions as a safety net, automatically escalating tickets to the Admin interface should the AI resolution prove inadequate. This entire process executes within a local infrastructure to guarantee data sovereignty.

The integration of RAG into a chatbot system has proven effective in providing contextual information. Compared to conventional chatbot methods, RAG can reduce generic responses and provide more specific answers. However, the system lacks multilingual support and doesn't utilize real-time protocols like WebSocket, which could potentially be added in future development.

RAG is not merely a technological enhancement, but a strategic solution for operational challenges and data security. The transition to RAG provides three strategic leaps: First, From keyword matching to semantic understanding. Second, From generic responses to grounded, internal fact-based responses. Third, From cloud dependency to local infrastructure sovereignty (privacy-preserving)."

IV. CONCLUSION

This research successfully designed and implemented a customer service web application at Raden Inten II Airport, Lampung, integrating live chat and the RAG chatbot. This system provides easy access to information, fast responses, and increased operational efficiency for customer service. Testing results showed that all features performed according to specifications.

Recommendations for further development could consider the proposed improvements for the system include using WebSocket to increase the speed and responsiveness of the live chat feature, adding multilingual capabilities to accommodate users from diverse linguistic backgrounds, and integrating the platform with mobile applications to achieve a broader reach and enhance accessibility.

ACKNOWLEDGMENT

The author would like to express his gratitude to Allah SWT for His blessings so that this scientific article can be completed well. Thanks are also extended to Customer Service of Raden Inten II Airport Lampung who has provided support and data for research purposes, as well as to Mr. Indra Gunawan and Mr. Mezan el-Khaeri Kesuma as supervisors who

have guided him patiently. Thanks are also extended to his beloved parents and all parties who have helped directly or indirectly in the process of compiling this scientific article.

.

REFERENCES

[1] N. Ali dan M. Alfayez, "The impact of E-CRM on customer loyalty in the airline industry: the mediating role of customer experience," *Cogent Business & Management*, vol. 11, no. 1, hlm. 2364838, Des 2024, doi: 10.1080/23311975.2024.2364838.

[2] T. Destiany dan Y. Iskandar, "ANALISIS CUSTOMER SERVICE MENGGUNAKAN CHATBOT BERBASIS ARTIFICIAL INTELLIGENCE (Suatu Studi pada Daya Motor Honda Ciamis)," vol. 4, 2022.

[3] C. A. Oktavia, "Implementasi Chatbot Menggunakan Dialogflow dan Messenger Untuk Layanan Customer Service Pada E-Commerce," *JIMP*, vol. 4, no. 3, Jan 2020, doi: 10.37438/jimp.v4i3.230.

[4] Y. Tribber dan M. Asfi, "Implementasi Retrieval Augmented Generation untuk Layanan Informasi Kampus dengan Chatbot Berbasis AI," 2024.

[5] Muhammad Irfan Syah, Nazruddin Safaat Harahap, Novriyanto, dan Suwanto Sanjaya, "PENERAPAN RETRIEVAL AUGEMENTED GENERATION MENGGUNAKAN LANGCHAIN DALAM PENGEMBANGAN SISTEM TANYA JAWAB HADIS BERBASIS WEB," *zn*, vol. 6, no. 2, hlm. 370–379, Mei 2024, doi: 10.31849/zn.v6i2.19940.

[6] Universitas Internasional Batam, Y. Christian, M. Siahaan, dan H. Hansvirgo, "Designing a Web-Based Light Novel Application with an LLM-Powered Chatbot Recommendation System Using Scrum Methodology," *jmika*, vol. 8, no. 2, hlm. 174–186, Okt 2024, doi: 10.46880/jmika.Vol8No2.pp174-186.

[7] J. Miao, C. Thongprayoon, S. Suppadungsuk, O. A. Garcia Valencia, dan W. Cheungpasitporn, "Integrating Retrieval-Augmented Generation with Large Language Models in Nephrology: Advancing Practical Applications," *Medicina*, vol. 60, no. 3, hlm. 445, Mar 2024, doi: 10.3390/medicina60030445.

[8] M. Adam, M. Wessel, dan A. Benlian, "AI-based chatbots in customer service and their effects on user compliance," *Electron Markets*, vol. 31, no. 2, hlm. 427–445, Jun 2021, doi: 10.1007/s12525-020-00414-7.

[9] J. Heinonen dan E. Sthapit, "Service Agent Driven Co-Created Caring in Chat-Based Customer Service Encounters," *Services Marketing Quarterly*, vol. 45, no. 1, hlm. 1–24, Jan 2024, doi: 10.1080/15332969.2023.2288733.

[10] A. Waworuntu, "Rancang Bangun Aplikasi e-Commerce Dropship Berbasis Web," *Ultimatics*, vol. 12, no. 2, hlm. 118–124, Des 2020, doi: 10.31937/ti.v12i2.1823.

[11] H. Wijaya, D. Supriyanti, dan A. Saefullah, "Penggunaan Teknologi Web 2.0 dan Dampak Perubahannya pada Aplikasi Website berbasis Rich Internet Application (RIA)," *Ultimatics*, vol. 9, no. 2, hlm. 72–81, Sep 2017, doi: 10.31937/ti.v9i2.621.

[12] F. E. Office, "AI-based Chatbot Service for Financial Industry," vol. 54, no. 2, 2018.

[13] G. D. Albert dan A. Voutama, "PENGEMBANGAN CHATBOT BERBASIS PDF MENGGUNAKAN LOCAL RETRIEVAL-AUGMENTED GENERATION (RAG) DAN OLLAMA," *JITET*, vol. 13, no. 2, Apr 2025, doi: 10.23960/jitet.v13i2.6361.

[14] I. I. R. Pratama dan B. Sisephaputra, "Pengembangan Sistem Helpdesk Menggunakan Chatbot Dengan Metode Retrieval-Augmented Generation (RAG)," *JINACS: Journal of Informatics and Computer Science*, vol. 06, no. 3, 2024.

[15] I. P. H. Putro, J. Antoni, M. K. Adhitya, N. A. Herawati, A. Purwarianti, dan N. P. Utama, "Retrieval-Augmented Generation (RAG) Chatbot for Handling Customer Complaints in the Energy Sector," *Jurnal Infomedia : Teknik Informatika, Multimedia, dan Jaringan*, vol. 10, no. 2, hlm. 105–111, 2025.

[16] "A_Comparison_Between_Three_SDLC_Models_W."

[17] P. G. S. C. Nugraha, "Rancang Bangun Sistem Informasi Software Point of Sale (Pos) Dengan Metode Waterfall Berbasis Web," *JST (Jurnal Sains dan Teknologi)*, vol. 10, no. 1, hlm. 92–103, 2021, doi: 10.23887/jstundiksha.v10i1.29748.

[18] F. Nurdiansyah, E. Daniati, dan A. Ristyawan, "Pengembangan Sistem Informasi Kasir Apotek Dengan Metode Waterfall," *EDUSAINTEK*, vol. 9, no. 3, hlm. 752–773, Agu 2022, doi: 10.47668/edusaintek.v9i3.550.

[19] R. Susanto dan A. D. Andriana, "Perbandingan model waterfall dan prototyping untuk pengembangan sistem informasi," *Majalah Ilmiah UNIKOM*, vol. 14, no. 1.

[20] "Methods Of Implementing Retrieval-Augmented Generation In Combination With Modern Large Language Models," *SNSUT*, vol. 7, no. 12, 2025, doi: 10.31673/2786-8362.2025.012843.

[21] G. D. Marcantonio, "Intelligenza artificiale, Large Language Models (LLMs) e Retrieval-Augmented Generation (RAG).1 Nuovi strumenti per l'accesso alle risorse archivistiche e bibliografiche," *Intelligenza artificiale*.

[22] X. Li *dkk.*, "Building A Coding Assistant via the Retrieval-Augmented Language Model," 2 November 2024, *arXiv*: arXiv:2410.16229. doi: 10.48550/arXiv.2410.16229.

[23] J. Chen, H. Lin, X. Han, dan L. Sun, "Benchmarking Large Language Models in Retrieval-Augmented Generation," *AAAI*, vol. 38, no. 16, hlm. 17754–17762, Mar 2024, doi: 10.1609/aaai.v38i16.29728.

[24] F. C. Ningrum, D. Suherman, S. Aryanti, H. A. Prasetya, dan A. Saifudin, "Pengujian Black Box pada Aplikasi Sistem Seleksi Sales Terbaik Menggunakan Teknik Equivalence Partitions," *JIUP*, vol. 4, no. 4, hlm. 125, Des 2019, doi: 10.32493/informatika.v4i4.3782.