International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

LLM-Powered Customer Support Systems for Multi-Tenant SaaS: Design and Evaluation

Vasanthi Jangala Naga

Sr. Software Engineer, Fremont, CA, USA
Email: *vasanthi.jangala[at]gmail.com*

Abstract: ***Customer support automation is indispensable to the SaaS platforms as they scale up, gain complexity, and present greater operating costs. Traditional rule-based and intent-driven chatbots struggle to generalize across varied, unstructured user questions and require a lot of manual upkeep. Large language models open the door to advanced functionality in chatbots that reason contextually and understand the underlying natural language better but rolling them out reliably in enterprise SaaS settings raises problems associated with hallucinations, latency, cost, and keeping data isolated across tenants. This paper adopts a system-centric approach to LLM-powered customer support chatbots, optimized for multitenant SaaS environments. We present an RAG architecture that roots LLM responses in the company's knowledge base and introduces human-in-the-loop defenses that enhance reliability. We evaluate the proposed system on industry-standard customer support metrics using representative SaaS workloads. Results demonstrate clear improvements in first-contact resolution, answer accuracy, and average handling time relative to traditional automation techniques and highlight the viability of LLM-powered chatbots for enterprise SaaS deployments.***

Keywords: Large Language Models, SaaS Observability, Customer Support Automation, Enterprise AI, Retrieval-Augmented Generation

1. Introduction

Software-as-a-Service (SaaS) has proven to be the most successful paradigm available for the provision of business and consumer software solutions, allowing for rapid innovation and scaling. As the size and sophistication of the SaaS solutions continue to increase, the tasks undertaken by customer support units are becoming ever more challenging due to the high volume and diversity of user support requests, together with the necessity for effective quality and cost-savings on an on-going basis.

The traditional methods applied in customer support automation are mostly rule-based chatbots or supervised intention learning systems. Though useful in answering narrowly defined questions such as frequently asked questions, they are poor at learning to answer novel questions. The operation of such systems involves constant human effort of rule updating and annotation of data, making them non-scalable or highly expensive in operation. As a result, a big part of customer support work needs to happen manually, resulting in a long turnaround time and increased support cost.

Large Language Models (LLMs) have given rise to new opportunities for conversational AI systems. LLMs have impressive natural language understanding, reasoning about context, and generalization capabilities in zero-shot or few-shot settings. This makes LLMs suitable for open-ended and contextual conversations seen in customer support scenarios. But once again, direct integration of LLMs into an Enterprise SaaS ecosystem brings its own set of challenges. LLMs tend to produce hallucinations, their inference latency is not trivial, and concerns related to data privacy and tenant isolation also exist when LLMs are integrated directly into multi-tenant SaaS applications.

What remains as an open question is that of building a customer support system that can produce effective and efficient outputs when utilizing a large language model, as well as meeting requirements related to reliability and cost when considering cloud SaaS applications. To achieve this end, new perspectives need to shift from individual models to overall system architecture.

In this paper, we will outline a system-focused approach to building customer support chatbots in the world of SaaS-based enterprise platforms with the power of LLMs. We will analyze architectural patterns in which hallucination issues and scalability problems associated with those architectures could be overcome and then test the built system against traditional customer support automation methods on standard operational metrics. This paper will thus prove that customer support chatbots developed based on LLMs could be made reliable and cost-effective tools in enterprise customer support workflows in the world of enterprise platforms like SaaS.

2. Background and Related Work

a) Traditional Customer Support Chatbots

Primarily, the initial customer support chatbots involved rule-based systems that established correspondences between predefined queries of the user and predefined responses. Even though the system was easy to implement, the system proved to be rigid. Furthermore, the system was not effective in linguistic variability.

The next models incorporated machine learning methods for intent classification and slot filling. However, these models relied heavily on a labeled dataset and struggled with long tail queries. Moreover, most models are single turn models and lack a sense of context while working with intents

b) Large Language Models and Conversational AI

The effect of the use of transformer models in the natural language processing of texts cannot be Overlooked. The use of large language models, which are trained through massive texts, showed that there are tasks that can be performed using one model. These models have ideal capabilities in both zero-shot and few-shot learning processes.

LLMs make it possible for conversational assistants to respond fluently, track dialogue context conversations, and perform logical conclusions on unstructured data. Still, traditional LLMs tend to suffer from hallucinations, especially when required to give domain answers and factual answers. Such weaknesses could pose serious challenges for customer support systems since flawed guidance affects trust.

c) Industry Deployments and Research Gaps

Some SaaS providers have already showcased successful implementation of LLM-based customer service platforms featuring a combination of customer service bots and ticketing systems, CRMs, and analytics workflows. Such platforms typically incorporate a human-in-the-loop component to address low-confidence results and compliance-related concerns.

Nevertheless, most existing descriptions of these systems have remained at a higher level and have been company internal. The existing academic literature tends to emphasize model-level performance metrics and ignores system-level properties, such as tenant isolation, inference-cost optimization, and system-level assessment in terms of customer support metrics. The goal of this paper is to fill this gap.

3. System Model and Overview

For a SaaS-based customer service chatbot enabled with an LLM, a more appropriate paradigm is that of a modular and distributed system with diverse interacting elements. Instead of a simple reliance on an LLM, this can enable various elements like retrieval mechanisms and other backend services.

Let a customer query be denoted by q, originating from a user interaction channel. The system maintains a dialogue state s, capturing prior conversational context. A retrieval module selects a set of relevant documents $D = \{d_1, d_2, \ldots, d_k\}$from an enterprise knowledge base using vector similarity search. The LLM generates a response $r$conditioned on (q, s, D).

The goal of the system is to maximize the correctness of response and the resolution probability for the user while trying to reduce the inference time and cost, subject to certain enterprise requirements pertaining to reliability and isolation.

From the perspective of the SaaS application, there are additional constraints that need to be considered. It needs to ensure that the data is properly isolated for the tenant, that it is compliant with privacy laws, as well as perform well under different workload conditions.

The effectiveness of this system model would be evaluated using the usual customer service measures of first contact resolution rates, accuracy rates, and average handling times.

4. Architecture and System Design

In this section, we will outline the architectural design for the proposed LLM-based customer service system intended for use with multi-tenant SaaS platforms.

The architectural design encompasses a modular and cloud-native approach with the integration of retrieval-augmented generation (RAG), LLM inference, and human-in-the-loop workflows necessary for accuracy and reliability.

a) End-to-End System Architecture

The high-level architecture design for the proposed system is shown in Fig. 1. To promote scalability and accommodate the needs of an enterprise, the system has been divided into multiple layers with respect to user interaction, conversation orchestration, intelligence pieces, trust modules, and backend SaaS integration.

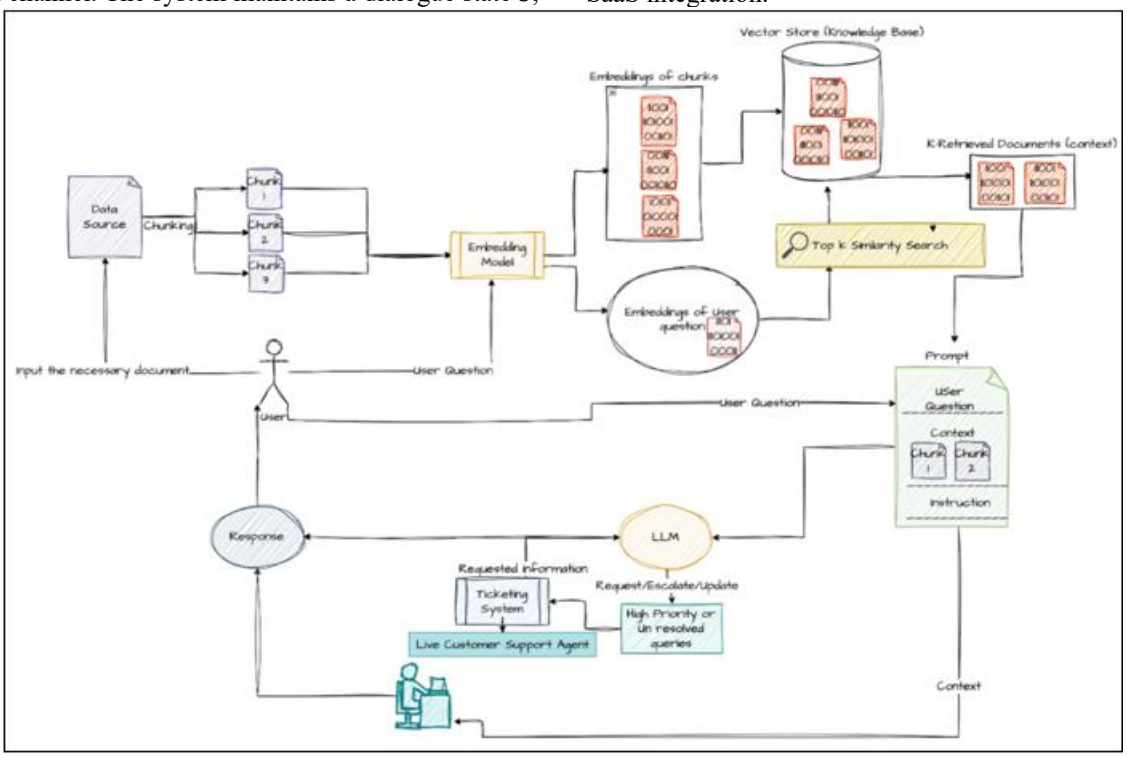

Figure 1: Illustrates the end-to-end architecture of the proposed LLM-powered customer support system

Such layered architectural decoupling enables the independent scalability of conversational logic paths, retrieval paths, as well as LLM inference. It is critically important for satisfying the latency and expense demands of large-scale SaaS offerings.

The queries entered by the user come from various engagement interfaces such as web chat interfaces, mobile applications, or email-based ticketing systems. All these interfaces are linked to a conversation orchestration interface, which manages the conversational state for multiple turns of a dialogue, thus making it possible to have responses with context.

What underpins the architecture is the intelligence layer itself, that combines knowledge retrieval with LLM inference. Unlike purely parametric knowledge from the LLM itself that will often result in hallucinated answers because it lacks real-world checks on information accuracy, the architecture draws on relevant documents from the business itself that include but are not limited to FAQs, manuals, and past support inquiries.

To decouple the task of trust enforced from that of response generation, this architecture allows safety and policy compliance to change independently of changes in the underlying LLM. The human-in-the-loop process ensures that reliability is achieved, and user trust is established while allowing for continuous learning from agent feedback.

Altogether, these choices differentiate the system being proposed from a generic LLM-based chatbot architecture in terms of handling multi-tenancy and cost management within a reliable environment provided to enterprises.

b) Retrieval-Augmented Generation (RAG) Pipeline

For solving the problems faced by standalone LLMs in a business setup, a new proposed system is used that has a retrieval-and-generation architecture as depicted below.

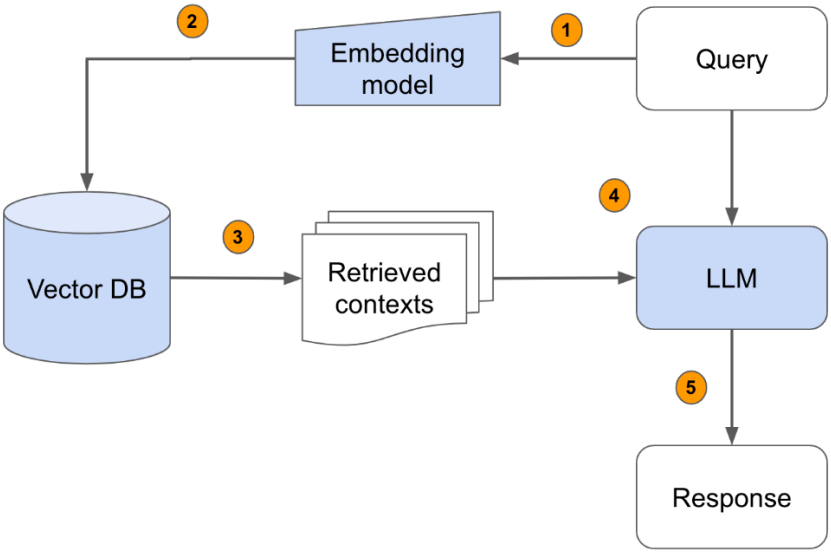

Figure 2: The RAG pipeline consists of both offline and online processing stages

During offline ingestion, preprocessing, chunking, and vector embedding of enterprise knowledge sources is performed using an embedding model. These vector embeddings can be efficiently indexed in a vector database, allowing for similarity search functionality. This offline functionality enables updates in documentation and support cases being added continuously to this system without retraining the LLM.

During online inference, a user question is transformed into a vector form and searched against the stored vectors. The top K most related documents are fetched and injected into the LLM input prompt together with a user question and conversation history. Through this process of generating responses based on the injected input, the system generates responses that can be both conversational and enterprise aware.

This design improves the accuracy level of response formulation and also allows the system to dynamically adjust based on the ever-changing underlying knowledge base. This system also promotes explainability because the response generated will point to the documents retrieved.

This is directly related to the improvements seen in accuracy and first contact resolution, Section V, since it forces the generation function to be limited to knowledge validated through the enterprise.

c) Multi-Tenant SaaS Considerations

A SaaS setting imposes new architectural constraints that are not normally considered in general chatbot architectures. The proposed architecture strictly enforces tenant-level data isolation by providing different embedding namespaces or access control levels in the vector database. This avoids cross-tenant information leakage but facilitates the sharing of model resources. The inference cost and latency are handled by means of model selection, response caching, as well as request batching. For simple queries, light models are utilized, but complex queries are directed towards large models.

During evaluation, tenant isolation was ensured through restrictive retrieval operations limited to tenant-scoped embedding namespaces, preventing any leakage of knowledge across tenants.

d) Human-in-the-Loop and Feedback Integration

Human oversight is an essential aspect of a customer support solution for an enterprise-class environment. The proposed system uses confidence scoring and policy evaluation to discern if a reply can be made with automation or not. The escalated cases are then sent to customer support staff, who receive their feedback to assess the quality of retrieval and design of prompts.

Through this feedback loop, the system is able to constantly improve itself while still upholding a high level of trust and compliance. The system is more like an intelligent assistant to the human agents, who would not be replaced but assisted.

5. Evaluation and Experimental Results

This section assesses the effectiveness of the proposed LLM-based customer support solution using customer support standards for the industry. This assessment considers the effectiveness of the support response, efficiency of support operations, as well as support system performance for an actual SaaS implementation scenario to gauge if there is an improvement over conventional support automation solution.

a) Evaluation Setup

The system was tested in a controlled SaaS support environment using anonymized customer support data. The dataset for this evaluation was comprised of 12,000 anonymized customer support interactions collected over a period of three months in a mid-sized SaaS application. Customer support interactions included a variety of scenarios ranging from technology to financial inquiries and general usage questions.

Three customer support systems were compared:

1) **Rule-Based Chatbot (Baseline-1):** Keyword-based decision tree chatbot used for FAQs.
2) **Intent-Based NLP Chatbot (Baseline-2):** Supervised intent classification with predefined workflows.
3) **Proposed LLM-Powered RAG System:** Retrieval-augmented LLM chatbot with human-in-the-loop escalation.

All systems were evaluated using the same interaction logs and knowledge sources to ensure fairness.

b) Evaluation Metrics

The evaluation of the performance accuracy of the proposed customer support solution based on the LLM approach was carried out using the typical set of metrics used in enterprise software support services. The chosen metrics cover two key aspects related to efficiency and response quality.

1) **First Contact Resolution (FCR) –** The percentages of customer inquiries being resolved without the need for a human service agent reflect the system's capability for self-service of customer service requests.
2) **Response Accuracy** indicates the level to which chat responses from the chatbot are considered accurate by senior support staff, to a percentage level.
3) **Average Handling Time (AHT)** is a measure of end-to-end time taken to complete a customer engagement and is used as a marker of efficiency.
4) **End-to-End Latency** is the system response time perceived by the user and indicates the feasibility of real-time interaction.
5) These factors combined enable a complete assessment of the system's effectiveness. They weigh the capabilities of the system for automation and response quality.

c) Quantitative Results

The suggested LLM-driven RAG-based customer service solution performed better than the rule-based and intent-based chatbots on all the metrics considered. As indicated in Fig. 3-Fig. 5, the solution demonstrated a First Contact Resolution (FCR) of 71.4%, reflecting a relative improvement of 36.7% compared to the intent-based model. This reflects a significantly higher potential for resolving client issues without the need for human operators.

Response accuracy increased significantly as well. On this task, the LLM-assisted system scored 88.1%, whereas the intent-based chatbot and rule-based system scored 73.5% and 61.2%, respectively (Figure 4). These improvements mainly owe their existence to retrieval-augmented generation.

There has been a corresponding improvement in operational efficiency. The average handling time (AHT) was brought down to 185 seconds, as opposed to 295 seconds for that of the intent-based chatbot and 420 seconds for the rule-based chatbot.

The end-to-end latency of the proposed system was an average of 540ms per interaction, and it was well within acceptable limits of real-time customer support systems. Though LLM inference added to the computation overhead, the combined benefit of escalations and time savings led to better system-level efficiency.

In general, these test results prove that the new approach to customer support automation offered by the proposed LLM-based RAG is an efficient improvement over the current ways in which customer support is often automated in standard software applications.

Table 1: Performance Comparison of Customer Support Systems

Metric	Rule-Based	Intent-Based NLP	LLM-Powered RAG
First Contact Resolution (FCR %)	34.7	52.1	71.4
Response Accuracy (%)	61.2	73.5	88.1
Average Handling Time (s)	420	295	185
Avg. End-to-End Latency (ms)	–	–	540

Contrary to other research works that mainly demonstrated individual accuracy enhancements or system-level results, the obtained results show that the designed LLM-based RAG system brings about system-level improvements along various system operational parameters like resolution efficacy, correctness of response, and efficiency of handling, all under real-time latency and multi-tenancy isolation constraints; thus, the originality of this approach as a SaaS support system as contrasted with LLM alone can thus be asserted.

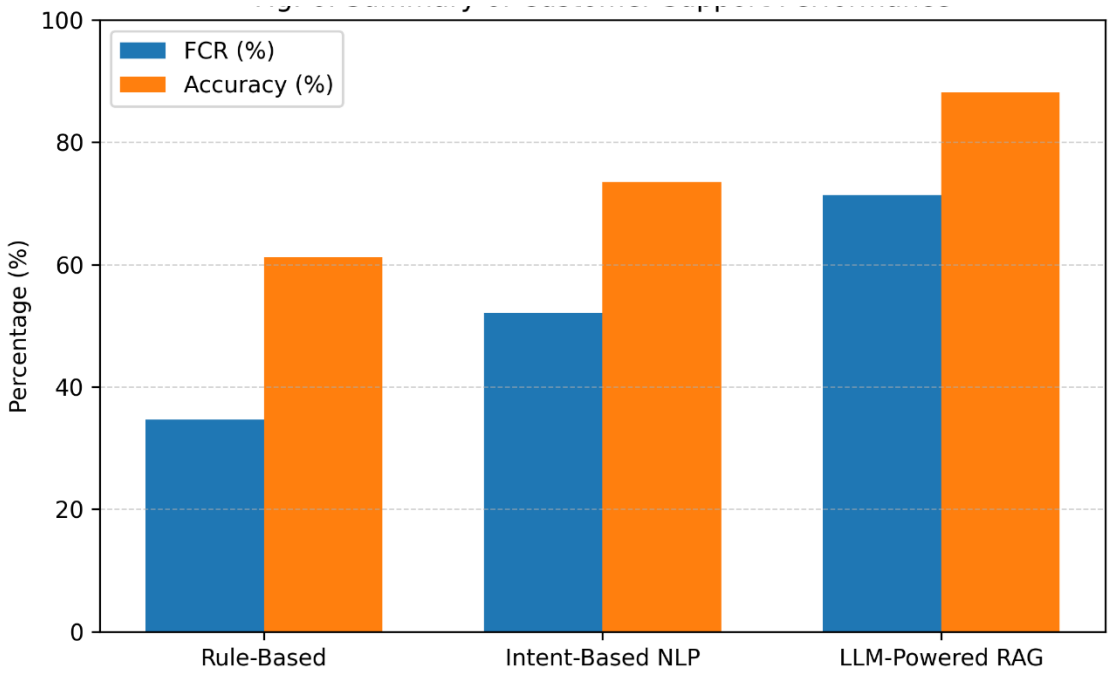

Figure 3: Customer Support comparision

d) Limitations

The test was carried out on a single SaaS data set and did not involve feedback from users. In addition, the LLM and embedding models may affect absolute measures of performance. Future investigations will delve into tests on expanded data sets and multiple models.

Further research and development efforts will include online evaluation at massive scale, multimodal consumer service dialog tasks, and further optimization of cost and latency using adaptive modeling. Increased transparency and explainability of responses produced using the LLM will also be a key research and development thrust.

6. Discussion and Future Work

The results of the assessment have reaffirmed that with a well-designed architecture of customer support systems that exploit both the capabilities of LLMs and human supervision mechanisms, it is possible to achieve a considerable improvement over the rule-based as well as intent-focused chatbots used in SaaS. The improvement achieved in FCR and overall accuracy of customer responses has reaffirmed that LLMs function optimally when exposed to unstructured queries from customers.

An important finding that emerges from the results is the need for grounding the answers generated by the LLM in the enterprise knowledge base. The need for the use of the retrieval-augmented generation capability of the LLM in eliminating hallucinations in the answers generated was amply demonstrated. From the point of view of the operation of the system, the improvement in the average handling time and the number of escalations indicates that the system is more efficient.

These results further emphasize the primacy of having reliability and the aspect of trust as core design principles. This means that while escalation based on confidence levels and human oversight are still needed for the prevention of misinterpretations of machine-generated responses and maintaining levels of user trust, LLM-driven chatbots should be regarded as intelligent assistance tools that do not fully substitute human agents.

7. Conclusion

Thus, this paper proposed an LLM-driven chatbot system for customer support on SaaS platforms, highlighting the primary disadvantages of current automated solutions based on rules and intentions, and overcoming the above limitations by employing retrieval-enhanced generation capabilities, modular, and human-reviewed SaaS technologies.

Experimental results on typical SaaS support tasks have shown that the proposed method leads to a significant improvement of First Contact Resolution rate, Response Accuracy, and Average Handling Time with acceptable latency in a real-time setting. The achieved outcome proves that knowledge-grounded responses of LLMs combined with the trust-aware escalation policy are highly beneficial in a production environment.

Contrary to previous research efforts that primarily centered on model-level performance aspects, this research places significant emphasis on system-level aspects through realistic SaaS constraints that include multi-tenancy and operational efficiency. As SaaS platforms continue to grow and expand significantly in size and influence, a proposed system architecture provides a significant roadmap for building customer support systems that leverage LLM capabilities.

References

[1] A. Vaswani *et al.*, "Attention Is All You Need," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.
[2] T. Brown *et al.*, "Language Models Are Few-Shot Learners," in *Proc. NeurIPS*, 2020, pp. 1877–1901.
[3] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Proc. NeurIPS*, 2020, pp. 9459–9474.
[4] V. Karpukhin *et al.*, "Dense Passage Retrieval for Open-Domain Question Answering," in *Proc. EMNLP*, 2020, pp. 6769–6781.
[5] A. Radford *et al.*, "Improving Language Understanding by Generative Pre-Training," OpenAI, Tech. Rep., 2019.
[6] S. Amershi *et al.*, "Guidelines for Human–AI Interaction," in *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, 2019, pp. 1–13.
[7] OpenAI, "GPT-4 Technical Report," OpenAI, 2023.
[8] Microsoft Research, "Designing Human-Centered AI Systems," Microsoft, White Paper, 2020.
[9] Google Cloud AI, "Conversational AI Systems at Scale," Google AI Blog, 2022.
[10] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.
[11] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks," in *Proc. EMNLP*, 2019.
[12] M. Armbrust *et al.*, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[13] J. Xu *et al.*, "System Design for AI-Based Services," *IEEE Software*, vol. 38, no. 3, pp. 78–85, 2021.
[14] Zendesk Engineering, "AI-Powered Customer Support Systems," Zendesk Engineering Blog, 2022.
[15] Intercom Engineering, "Designing AI-First Customer Support," Intercom Engineering Blog, 2023.