Architecting a Low-Latency RAG System for Fast-Moving Consumer Goods (FMCG) Customer Support: A Case Study in Industrial Software Deployment

Meredita Susanty, Alghifari Rasyid Zola
Ilmu Komputer, Fakultas Sains dan Ilmu Komputer, Universitas Pertamina, Indonesia

***Abstract*—Deploying large language models (LLMs) in industrial customer support environments require balancing response accuracy with system latency. This study presents the software architecture and implementation of a Retrieval-Augmented Generation (RAG) system designed for the Fast-Moving Consumer Goods (FMCG) sector. Addressing the limitations of generic LLMs in domain-specific knowledge tasks, we engineered a retrieval-augmented inference pipeline that integrates unstructured data ingestion, Pinecone for vector indexing, and Groq-based inference for low-latency response generation. The proposed system aims to improve response grounding by incorporating organizational product information into the generation process while maintaining responsive interaction times suitable for customer support applications. This study details the software architecture, system integration approach, and experimental evaluation of the proposed deployment-oriented RAG framework in an industrial FMCG case-study setting.**

***Keywords*—Software architecture; RAG; industrial AI; Groq LPU; vector database; customer support systems**

I. Introduction

Large language models (LLMs) have become increasingly popular in customer service applications due to their ability to generate natural language responses and assist users across a wide range of tasks [1]. Organizations are progressively exploring LLM-based solutions to improve service quality, reduce operational workload, and provide faster access to information [1], [2]. In the Fast-Moving Consumer Goods (FMCG) sector, customer support teams frequently handle inquiries related to product specifications, ingredients, usage instructions, and company procedures. As product portfolios evolve rapidly and organizational knowledge changes continuously, maintaining accurate and consistent responses across customer support channels becomes a significant challenge [2].

Despite their capabilities, general-purpose LLMs exhibit fundamental limitations when applied to enterprise-specific domains. These models primarily rely on parametric knowledge acquired during pre-training and therefore lack direct access to proprietary, confidential, or recently updated organizational information [3]. Consequently, when queried about domain-specific products, internal procedures, or newly released information, LLMs may generate plausible yet factually incorrect responses, a phenomenon commonly referred to as hallucination [4]. In customer support settings, such inaccuracies may negatively affect customer trust, service consistency, and operational effectiveness [5].

Retrieval-Augmented Generation (RAG) has emerged as a promising approach for mitigating these limitations by integrating external knowledge sources into the response generation process [6]. Instead of relying solely on internal model parameters, RAG systems retrieve relevant information from an external knowledge repository and provide this information as contextual evidence for generation [6]. This architecture enables organizations to update knowledge independently of the language model itself, making RAG particularly suitable for dynamic business environments where information changes frequently [7].

Although RAG can improve factual grounding, the architecture introduces additional computational stages, including query processing, retrieval, context assembly, and response generation [2]. These stages increase overall response latency compared to direct LLM inference. In customer support scenarios, where users typically expect responses within only a few seconds, excessive latency may reduce perceived responsiveness and negatively impact user experience [2]. Therefore, practical deployment of RAG systems requires balancing response quality with operational performance [7].

Recent advances in inference infrastructure have created opportunities to improve the responsiveness of Retrieval-Augmented Generation systems. Among these developments, Groq's inference platform has attracted attention for its ability to provide low-latency language model inference and high token-generation throughput. While previous studies have extensively investigated retrieval effectiveness and answer quality in RAG systems [6], comparatively less attention has been given to deployment-oriented architectures that simultaneously consider retrieval-based grounding and low-latency response requirements for enterprise customer support applications [7].

This study presents the design and evaluation of a low-latency RAG architecture for FMCG customer support. The proposed system integrates document ingestion, vector-based knowledge retrieval using Pinecone, and LLM inference served through Groq's infrastructure. Rather than proposing a novel retrieval algorithm, this work focuses on the architectural design and practical implementation of a retrieval-augmented customer support system using real industrial knowledge assets [7]. The objective is to investigate whether retrieval-based grounding can improve response accuracy while maintaining latency suitable for interactive customer support scenarios [2].

The main contributions of this study are summarized as follows:

- The design and implementation of a Retrieval-Augmented Generation architecture for FMCG customer support applications.
- An empirical evaluation of response accuracy and end-to-end latency using domain-specific customer support queries and proprietary organizational knowledge.
- An empirical comparison between a local GPU-based deployment and a Groq-based deployment within a retrieval-augmented customer support pipeline.
- A discussion of practical implementation considerations, limitations, and lessons learned from developing a deployment-oriented RAG system for enterprise use cases.

II. Related Work

A. Fine-Tuning and Retrieval-Augmented Generation

Several approaches have been proposed to adapt large language models (LLMs) to domain-specific applications [3]. One widely adopted approach is fine-tuning, where pre-trained models are further trained on task-specific or domain-specific datasets to improve performance in specialized contexts. Previous studies have demonstrated that fine-tuning can improve downstream task performance by incorporating domain knowledge into model parameters [3].

However, fine-tuning presents several challenges in enterprise environments. Updating organizational knowledge typically requires additional training cycles, computational resources, and model maintenance [7]. As organizational information evolves over time, maintaining an up-to-date fine-tuned model can become costly and operationally complex. Furthermore, fine-tuned models may continue to rely on outdated information embedded within model parameters when knowledge sources change frequently [7].

To address these limitations, Retrieval-Augmented Generation (RAG) was introduced as an alternative architecture that separates knowledge storage from language generation [6]. Instead of embedding all knowledge within model parameters, RAG retrieves relevant information from an external knowledge source and incorporates the retrieved context into the generation process [6]. Similar findings were reported by Guu et al. [8], who demonstrated that retrieval-enhanced architectures can improve factual grounding while maintaining the reasoning capabilities of large language models.

Recent studies have further highlighted the advantages of retrieval-based systems for knowledge-intensive applications [6]. Kandpal et al. [9] observed that large language models often struggle to accurately recall long-tail or domain-specific information, reinforcing the importance of external retrieval mechanisms for enterprise knowledge management scenarios.

B. Retrieval-Augmented Generation in Enterprise Knowledge Systems

The adoption of Retrieval-Augmented Generation has expanded beyond academic benchmarks into practical enterprise applications [6]. Organizations increasingly utilize RAG architectures for knowledge management, document question answering, internal search systems, and customer support automation [2]. By retrieving relevant documents from external repositories, RAG systems can provide responses grounded in organizational knowledge while reducing the likelihood of hallucinated content [6].

Gao et al. [7] identified document chunking, retrieval quality, and context construction as critical components influencing overall RAG performance. Similarly, several enterprise-oriented implementations have demonstrated the effectiveness of vector databases and retrieval pipelines in improving access to large collections of unstructured organizational information [5].

Despite these advances, most prior work focuses primarily on retrieval effectiveness and answer quality [6], [7]. Comparatively fewer studies investigate deployment-oriented considerations such as inference latency, architectural integration, and operational constraints encountered in enterprise environments [2]. As a result, understanding how retrieval-enhanced systems can be implemented while maintaining acceptable response times remains an important area of research.

C. Low-Latency LLM Inference Architectures

While retrieval can improve factual grounding, additional processing stages introduce latency into the overall system [6]. Consequently, recent research has increasingly focused on optimizing large language model inference to support real-time applications [10].

Traditional GPU-based deployments remain the dominant approach for serving transformer-based language models. To improve inference efficiency, several optimized serving frameworks have been proposed, including vLLM and TensorRT-LLM. These frameworks improve throughput and memory utilization through techniques such as paged attention, kernel optimization, efficient batching strategies, and optimized execution pipelines [10].

In parallel, specialized inference hardware has also emerged as an alternative approach for reducing inference latency. One notable example is Groq's tensor streaming architecture, which employs deterministic execution and compiler-driven optimization to improve token generation throughput and reduce inference overhead. Such approaches aim to improve responsiveness for interactive AI applications where low latency is a critical requirement [10].

Although these approaches significantly improve inference performance, existing studies primarily evaluate hardware acceleration or serving optimizations independently from retrieval-enhanced architectures. Limited research has examined how low-latency inference infrastructures interact with end-to-end Retrieval-Augmented Generation pipelines in enterprise customer support settings [6], [2].

D. Research Gap

Based on the reviewed literature, prior studies have extensively investigated the benefits of retrieval augmentation for improving factual accuracy and reducing hallucinations. Likewise, substantial efforts have been devoted to accelerating

large language model inference through specialized hardware and optimized serving frameworks.

However, limited attention has been given to deployment-oriented architectures that simultaneously address retrieval-based knowledge grounding and low-latency response requirements in enterprise customer support environments. Furthermore, relatively few studies report practical implementation experiences using real organizational knowledge assets within industrial case-study settings.

Therefore, this study focuses on the design and evaluation of a low-latency Retrieval-Augmented Generation architecture for FMCG customer support. Rather than proposing a new retrieval algorithm or language model, the work investigates how retrieval-based grounding and different deployment configurations can be combined within a practical enterprise-oriented architecture to improve response quality while maintaining interactive response times.

III. Methods

This study adopts a Retrieval-Augmented Generation (RAG) architecture designed to support domain-specific question answering within FMCG customer support scenarios. The proposed system combines document preprocessing, vector-based retrieval, prompt augmentation, and large language model inference into a unified pipeline. The methodology consists of four primary stages: knowledge base construction, retrieval pipeline development, response generation, and performance evaluation.

A. System Overview

The overall architecture of the proposed system is illustrated in Fig. 1. The system follows a modular design consisting of four primary components: document ingestion, vector retrieval, context augmentation, and language model inference.

During the offline phase, organizational documents are processed and transformed into retrieval-ready chunks. These chunks are subsequently converted into vector representations and indexed within a Pinecone vector database. During the online inference phase, user queries are transformed into vector representations using the same procedure and submitted to the retrieval system. The most relevant document chunks are retrieved and incorporated into a structured prompt before being passed to the language model for response generation.

This architecture separates knowledge storage from reasoning, allowing organizational knowledge to be updated without modifying the underlying language model.

B. Knowledge Base Construction and Chunking Strategy

The knowledge base used in this study consists of proprietary FMCG product information documents from one brand within a portfolio of fourteen FMCG brands. The selected brand contains a comprehensive collection of product specifications, ingredients, product descriptions, and usage-related information commonly referenced during customer support interactions. This single-brand repository was used as an industrial case-study dataset to evaluate the proposed Retrieval-Augmented Generation architecture in a realistic customer support scenario.

Prior to indexing, all documents underwent a preprocessing stage to generate retrieval-ready chunks. As illustrated in Fig. 2, documents were initially segmented into paragraphs, after which non-informative segments and excessively short paragraphs were removed. Longer paragraphs were subsequently divided into smaller sentence groups to improve retrieval granularity while preserving local contextual coherence. This chunking strategy was designed to reduce context fragmentation and increase the likelihood that retrieved chunks contain complete product-related information. Each resulting chunk was associated with relevant metadata and prepared for vector indexing within the retrieval pipeline.

C. Vector Representation and Retrieval Pipeline

Following document segmentation, each chunk was transformed into a fixed-length vector representation to support similarity-based retrieval. The current implementation employs a lightweight hash-based vectorization approach rather than a transformer-based sentence embedding model. Individual lexical tokens are mapped into a 1024-dimensional vector space using a deterministic hashing function, producing a fixed-length representation for each document chunk. The resulting vectors are subsequently normalized prior to similarity comparison. This approach was selected due to its implementation simplicity and compatibility with the lightweight retrieval architecture adopted in this study. This choice was made to align with the lightweight prototype implementation, where vectorization was performed directly within the application service without requiring a separate neural embedding inference component. Consequently, the retrieval layer could be deployed with minimal additional infrastructure and computational overhead.

Because the adopted representation is primarily based on lexical features, it captures token-level similarity rather than the semantic relationships typically learned by neural embedding models. Consequently, retrieval performance may be affected when semantically similar queries exhibit substantially different lexical forms. The generated vectors are indexed using Pinecone as the retrieval backend. During inference, incoming user queries undergo the same vectorization procedure before similarity search is performed against the indexed knowledge base. For the experiments reported in this study, the retrieval component returns the Top-K most relevant document chunks, where $K = 15$. These retrieved chunks are subsequently incorporated into the prompt construction stage and supplied to the language model as contextual evidence for response generation.

D. Retrieval-Augmented Response Generation

After retrieval, the selected document chunks are assembled into a structured prompt and supplied to the language model for response generation. The retrieved information serves as contextual evidence, allowing the model to generate responses grounded in the available product knowledge rather than relying solely on its internal parametric knowledge.

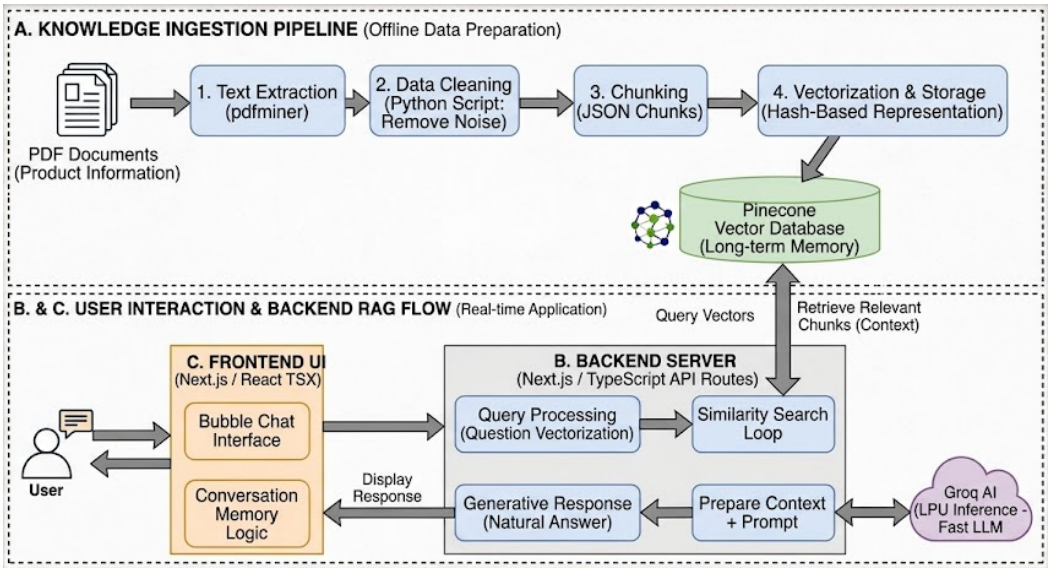

Fig. 1. Overall architecture of the proposed Retrieval-Augmented Generation pipeline.

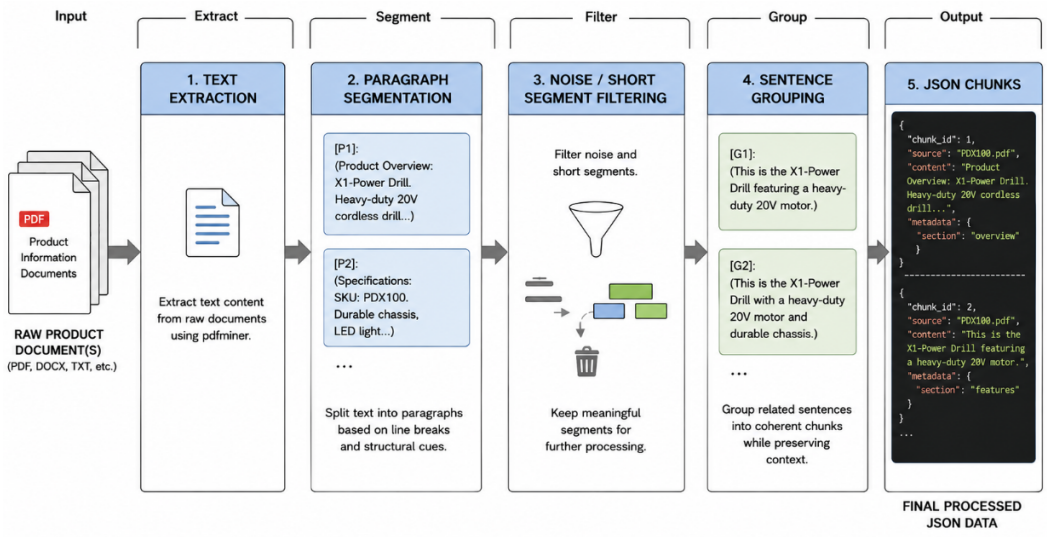

Fig. 2. Knowledge base construction and chunking strategy used for document preparation.

To improve factual consistency, the system adopts a closed-context prompting strategy. As illustrated in Fig. 3, the prompt explicitly instructs the model to generate responses only from the retrieved information and to avoid introducing unsupported content beyond the available context. This approach aims to reduce hallucinations by constraining the generation process to the evidence provided by the retrieval stage.

The overall inference workflow is illustrated in Fig. 4. Upon receiving a user query, the system performs retrieval, prompt construction, and response generation sequentially before returning the final response to the user. For language generation, this study utilizes the Llama-3.3-70B-Versatile model served through Groq's inference infrastructure. A temperature value of 0 was selected to improve output consistency and reduce response variability, while the maximum output length was limited to 600 tokens.

Groq's inference platform was selected due to its low-latency serving capabilities and compatibility with interactive question-answering applications. The objective of incorporating this infrastructure is to evaluate its contribution to overall response latency within the proposed retrieval-augmented customer support pipeline rather than to isolate the effects of hardware acceleration independently.

E. Experimental Environment

To evaluate the practical performance of the proposed architecture, two deployment configurations were considered in this study: a local GPU-based deployment and a Groq-based deployment. The proposed configuration utilized the Llama-3.3-70B-Versatile model served through Groq's cloud inference platform, while the baseline configuration employed a locally deployed language model served through Ollama and

```
System: You are an expert Customer Support Agent for
Crystallure.

Instructions:
1. Answer the user's question based ONLY on the information
provided below.
2. Read all information carefully before answering.
3. If the answer is not present in the information, strictly
reply: "Maaf, tidak ada informasi yang tersedia untuk
menjawab pertanyaan ini."
4. Do not use outside knowledge or training data.
5. Extract exact values (ml, gram, gr, g, Rp, %, etc.) when
present.
6. For product list questions, extract all unique product
names from the metadata.
7. For ingredient questions, provide complete ingredient
lists from the information.
8. Use conversation context to understand references like
"itu", "tadi", "sebelumnya" when available.

INFORMASI:
{retrieved_chunks}

PERTANYAAN: {user_query}

KONTEKS PERCAKAPAN:
{conversation_context}

INSTRUKSI:
- Cari angka/nilai yang tepat (ml, gram, gr, g, Rp, %, dll)
- Jika ada angka di awal kalimat seperti "12 gr" atau "150
ml", SEBUTTKAN angka tersebut
- Contoh: "12 gr Bedak..." berarti beratnya adalah 12 gr
- Untuk pertanyaan "apa aja produk crystallure", CARI semua
nama produk yang unik dari informasi yang diberikan dan buat
daftar lengkap
- Untuk pertanyaan "ingredientsnya" atau "kandungannya",
berikan daftar lengkap ingredients dari produk yang sedang
dibicarakan
- Jika ada konteks percakapan, gunakan untuk memahami
referensi seperti "itu", "tadi", "sebelumnya"
- Berikan jawaban yang natural dan mengalir berdasarkan
konteks percakapan
- PENTING: Untuk daftar produk, ekstrak semua nama produk
unik dari metadata "Product" yang ada di informasi
- JANGAN katakan "Informasi tidak tersedia" kecuali benar-
benar tidak ada informasi sama sekali

JAWABAN:
```

Fig. 3. The closed-context prompt structure.

executed on consumer-grade GPU hardware.

The local deployment environment consisted of an NVIDIA RTX 3060 GPU with 16 GB of system memory. Both deployment configurations utilized the same retrieval pipeline, including the Pinecone vector database, document chunking strategy, retrieval parameters, and prompt construction mechanism. This configuration was intended to ensure that the retrieval process remained consistent across experiments, allowing performance differences to be evaluated at the deployment level rather than the retrieval level.

For retrieval, the system employed a Top-K configuration of $K = 15$. Retrieved document chunks were incorporated into a closed-context prompt before response generation. All responses were generated using a temperature setting of 0 and a maximum output length of 600 tokens. These parameters were maintained throughout the evaluation process to ensure consistency in both latency and response quality measurements.

It should be noted that the comparison was designed to evaluate the practical latency characteristics of two deployment approaches within the same Retrieval-Augmented Generation workflow. The objective was not to isolate the contribution of hardware acceleration independently, but rather to assess the overall responsiveness of a locally deployed GPU-based configuration compared with the proposed Groq-based deployment in an interactive customer support scenario.

F. Evaluation Framework

The proposed system was evaluated using a domain-specific benchmark consisting of 50 question-answer pairs derived from organizational product information and customer support materials. The evaluation dataset was designed to represent common inquiry categories encountered in FMCG customer support scenarios, including product specifications, ingredients, usage instructions, and general product-related information.

System performance was assessed using two primary dimensions: response accuracy and response latency. Response accuracy was evaluated through manual assessment by comparing generated responses against the corresponding source documents contained within the knowledge base. Responses were classified as correct when the generated answer was factually consistent with the available product information and incorrect when factual inconsistencies or unsupported information were identified.

Latency evaluation focused on end-to-end response time, defined as the elapsed duration between query submission and receipt of the final generated response. Additional measurements included Time-to-First-Token (TTFT) and token generation throughput to assess inference efficiency. Latency measurements were collected under the deployment configurations described in Section III E.

The evaluation dataset was derived from a single FMCG brand and, therefore, reflects the characteristics of a specific industrial case-study setting. Consequently, the reported results should be interpreted as evidence of the practical feasibility of the proposed architecture rather than as a generalized assessment of performance across all FMCG customer support environments. Furthermore, the evaluation was conducted using only 50 test queries and relied on a single evaluator for manual response assessment. Statistical significance testing and inter-rater reliability analysis were not performed within the scope of this study. These limitations should be considered when interpreting the reported results.

Although retrieval quality plays an important role in overall RAG performance, this study did not explicitly evaluate retrieval effectiveness using metrics such as Recall@K, Precision@K, Mean Reciprocal Rank (MRR), or context coverage. Instead, the evaluation focused on end-to-end response accuracy and system latency to assess the practical feasibility of the proposed architecture in a customer support setting. Consequently, the contribution of the retrieval component was evaluated indirectly through overall response quality rather than through dedicated retrieval benchmarks.

IV. Results and Discussion

A. Response Latency Evaluation

To assess the responsiveness of the proposed architecture, a latency comparison was conducted between a local GPU-based deployment and the proposed Groq-based deployment. The evaluation focused on three metrics: Time-to-First-Token (TTFT), token generation throughput, and end-to-end response

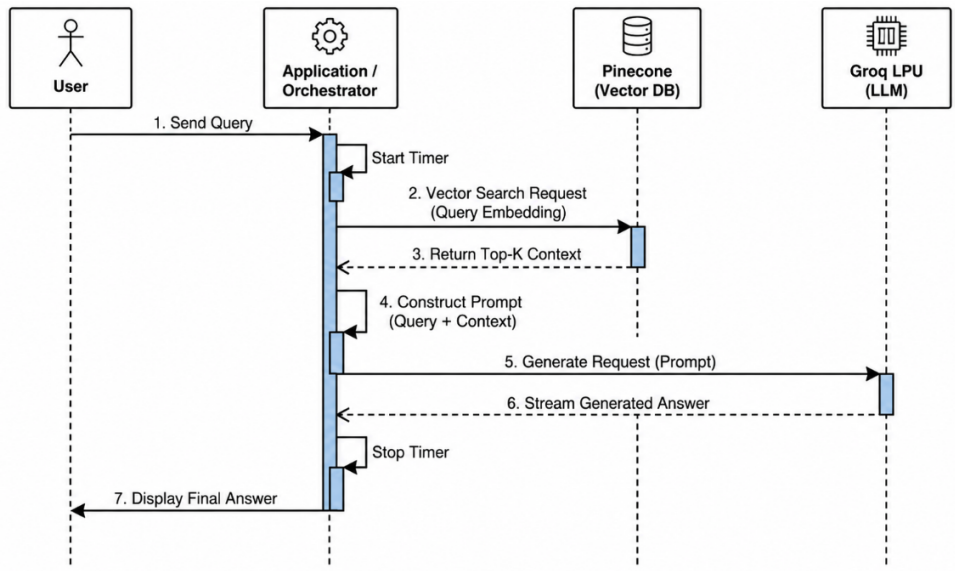

Fig. 4. Sequence diagram of the inference request.

latency. TTFT measures the time required for the first generated token to be produced, throughput measures the token generation rate, and end-to-end latency represents the total elapsed time between query submission and receipt of the final response.

Table I summarizes the latency characteristics observed for both deployment configurations. The local deployment utilized Ollama running on an NVIDIA RTX 3060 GPU, while the proposed deployment employed the Llama-3.3-70B-Versatile model served through Groq's cloud inference platform.

TABLE I. LATENCY COMPARISON BETWEEN LOCAL GPU-BASED AND GROQ-BASED DEPLOYMENTS.

Metric	Local GPU-Based	Groq-Based
Embedding & Retrieval	245 ms	238 ms
Time-to-First-Token (TTFT)	742 ms	118 ms
Token Generation Rate	42.8 tokens/sec	305.9 tokens/sec
Total Response Time	**2847 ms**	**1246 ms**

B. Response Accuracy Evaluation

The response accuracy evaluation was conducted using 50 domain-specific customer support queries derived from organizational product information. Generated responses were manually compared against the corresponding source documents contained within the knowledge base. Responses were classified as correct when the generated answer was factually consistent with the available product information and incorrect when factual inconsistencies or unsupported information were identified.

Table II summarizes the overall response accuracy obtained during the evaluation. Out of 50 evaluated queries, 36 responses were classified as correct, while 14 responses contained inaccuracies or incomplete information. This resulted in an overall response accuracy of 72.0%.

The results indicate that the proposed Retrieval-Augmented Generation architecture was able to provide factually consistent responses for the majority of evaluated customer support queries. The integration of retrieval-based grounding enabled the language model to access product-specific information that was not explicitly contained within its parametric knowledge.

TABLE II. RESPONSE ACCURACY RATE EVALUATION RESULTS

Evaluation Outcome	Count
Correct Responses	36
Incorrect Responses	14
Total Queries	50
Accuracy	**72.0%**

Analysis of incorrect responses revealed that most errors originated from incomplete retrieval results, ambiguity within product descriptions, or situations where relevant information was available in the retrieved context but was not fully incorporated into the generated response. These findings suggest that retrieval quality and context utilization remain important factors influencing overall system performance.

C. Discussion

The experimental results demonstrate that retrieval augmentation can improve the ability of large language models to answer domain-specific customer support queries. The proposed system achieved an overall response accuracy of 72.0% across the evaluated FMCG customer support dataset, indicating that retrieval-based grounding can assist the language model in generating responses that are consistent with organizational product information.

The results suggest that integrating an external knowledge repository enables the language model to access product-specific information that is not explicitly stored within its parametric knowledge. By incorporating retrieved document chunks into the generation process, the system can provide responses grounded in available product documentation rather than relying solely on pre-trained model knowledge.

The latency evaluation further indicates that the Groq-based deployment provides faster response times than the local GPU-based deployment. As shown in Table I, the proposed deployment reduced Time-to-First-Token (TTFT) from 742 ms to 118 ms and decreased end-to-end response latency from 2847 ms to 1246 ms. In addition, token generation throughput increased from 42.8 tokens per second to 305.9 tokens per second. These findings suggest that low-latency inference infrastructure can improve the responsiveness of retrieval-augmented customer support systems.

Despite the observed improvements, the retrieval component remains an important source of performance limitations. The current implementation employs a lightweight hash-based vector representation that primarily captures lexical similarity rather than deep semantic relationships between terms. Consequently, retrieval effectiveness may decrease when semantically similar queries are expressed using substantially different vocabulary. This limitation may partially explain some of the incorrect responses observed during evaluation.

Analysis of incorrect responses revealed that most errors originated from incomplete retrieval results, ambiguity within product descriptions, or situations where relevant information was available in the retrieved context but was not fully incorporated into the generated response. These findings indicate that retrieval quality and context utilization remain important factors influencing overall system performance.

From an operational perspective, the proposed architecture enables organizational knowledge to be updated independently from the language model itself. New product information can be incorporated by updating the knowledge base and vector index without requiring model retraining. This characteristic may be beneficial in FMCG environments where product information and operational procedures change frequently and require timely updates across customer support channels.

D. Limitations and Future Work

Several limitations should be considered when interpreting the findings of this study. First, the evaluation was conducted using a dataset consisting of 50 question-answer pairs derived from product information belonging to a single FMCG brand. Although the selected dataset represents a realistic industrial use case, the reported results may not generalize to larger knowledge repositories, multiple brands, or different organizational environments. Furthermore, response quality was assessed through manual evaluation and did not involve multiple independent annotators. Consequently, inter-rater reliability was not measured, and the evaluation dataset of only 50 query-response pairs was insufficient to support meaningful statistical significance testing. Future studies should expand the evaluation benchmark and incorporate multiple independent annotators to improve evaluation reliability and generalizability.

A second limitation relates to the retrieval component. The current implementation employs a lightweight hash-based vector representation that primarily captures lexical similarity rather than deep semantic relationships between terms. While computationally efficient, this approach may reduce retrieval effectiveness when semantically similar queries are expressed using substantially different vocabulary. In addition, the study focused on end-to-end system performance and did not evaluate retrieval effectiveness using dedicated retrieval metrics such as Recall@K, Precision@K, Mean Reciprocal Rank (MRR), or context coverage. Consequently, the contribution of the retrieval component was assessed indirectly through overall response quality rather than through dedicated retrieval benchmarks.

Controlled ablation experiments were also not conducted within the scope of this study due to the limited experimental setup and available evaluation data. Consequently, the individual contributions of retrieval configuration, prompt design, serving infrastructure, and inference hardware could not be isolated quantitatively. Future work should perform systematic ablation studies to better understand the relative impact of each component on overall system performance.

Finally, the study did not investigate scalability under concurrent user workloads, larger knowledge base sizes, or alternative deployment frameworks such as vLLM and TensorRT-LLM. Future research should address these aspects while also exploring transformer-based embedding models, hybrid retrieval strategies, and reranking techniques to improve retrieval quality and response accuracy. Such investigations would provide a more comprehensive understanding of deployment-oriented Retrieval-Augmented Generation systems in industrial customer support environments.

V. Conclusion

This study presented the design and evaluation of a Retrieval-Augmented Generation (RAG) architecture for domain-specific customer support within the Fast-Moving Consumer Goods (FMCG) sector. The proposed system integrates document preprocessing, vector-based retrieval, closed-context prompting, and large language model inference to generate responses grounded in organizational product information. In addition, the study evaluated two deployment configurations, namely a local GPU-based deployment and a Groq-based deployment, to investigate their practical latency characteristics within the same retrieval-augmented workflow.

The experimental evaluation demonstrated that the proposed system achieved an overall response accuracy of 72.0% across 50 domain-specific customer support queries. The results indicate that retrieval-based grounding can assist large language models in providing responses that are more consistent with organizational knowledge sources. In terms of system responsiveness, the Groq-based deployment reduced end-to-end response latency from 2847 ms to 1246 ms, improved Time-to-First-Token (TTFT) from 742 ms to 118 ms, and increased token generation throughput from 42.8 to 305.9 tokens per second. These findings suggest that low-latency inference infrastructure can support responsive customer support interactions while maintaining retrieval-grounded response generation.

The findings should be interpreted within the scope of an industrial case-study setting involving product information from a single FMCG brand. Furthermore, the evaluation dataset consisted of a limited number of manually assessed queries and did not include dedicated retrieval effectiveness metrics, multiple independent annotators, or large-scale scalability experiments. Consequently, the reported results should

be viewed as evidence of practical feasibility rather than a generalized assessment of Retrieval-Augmented Generation performance across all enterprise environments.

Future work should investigate transformer-based embedding models, retrieval effectiveness metrics such as Recall@K, Precision@K, and Mean Reciprocal Rank (MRR), as well as hybrid retrieval and reranking approaches to improve retrieval quality. Additional studies should also evaluate scalability under larger knowledge bases, concurrent user workloads, and alternative deployment frameworks to provide a more comprehensive understanding of deployment-oriented Retrieval-Augmented Generation systems in industrial customer support applications.

ACKNOWLEDGMENT

We express our gratitude to the PT Paragon Technology and Innovation for facilitating this case study, particularly Mrs. Nurul Rizka Hadiningsih for their industrial mentorship. We also appreciate the valuable input provided by the Customer Care team leaders and agents during the system's development and testing.

REFERENCES

[1] S. R. Peddinti, S. R. Katragadda, B. K. Pandey, and A. Tanikonda, "Utilizing large language models for advanced service management: potential applications and operational challenges," *Journal of Science & Technology*, vol. 4, no. 2, 2023.

[2] S. Veturi, S. Vaichal, R. L. Jagadheesh, N. I. Tripto, and N. Yan, "Rag based question-answering for contextual response prediction system," *arXiv preprint arXiv:2409.03708*, 2024.

[3] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–55, 2025.

[4] Z. Xu, S. Jain, and M. Kankanhalli, "Hallucination is inevitable: An innate limitation of large language models," *arXiv preprint arXiv:2401.11817*, 2024.

[5] A. G. Larsen, M. B. Skjuve, A. Følstad, and N. Van As, "Llm hallucinations in conversational ai for customer service: Framework and end-user perceptions," *International Journal of Human–Computer Interaction*, pp. 1–22, 2025.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[7] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[8] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International conference on machine learning*. PMLR, 2020, pp. 3929–3938.

[9] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," in *International conference on machine learning*. PMLR, 2023, pp. 15 696–15 707.

[10] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.