Re-defining Consumer Interactions: Evaluating the Impact of Llama 3.1 and Agentic RAG in the Retail Realm.

Manzur Elahi Shaik
Department of Information Science
University of North Texas
Denton, USA
manzurelahishaik@my.unt.edu

Suprathika Vangari
Department of Information Science
University of North Texas
Denton, USA
suprathikavangari@my.unt.edu

Harish Inavolu
Department of Information Science
University of North Texas
Denton, USA
harishinavolu@my.unt.edu

Sowmya Katla
Department of Information Science
University of North Texas
Denton, USA
Sowmyakatla@my.unt.edu

***Abstract*—The Retailia project works on a retail customer service transformation using the power of new AI technologies, including Agentic RAG and Llama 3.1 to increase answer accuracy, ensuring effective question handling, and thus improving customer satisfaction in general. This will increase the accuracy of the answers, improve the efficiency of the question handling, and improve general customer satisfaction. Llama 3.1 is the most developed LLM that offers rich natural language creation and comprehension, thus allowing conversational and contextually aware dialogues. It fulfills the demand for real-time retrieval and integration using the agentic RAG framework. Specialized agents are used to perform certain activities, such as obtaining data from databases and knowledge bases.**

The key findings of this study indicate that response times to services have improved significantly and Retailia outperformed conventional models in both accuracy and personalization. The system architecture is modular, powered by Agentic RAG, thus allowing the processing of a wide range of client enquiries to proceed smoothly and adjust to changing interaction complexity. This innovative use of Agentic RAG and Llama 3.1 in a retail environment exemplifies the vanguard strategy that integrates real-time data retrieval with AI-powered interaction. A scalable and flexible solution that empowers superior customer engagement and operational efficiency.

The implications are huge for retail, and with Retailia, businesses can finally meet the growing expectations of today's consumers by responding faster, more accurately, and more personally. Along with improving customer satisfaction, this approach reduces operational costs and enables scalable growth. By harnessing the power of Llama 3.1 and Agentic RAG, Retailia sets a new standard for customer service and a path toward a more efficient and satisfying retail customer experience.

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The customer experiences change under rapid transformation, and efficient services become personalized like everything else offered by retail. Retailers stand at the most critical crossroads of their existence and traditional models of customer services work, albeit mostly, do not provide the easiest way because they fail to scale. Barriers on the last leg such as accuracy, do not really solve the inherent problem because of the scale and the accuracy limits [2].

The latest intervention constructs have transformed these models into an infused all-AI technology using LLMs and RAGs at the cutting edge of the research. Demonstrate that an LLM enables much more fluent and contextually suitable interaction in natural language processing. For instance, Llama 3.1 is a suitable candidate for applications requiring customer contacts of very high volumes [10] [17].

RAG frameworks, with their features of combining data retrieval with the ability to generate texts, It has been necessary for the ever-changing needs of retail data. The Agentic RAG method adds a contributory layer efficiency-wise and adaptability-wise [8], because it uses specialized agents to interact specifically for this task. This study investigated the extent to which Retailia could fill the gap and constitute a revolutionary aspect in retail customer service by integrating Llama 3.1 [7] [3].

A. Insight into Large Language Models

The greatest pioneering advancement of AI comes up as LLMs capable of understanding and generating text with very high accuracy and human-like quality. These models have greatly improved the jobs of Natural Language Processing, such as GPT, BERT, Llama, all developed by transformer-based architecture and larger datasets [10]. Such recent advances as Llama 3.1 guarantee major improvements in efficiency and context comprehension, therefore, opening new frontiers in developing applications requiring sophisticated dialogues such as customer service [12].

B. Fundamentals of Retrieval-Augmented Generation

RAG integrates the retrieval mechanisms and generative AI to overcome the limitations of standalone LLMs with respect to the use of external knowledge sources [17] [11]. Therefore, precise and domain-specific updated information can be fed into the system, which increases not only its relevance but also its accuracy. By harnessing both retrieval and generation, RAG gives large models the ability to ground themselves in reliable data and reflect on it during the generation stage [7].

C. How Agentic RAG builds on top of RAG

An Agentic RAG is simply an expansion of the term full RAG; it extends traditional RAG in that special agents can autonomously perform specific functions under the auspices of the RAG [8]. The agents consume data from the graph-based architecture to manage a variety of complex workflows and decision processes. Thus, for example, Retailia contains an agent who collects product facts from a database, whereas an additional agent collects contextual FAQ answers. This is an architecture that supports both scalability and modularity and thus is a great step into the future of AI [14].

D. Unveiling Chatbot Systems

Chatbots have evolved as much from just traditional rule-based systems to AI-controlled agents for conversation management [2]. Old systems were always constrained in that they could not assimilate new queries or contexts. With the capability of machine learning and natural language understanding (NLU), today's AI-based chatbots offer interactive and lively experiences to users [4].

E. AI Chatbots Transforming Customer Service

Artificial Intelligence chatting bots revolutionized the customer services in a retail company; they solve problems like long waiting periods and poor quality support [1]. Chatbots automate repetitive work processes and, consequently, create avenues for personalization from insights offered by the data they analyze. However, we are met with a kind of problem that most of it relies on developing a balance between mankind interaction quality and automation, which is an aspiration objective of Retailia [6].

F. LLM Advancements in Customer Service to date

The retail industry has seen very rapid technology adoption for employment in artificial intelligence focusing on improving customers' service. Real-time query resolution and personalized recommendations, together with proactive engagement strategies, are other functionalities enhanced by LLMs [9]. A new innovation, Agentic RAG frameworks, would further streamline the incorporation of retail organizations into already existing databases and knowledge repositories at the business level, thus allowing customer engagement users to interact seamlessly as if they were in a single user interface [3].

G. Emerging Issues in Retail Customer Service

The retail industry has been advancing a lot in the face of continuous consumer demand for speedy, accurate, and complete personalized service. The traditional models of customer services are failing to meet this requirement. Yet there are challenges, such as scalability; as larger numbers of customers are added, the response time stretches and the costs become huge because a large portion of them is dependent upon human agents. Accuracy in response is another important challenge; inconsistent answers across the support channels further erode trust and satisfaction. Legacy systems are not able to provide clear answers for very complicated product details or policies. Lack of personalization implies that customers get generic assistance that does not reflect their preferences or purchase history. High turnover rates due to operational inefficiencies, including repetitive task management, exacerbate this. Fragmented knowledge management across platforms delays resolution and degrades user experience.

H. Customer Centric AI-driven innovations

AI today has become a ready-made solution for solving customer service issues. It promises to make services scalable, precise, and personalized through technologies such as Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) [12]. Retailia is also an AI-powered chatbot, on which the urgent issues of the retail sector are relevant to cutting-edge technologies. LLMs such as Llama 3.1 process large datasets for contextually correct human-like response generation and understanding, thereby improving customer experience. Using this technology allows RAG to enhance generative features by adding retrieval features. It allows a chatbot to supply references for outside knowledge bases to ground responses in the accurate and world-class information, such as product and policy documents [11]. With Agentic RAG integration, Retailia also allows the use of specialized agents for specific tasks, such as retrieving FAQ answers or querying databases, resulting in more efficient and adaptable operations [16].

I. Analyzing customer service challenges in retail

Running retail has become a real headache and at the same time a balancing act of efficient service and an exploding complexity of what today's customers expect. Old systems for customer service really worked well with simple problems, but they were never designed for a fast-moving, data oriented environment. Such disjointed customer experiences lead to dwindling brand loyalty, which is further sharpened by scattered support channels. The support teams have to contend with delays and errors due to an endless call inundation from consumers. The outcome is helplessness and frustration on the part of the customer service advisory team, which is under-privileged and, therefore, denied access to complete and centralized knowledge and resources to provide speedy and correct resolutions. In fact, better service quality and speedier turnaround, coupled with wider and better customer experience, are needed in these terms.

J. AI's role in addressing the challenges

Natural language models (LLMs) with retrieval-augmentation (RAG) build very powerful solutions to the retail industry challenges faced by their stakeholders. AI-based chatbots never close their doors and thereby reduces waiting time while ensuring continuous customer services [1]. Moreover, it improves overall efficiency as human agents can now devote their attention and time to tackling more complicated interactions-not just dealing with fixed repetitive queries. RAG technology enriches retrieval of accurate responses by seeking from external resources addressing reliability and timeliness of answers. With context-aware, it gives more natural conversation quality expected by customers in conversations. Real-time and predictive data analysis enables personalization to tailor outbound responses and proactively reach out to customers based on browsing history and preferences. In addition, machine learning solutions are easily extensible across resource deficient organizations, providing them the flexibility to shoulder increased volumes of queries with no performance drop. Modular architecture like Agentic RAG allows adding and changing functionality on the fly.

K. Addressing the Gap with Retailia

There have been improvements in AI and other current solutions fail to fulfill the accuracy, personalization, and scalability of modern retail requirements. Retailia connects LLMs and RAG for a system through which complex and diverse customer interactions can be managed. Its unique deployment of Agentic RAG assures efficient performance of task-specific capabilities by specific agents in modular and flexible configurations. Therefore, queries posed by customers can be answered by Retailia, improving customer satisfaction and engagement with the business. Furthermore, they personalize interactions with real-time data analytics to make closer connections with customers. Reduces the cost of operations through automating repetitive tasks while being sufficiently scalable to meet the demands at minimal costs imparting flexibility in future growth. Retailia thus sets the yardstick very high with regard to the adaptable end-to-end AI solution for customer care that retail needs today.

II. Literature Review

A. Synthesis of Current Research Findings

There have been several improvements made to AI customer service during the course of developing LLMs and highly recommended frameworks, RAG, among others. LLMs, including programs such as GPT-3 and BERT, are known for their phenomenal understanding and generation of human-like text [10]. These models are meant to provide context-aware responses that are able to facilitate the interaction of every industry without barriers. More so, these LLMs are not utterly capable of processing and integrating real-time and complicated data; this is where RAG comes in-world. RAG is an improvement in the overall performance of the LLMs-that brings such systems into the retrieval of external data so that the outputs become more accurate and relevant [17] [7]. Retailia then relies on this by merging Llama 3.1 with an Agentic RAG approach to do short solutions for real-time, accurate, and scalability customer service.

B. Notable LLMs in Retail

Availability of new promises of LLM technology includes Llama 3.1 among other developments sooner, showing improvement over designs such as GPT-3 and BERT [10]. From Llama 3.1 alone, it has brought with itself benefits which make it incomparable to its predecessor models in comprehending context and processing natural languages in highly demanding environments like retail because such an LLM can consume a massive amount of the input data and provide contextually resonant output. Such LLMs further train themselves according to a more complex type of conversation, making them an essential utility in modern customer interaction. It has been proven to help facilitate better engagement, reduced time of waiting, and resulted in higher satisfaction through its speedier and customized forms of direct interactions in the retail environment.

C. Comparing diverse chatbot frameworks

These are chatbots that evolved from mere little rule-based bots to very well-defined AI technologies using natural language processing and machine learning to provide better user experiences [2]. Although rule-based systems work predictably and are easier to implement, they cannot manage many complex interactions, which require context and adaptability; that's where AI-driven chatbots come in [1].

Most importantly, the development of a high level of communication is enabled with the aid of an LLM, making conversations appear more spontaneous and interactive [4]. Retailia, for example, would integrate retrieval-augmented generation with LLMs thereby joining the two strongholds of data retrieval with generative response, making accurate business responses to customer queries possible and efficient [17] [7] [11].

Fixed those areas which lack the minor mood form and yet present major aspects of the agent-based applications. In an AI framework, scalability and modularity cannot be isolated or prescriptive. Finally, the Agentic RAG Model utilizes such agents to fetch information from a knowledge base or query a database.

Retailia uses Llama 3.1 in combination with the Agentic RAG application as a new way to manage solving complex customer queries quickly and efficiently that builds on previous studies that focused on making chatbots better adaptable and accurate through AI-enabled retrieval and generation [3].

Indeed, these developments speak volumes about the importance of integrating cutting-edge LLM technologies with RAG, for the changes in customer service are that deep. The similar foundations are laid by Retailia to combine Llama 3.1 with Agentic RAG into a scalable, flexible, and highly efficient model for reaching customers.

III. Objective and Scope

The main aim of this project is to evaluate the effectiveness and viability of combining Llama 3.1 with Agentic RAG in improving retail customer service. To this end, the following objective is defined:

A. Assessing Llama 3.1 and Agentic RAG in Retail

Evaluating how much Llama 3.1 in conjunction with Agentic RAG deals with the complicated situations of customer client interaction components. The measures of progress include the degree of gains in response accuracy, fluency, and contextuality, among others ([10] [17].

B. Enhancing Customer Responses

Investigate the contributions of Llama 3.1 and Agentic RAG for efficient customer service. The performance of these technologies can be evaluated using metrics, including reduced wait times and other general satisfaction aspects, in line with new age consumer expectations about fast and effective service [7] [2].

C. Real-World Performance of Retailia

Investigate Retailia's adaptation into employing existent retail infrastructures and practices. Real-time query handling ability, flexibility in scenarios, and stable service quality under different conditions are a few factors to judge their adaptability [8].

D. Scalability Assessment of Retailia

To study the scalability and agility in handling increasing amounts of customers by Retailia. The object is vital to check whether Retailia can be scalable in larger retail spaces and adapted in future changes without loss of efficiency [3].

These objectives are SMART: Specific, Measurable, Attainable, Relevant, and Time-bound. This guarantees that the research can give actionable insights into the use of Llama 3.1 along with Agentic RAG in modern retail customer services.

This paper primarily evaluates the feasibility and effectiveness of integrating Llama 3.1 and Agentic RAG in the Retailia project as a customer service improvement for the retail industry. The evaluation scope focuses on enhancing the query handling and response accuracy and ensuring customer satisfaction delivered by these AI technologies. It investigates how adaptable Retailia would be in real-world retail environments and how it scales to high customer interactions. The analysis seeks the efficiency and cost dimensions of the system while ensuring service quality. This research also aims to compare the implementation of Retailia to other existing models to differentiate and show the way for broader application throughout the retail industry. Finally, the paper discusses the theoretical rationale and practical implications of combining Llama 3.1 and Agentic RAG to create the new benchmark for AI-driven customer interactions in retail.

IV. Data Collection

The SQL Agent and the FAQ Agent are two types of specialized agents which are used in Retailia. They have been developed to take care of certain classes of queries based on the data sources of each agent. Organizing the sources proves that they are not only effective in managing the structured and unstructured data but also make sure that it will answer accurately and relevant answers to the questions asked by users. [17] [7].

A. Working of the SQL Agent

Since we used the Faker library to generate synthetic data for our project, the data was already preprocessed and formatted in such a way that all fields are reasonably good. Faker actually consists of real names, addresses, emails, and other fields, so not every new one needs data cleansing. We made sure that there was no missing data in the generated data, because Faker made sure that every field was filled with the correct values. For any additional necessary verification, we manually verified data consistency (e.g., we checked that the generated data had no missing values or contradictions). This database also simulates real situations and has the following major tables to form the framework for e-commerce management:

a) Users Table: In this table, there is user related content like usernames, passwords, e-mail id's, and other secured information or data related to users.

b) Products Table: In this table, there is product related content like product names, description for products, prices, quantities, which is efficient for inventory management.

c) Orders Table: In this table, there is order related content which tracks the order placed by users, linking their order to relevant users by a foreign key in the user table.

d) Cart items Table: In this table, there is cart items related content which itemizes products with each order and connecting orders and product tables, also enables detailed tracking of order.

e) Cart Table: In this table, there is cart related content which shows the items in cart for each individual account of the user and supports updates during purchases.

B. Analyzing the diversity of Queries handled

a) User-Specific Queries: These include queries of users, like fetching order details, cart details, user feedback, mainly focused on personalized data retrieval for user experience enhancements.

b) General Queries: These include queries of users which are generalized like, product details, insights for business operations, calculating total sales.

Using LangChain and SQL Database Toolkit, SQL agents can interact with relational databases, making sure that executing complex queries and integrating with Llama 3.1 for NLP responses.

C. Optimizing Customer Service with the FAQ Agent

The FAQ Agent has a specialization in processing unstructured data which comes from user-uploaded PDF documents. Mostly, these papers serve as a basis for similarity-based retrieval and are relevant information to user-frequent queries and customer assistance.

a) PDF Documents: Firstly, users upload some files, which many programs such as PyPDFReader and PyPDF2 process. Then, the text understood from these documents is further processed and extracted for analysis.

b) Content: Documents, literature on frequently asked questions and their answers, as well as product and troubleshooting manuals. There are some policy guidelines on rules around returns and exchanges.

c) Purpose: Using Google Gen AI, the extracted content is transformed into vector embeddings that are stored in an FAISS vector database. This allows the FAQ Agent to match user's queries with appropriate contents through embedding, enabling efficient similarity search [10] [8].
Thus, the FAQ Agent serves the same end as the larger RAG framework by allowing document-source information retrieval and ensuring precise, relevant replies. This is useful, especially when asked to respond to intricate questions that keep changing [17].

D. Benefits of Data Collection

a) Types of Data Responses: It is not only order-specific information but also the general FAQs which vary widely through the combined efforts of the SQL Agent and FAQ Agent.

b) Data Integration Scaling: Modular design allows easy addition of data sources while ensuring flexibility according to changing business needs.

c) Precise Query Responses: By getting accurate and context-aware responses with tailored sources and embedding schemes, the system actually increases users satisfaction by increasing operational efficiency [7].

V. Exploratory Data Analysis (EDA)

A. Data Cleaning and Preprocessing

The dataset was pre-cleaned as well as formatted to ensure that every field conformed logically, since they used the Faker package to create synthetic data for the project. Further cleaning was not needed since Faker automatically generated realistic names, addresses, emails, and other information.

- Training a large model requires a huge amount of data since it works with LLMs.
- Since Faker ensured that all fields were populated with valid values, this confirmed that there was no missing data in the dataset.

We checked data consistency for any further validation that was needed (e.g., verifying there were no missing values or anomalies in the generated data).

B. Data Integrity through Validation

For accuracy and consistency, we verified the generated data at our end. For example, this included checking for types of data that are consistent, such as numeric fields not containing strings, eliminating duplicate entries, and ensuring that all fields are properly filled. For example, we ensure that numerical fields, such as those for product pricing, quantities, and order totals, had realistic numbers and that email addresses were confirmed to appropriate formats.

We also ensured the logical integrity of the data. Sample checks include that the order date falls within the expected range and that a specific user id is unique. Before using the data for project work, the accuracy and integrity of the data were assured through this procedure.

This system chose the type of agents assigned to respond to particular user's inquiries depending on the structure of the query- either as structured or unstructured queries-in a way that will ensure appropriate answers mitigated to a certain extent by precision.

VI. Hypothesis

A. Mechanism of Dual- Agent Architecture

By using a multi-agent chat-bot system, incorporating SQL and FAQ agents, the best overall improvements in the efficiency and accuracy of response to user inquiries in e-commerce can be achieved. The SQL Agent is quite specifically optimized for structured data retrieval and would therefore do things like retrieve user information from relational databases-order history details about current carts-with ease alongside general queries such as inventory checks or calculations of sales.

Meanwhile, the FAQ Agent processes unstructured inquiries by using advanced vector-oriented embedding models (e.g. Google Gen AI embeddings) for accurate similarity matching. Pre-process and index user manuals and FAQs in a FAISS vector database; thus the FAQ Agent is capable of providing accurate answers in a very relevant context to frequently asked queries. This dual-agent approach caters for both structured and unstructured queries and removes delays while processing queries, reduces error rates, and overall improves the user experience [7] [17] [5].

B. Dynamics of Query handling through Agentic RAG

Such integration makes possible flexible dynamic query routing and processing with a single architecture through integrating structured retrieval data systems into unstructured retrieval data systems. The intelligent processing will have an efficient evaluation of user questions and right routing to correct subsystems of the Agentic RAG Handler that go either to the FAQ Agent for document-based queries or SQL Agent for database queries by making it the maximum process efficiency. Creating routing accuracy but generating conditions-aware responses by having the responses relied on credible data sources, this is one of the RAG approaches.

The modularity of the design allows scaling and the adaptability possible to manage system efficiency in processing

very large query volumes without compromising accuracy and response time. The outcome is significant improvement in chatbot performance, user satisfaction, and overall quality of interaction, making it quite an effective solution for dynamic and complex e-commerce environments [8].

VII. Methodologies

A. System Architecture

By making use of different sophisticated AI technologies such as the Agentic RAG framework and the Llama 3.1 model, the organizational architecture of Retailia seamlessly lends itself to intelligent, responsive and scalable customer support systems. Such a design that is meant to be able to address both structured and unstructured data guarantees the generation of context-aware answers to a range of queries and satisfies intricate requirements in e-commerce applications. Flexibility and agility to manage the constantly changing nature of customer contacts would be provided by the modular design that integrates many specialised agents [17].

The basic building block of the Agentic RAG Handler is as a decision-making unit that will pass the query to the specific agent (either give to the SQL Agent or give it to the FAQ Agent based on the question type) depending on the kind of query received from the users. By employing tools like PyPDF2, LangChain, and FAISS, the entire procedure follows a systematic line from data extraction to answer generation. With the aid of Llama 3.1, real-time, applicable, and precise conversations have been made possible. The input regarding the natural processing and generation of human language was also enhanced.

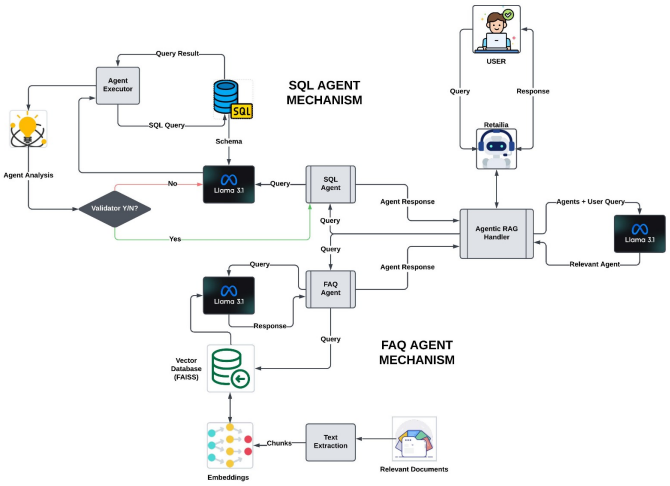

Fig. 1. Architecture diagram of Retailia Project.

B. Strategy for RAG implementation

This RAG method exploiting generative AI features and data retrieval techniques aims to overcome the limitations of conventional LLMs [11]. Besides generating language supposed responses, this hybrid system guarantees that the system bases its responses on precise, real-world data collected externally when asking for accurate, current information on complex and dynamic statements. The RAG approach works very well in such cases [7]. RAG bridges static knowledge bases with interactive user queries by facilitating context-sensitive creation.

C. Agentic RAG Implementation

It enhances the RAG methodology by the Agentic RAG framework allowing introducing task-specific agents functioning independently but collaborating with each other in the same system. The modular design ensures that this will grow the scalability and versatility of the architecture to make it appropriate for handling general forms of queries [8]. From processing unstructured material such as product manuals and FAQs to extracting structured database information, Retailia's agent-centric structure allows it to serve a wide variety of client needs.

D. The Operation of Agentic RAG Handler

SQL agents would handle specific types of structured data queries-input by a user-such as: inventory status or order details. For other queries, they could involve the application of an FAQ agent. In principle, therefore, this intelligent routing should not only ensure that the response is accurate and contextually pertinent, but also eliminate latency. The entire system of the Agency RAG handler would in general render decision-making algorithms used in the system more responsive and effective [17].

E. The Role of Two Specialized Agents Employed

1) FAQ Agent: The FAQ Agent handles the unstructured data, which includes product FAQs and support documents. Its workflow involves:

a) PDF Text Extraction - PyPDF2: Broader scope for extracting relevant texts from different types of documents like product manuals and policy documents such that all such information is available for use.

b) Text Splitting - Recursive Character Text Splitter: Segregation of bulky documents into chunks, which can then be embedded and retrieved for optimizing performance.

c) Embedding and Vector Storage - Google AI Embeddings and FAISS Index: Visualize and convert textual data into high-quality vector embeddings when facilitating similarity search retrieval [15].

d) Get Conversational Chain Function: Portal for humanizing the query and maintaining all other connective processes of the flow.

e) Get FAQ Response: Provides valuable answers from combined embeddings retrieved with the generative features of Llama 3.1 [11].

2) SQL Agent: The SQL Agent is the component specialized in the structured query language needed for queries and interaction with the database. The major steps should be:

a) Defining QUERY Template: Predefined SQL templates as a basis are used to form accurate and consistent queries on the database.

b) Initializing LLM: Llama 3.1 would then be aware of how queries to the database are interpreted and responded to using extremely fluid and pertinent language styles [10].

c) Database Set Up: Database configuration is quick and simple, ensuring that connection to the model toolchain is seamless.

d) Installing the SQL Database Toolkit: This integrates the LangChain Community Toolkit for efficient extraction and running of advanced SQL queries [3].

F. Utilizing LangChain and LangGraph for Enhanced Data Processing

When it comes to Retail architecture, it is inevitable to have LangChain as the main feature that allows you to process vast amounts of text data. One such tool provided in LangChain for processing huge documents is the Recursive Character Text Splitter, which tears it into manageable chunks. This segmentation enhances system responsiveness as well as reduces the latency time of processing, ensuring peak performance during embedding and retrieval. Efficient in-use Vectorization and Retention of text from the FAISS database by recording text structural decomposition are some of the features that LangChain supports.

Just like LangChain is, well, namely LangGraph, it visualizes the corresponding data flow and agent interactions throughout the system. It enables one to debug or otherwise enhance chatbot performance while monitoring the routing and processing requests.It scales by mapping different components, so introducing new agents or capabilities does not interfere with the existing design [8].

G. Extracting Information from Documents with PDF Reader

PyPDFReader is superior to PyPDF2 because it delivers a parsing architecture that is specifically designed for difficult PDFs, even those that have annotations, complex overlay text, or images embedded into them. This is an all-in-one; accurate processing is guaranteed even for the trickiest structures with thorough data extraction capabilities.

The FAQ Agent is more dynamic using PyPDFReader since it possesses an extensive range of support regarding the document formats, allowing it to deal with different types of sources like legal documents, technical manuals, and multilingual matters.

The built-In error-handling properties of PyPDFReader also ensure that the only little loss in data is encountered while processing even partially formatted and damaged files. For very large e-commerce majors, this robustness becomes very critical to ensure that the integrity of the database remains intact.

H. Google Gen AI Embeddings for Enhanced Query Understanding

The Google Gen AI embeddings were used for the creation of vector representations of unstructured textual information for efficient searching and retrieval mechanisms [5] [15]. Compared to the traditional embeddings, these embeddings have a better performance and hence became more useful in matching the user's searches with the relevant properties since they can contain extremely rich semantic correlations between words and sentences. While these embeddings performed faster compute times than the other contextual embeddings such as Hugging Face, Sentence Transformers, this development takes the system to a scale where queries in thousands can be processed in no time at all [15]. These embeddings are also optimized for multilingual data, in that sense, expanding Retailia's reach into the world [10].

I. Emphasis on Integration of Agentic RAG and LLMs

Agentic RAG collaborating with Llama 3.1 thus advances AI-based customer care. As Retailia claims, accuracy and personalized interaction have both heightened with the original creative powers of LLM combined with the retrieval powers of RAG. Even if Llama 3.1 gives responses that are contextually relevant and linguistically fluent, the Agentic RAG design enables query to be modularly handled e.g. several agents specializing in different tasks [7].

Furthermore, the link facilitates additional scope flexibility or training. With its newer knowledge base, new information can be added through availability of the FAQ Agent and SQL Agent, thereby ensuring the correctness and timeliness of its responses.

J. Llama 3.1 Model Overview and Predecessor Models

It is said that the firm ends its reports by putting up to develop a position of strength in the market through the usage of Llama 3.1, which will be a successor to other earlier models such as GPT-3 and BERT. It is enhancing efficiency as well as the contextual understanding of most queries raised in customer interactions. Following are the major improvements:

a) Fine-tuned Contextual Awareness: Llama 3.1 is good at reading the subtle query and is therefore very good for customer interactions.

b) Improved efficiency: The model is optimized for faster response rates even during peak query traffic so that real-time interaction is feasible.

c) Scalability: The architecture of Llama 3.1 supports deployment at a large scale, for instance, in an e-commerce site. Thus, the advances place Llama 3.1 at the heart of Retailia's architecture for providing very high-quality customer service in various and complex scenarios [10].

d) Steps of Data Preparation and Pre-processing: The performance of SQL Agent and FAQ Agent lies halfway in efficient preprocessing and data preparation. Thus, the following actions should be taken to ensure integrity of data and recovery accuracy:

- PyPDF2's PDF Text Extraction: This helps to extract text from various formats of document, sets up retrieval and embedding.
- Texts Splitting: The Recursive Nature of LangChain Text Splitter maximizes processing time by ensuring long documents into digestible pieces but stays within context.

- Vector Storing and Embedding: Google Gen AIs convert textual content into high-dimensional vectors, then stored into FAISS for retrieval on-the-fly [5].
  Because of this process, both highly structured and unstructured data will be ready for smooth integration into the systems, and the resultant output shall be excellent answers to various queries.

K. Information retrieval with `get_faq_response` function

In a nutshell, the Get FAQ Response module applies both creative faculties of Llama 3.1 and retrieval ability through FAISS, ensuring the best selection of appropriate and relevant responses. First, the user query goes into embedded actions to vector representation, whereupon searching content stored in FAISS to find the most appropriate sources for it. Next, Llama 3.1 consumes this material with its language and contextual understanding for output of a tailored response. There is inevitably total accuracy with respect to personalisation to the individual user's requirements.

The key features of the Get FAQ Response module include:

a) Dynamic Contextual: The ultimate answer to the queries of conversations, fetching them as relevant whenever required.

b) Real-Time Retrieval: It does real-time similarity searches by FAISS, thus helping in reducing response latency. Pulls in information from numerous documents and sources such as product manuals, FAQs, support documents, to provide complete responses.

This is actually the engine upon which the Retailia architecture rides because it addresses unstructured query handling, associated with reduced customer frustration and increased engagement.

L. Possible ways to deploy the Model

To meet varying operational needs and scalability requirements, Retailia has multiple deployment options. Each one of them has a different set of benefits which makes it apt for different use cases:

1) Local Deployment: The ideal dimensions of the development environment are laboratory and field trials, where the developer builds a model that can be used for performance evaluation without deploying any real customer infrastructure. A small-scale application or pilot testing will be useful before any large-scale deployment.

2) API Deployment: What really accesses Retailia functionalities in real-time is it allows integration with e-commerce platforms or third-party applications so smoothly. It provides excellent accessibility and ease of use because these are usually for businesses demanding faster implementation.

3) Container Deployment: Packages system into portable containers using Docker that guarantees consistency across environments. It simplifies scaling and maintenance, hence a choice among all-cloud or hybrid deployment. Supports microservices architecture with modularity for expansion and integration.

4) Cloud Deployment: An architecture intended for large-scale operations with high availability and performance. It can auto-scale to cater to varying loads, ensuring it can cope with peak demands, and integrates with different cloud platforms such as AWS, Google Cloud, or Azure, so an enterprise can rely on their infrastructures for reliability and security [5].

All deployment methods are tailored to specific business needs so that Retailia can be as flexible and scalable as its operating environments require.

M. Refining AI Responses through Prompt Engineering

Prompt engineering is an integral part of Retailia's design, intended to fine-tune the interaction between the users and Llama 3.1. The system would be able to provide precise and coherent responses, considering user expectations through the formulation of good and effective prompts [13].

1) Techniques Used:

a) Zero-Shot Prompting: Therefore, without a prior example being given, the Llama 3.1 can generate any number of responses, thus making it the most suitable to handle any fresh or varied query.

b) Few-Shot Prompting: Ciphers examples to be given to the model for better accuracy in contextually focused scenarios.

c) Iterative Refinement: Prompts tested and adjusted iteratively across different types of queries to achieve final optimal performance [13].

2) Value Delivered:

a) Improved Response Quality: Assuredly guarantees the system produces accurate and context-aware responses.

b) Enhanced Flexibility: Utilizes various discussions and user needs.

c) Reduced Training Overhead: Avoids the need for extensive retraining because good prompts effectively steer the model behavior. Avoids the need for extensive retraining because good prompts effectively steer the model behavior [2].

N. Growth potential for the system

While the current architecture of Retail displays considerable progress in AI-customer service, several advancements can still be made including:

a) Multilingual Support: An expanded Google Gen AI embeddings towards a wider range of languages with appropriate translation layers integrated within them, so that they can enable equal opportunities for global markets [6].

b) Real-Time Feedback Loop: Feedback mechanisms to optimize had their responses inferred by the level of satisfaction of patrons. Making use of reinforcement learning to adjust the policy to changing customer demands.

c) Advanced Personalization: Customer profile institution for personalized recommendation and response. Past customer data to predict likely future needs.

d) Enhanced Knowledge Base Management: FAISS to be updated automatically with the latest information and with dynamic content filtering for a full load of most relevant data.

e) Hybrid Deployment Models: A combination of the above both cloud and edge-deployed systems balancing the need for scalable services with low-latency interactions, and ensuring the offline availability of features for areas with limited connectivity.
That captures these states where limitations can be added to realize the complete advancement of Retailia as a self-contained possible solution for customer service via AI in a rapidly evolving age and industry.

VIII. Implementation

The integration of Retailia by API aims at constructing a highly scalable and efficient chatbot solution using the state-of-the-art AI capabilities and well-organized workflows for optimal performance.

A. Setting API Basis

1) Replicate API Integration: We have used Llama 3.1 70B Model as our core language model, for achieving integration through GROQ API. Where an API key was generated for securing access.

2) Model Initialization and Configuration: This model has been configured with the key parameters for fine tuning the behavior:

- Max Length: Maxlength parameter is to control the maximum context size and
- Temperature: Temperature parameter to control the randomness of the generated response, precision balancing.

These are the parameters that make sure the chat bot delivers the context aware, and user specific responses.

B. Preprocessing and Embeddings

1) For FAQ Agent:

a) Document Loading: We have loaded the document using PyPDF2 library, to extract the text from PDF documents.

b) Text Chunks: The document is converted into text chunks using RecursiveCharacterTextSplitter. Where the extracted text was divided and made into chunks.

c) Vectorization: The divided chunks are embedded into high dimensional vectors using GoogleGenerativeAIEmbeddings [6].

d) Storage: These Embeddings were stored using the FAISS vector database library. Which enables similarity based retrieval [15].

2) For FAQ Agent:

a) Dynamic Database Generation: We have generated a dynamic database using the faker module to import data to SQL database.

b) Database Accessing: Using sqlite3, we accessed the generated database. To make sure seamless interaction with SQL agents.

C. Querying and Results

1) For FAQ Agent: After uploading the dataset, a conversational chain is built to generate the chatbot response using the `get_conversational_chain()` function. It defines a prompt template that instructs the model on how to answer the user questions based on the context provided [13].

Then the `get_faq_response()` function takes user questions and finds the relevant answer based on embedded knowledge from the FAISS vector store. and generated responses using Llama 3.1.

2) For SQL Agent:

- First we set up a QUERY template, which is a string template that defines how user queries should be handled.
- It instructs the LLM to generate an SQL query based on the user's question. If the query is user-specific, it should use the `user_id` provided, but any mention of specific user IDs or names in the question should be ignored.
- Then we initialized the LLM by defining it, which represents the LLM model via ChatGroq. By defining db, we created the database engine. Now to build the relation between database (db) and LLM(llm) we defined a toolkit using SQLDatabaseToolkit, which accesses and queries the database.
- Here we used the zero shot approach, where the agent can answer questions without explicit training on similar examples.
- The agent uses Langchain's reactive capabilities to determine the correct response.

D. Routing the Query

The core of our chatbot's efficiency lies in its ability to intelligently route queries. Let's break down how this works:

1) Data Model (RouteQuery): The routing decision is managed using a class called RouteQuery. This model defines two potential paths for a user query—either to the SQL Agent for (structured, database related queries) or the FAQ Agent for (unstructured, general information queries).

2) Language Model (LLM) Routing: To determine the appropriate agent, we use a powerful language model called ChatGroq, specifically LLaMA 3.1. The model takes the user's question as input, evaluates it, and provides a structured response that indicates whether the question is about the database or general information. This is achieved using a specially crafted prompt `route_prompt`.

The prompt instructs the language model to act as a supervisor that decides which "worker" should handle the query: either the database worker (SQL Agent) for specific data or the vector store worker (FAQ Agent) for general information [13].

E. Decision Making Process

The Decision-Making Process Let me take you through the decision making workflow:

1) User Input: When a user inputs a question into the chatbot interface, the query first reaches our routing function called `route_question`.

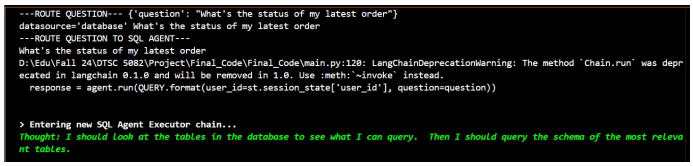

Fig. 2. Routing to SQL Agent.

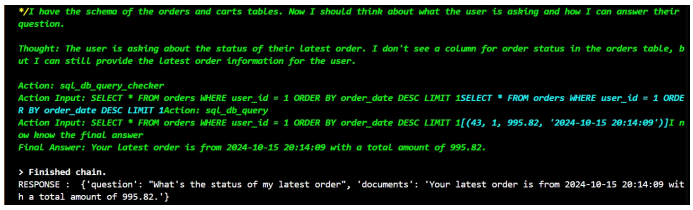

Fig. 3. Routing to SQL Agent.

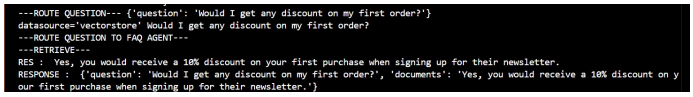

Fig. 4. Routing to FAQ Agent.

2) Prompt-Based Classification: The routing function uses the language model with the `route_prompt` to classify the query. Based on the output, the model chooses either the "database" path or the "vectorstore" path [13].

3) State Graph Workflow: This workflow is defined using a state graph called StateGraph. The state graph directs the flow based on the routing decision:

- If the query is routed to "database", the `db_search` function executes a SQL query to retrieve the required information from the database.
- If the query is routed to "vectorstore", the `retrieve()` function fetches the relevant response from the FAQ knowledge base.

IX. Interfacing with Streamlit

- Streamlit is an open-source library that facilitates the development of interactive web applications. Our retail chatbot designs itself for this because of its widely known simple syntax and multitude of features, which highly quicken the developing and deploying of apps.
- The safe login procedure at the start of the interface makes all user interaction secure and personalized. This is vital for keeping the user history and the context that a user operates within that context improves the user experience in a retail environment.
- With our UI, users can type in natural language questions into a chat input. This makes the chatbot much more accessible and user-friendly for all users for conversational interaction.
- The conversation history can be easily understood through the display of messages in `st.chat_message` within streamlit. This provides a user-friendly experience because users can always look back at previous conversations.
- User queries go through an internal conduit for dynamic dispatch to desired handlers, such as a database for straightforward queries or an encyclopedic frequently asked questions system for general inquiries. A large language model organizes that routing so that responses are always fast and relevant [12]. Therefore, clarity isn't even interrupted when the answers show up directly beneath the user's question.
- The interface includes uploading for PDF documents which is an added accessory for every user looking to process and consult FAQs regarding a product. It improves the interaction for users by allowing a more direct interaction, uploading-and-retrieving relevant information based on inquiries.
- This is placed in priority, as the interface is highly determined by usability and intuitiveness. A much simpler application of users could pose their queries to the chatbot using it without necessary hindrances. The arrangement is neat and structured, making the overall experience very holistic.

Fig. 5. User Authentication interface.

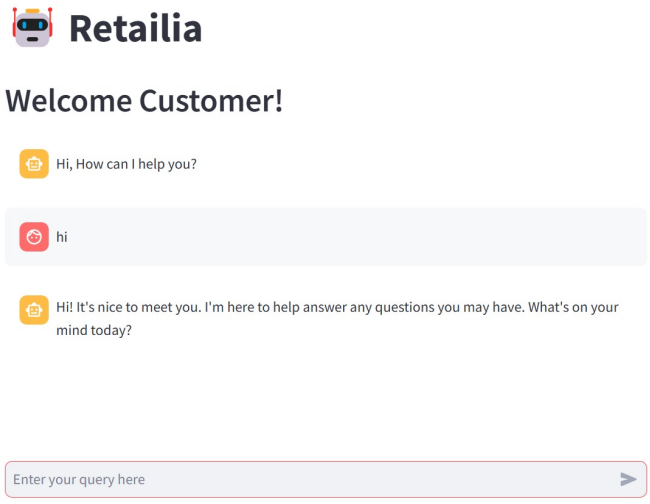

Fig. 6. ChatBot Interface interface.

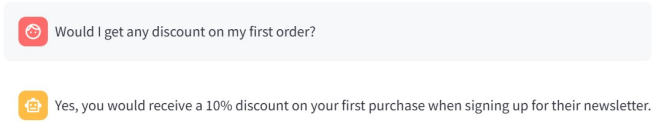

Fig. 7. ChatBot Response for User Query

X. Conclusion

A. Summary

Agentic Retrieval-Augmented Generation architecture and Llama 3.1 are topics explored in this research work by which they hope to implement an advanced AI-powered customer support platform for retail called Retailia. Retailia addresses some of the major challenges contemporary retailing confronts in today's rapidly evolving consumer landscape, such as scalability, personalization, operational efficiencies, and the accuracy of response.

Retailia operates on a dual-agent architecture-The SQL Agent and FAQ Agent. Each agent is designed to handle a particular category of queries; hence, structured data like relational databases and unstructured data like policy manuals or FAQS can easily be generically handled. That system was really established on a combination of Google Gen AI embeddings for accurate retrieval, FAISS for vector storage, PyPDF2 for text extraction, and LangChain for workflow optimization [6]. An intelligent query routing to the most suitable agent increased system performance thanks to Agentic RAG Handler. Improvements in conversational fluency and context-awareness guaranteed the accuracy, and contextuality by boosting the chatbot efficiency speedily with Llama 3.1.

Retailia is such an all-encompassing and flexible system that it meets all requirements of modern e-commerce processes. Its beneficial and customer-oriented solutions can considerably increase customer happiness while lowering operational hassles.

B. Key Technological Advancements in Retailia's Development

1) Dual Agent System: The first agent being SQL. This agent was to process all the structured queries; order history, stock levels, individual variations, etc. The second was FAQ Agent which used advanced embeddings for understanding unstructured queries and retrieving contextually relevant information from its document knowledge bases [15].

2) Smart Query routing: After the dynamic evaluation of the concerned one, the query was routed to the known agent by the Agentic RAG Handler. Modularly developed to provide improved accuracy with lower response times.

3) Integrated Llama 3.1: Llama 3.1 would provide the best natural language understanding and crafting responses to make up the language foundation of the system. It would be vital to task queries that are really complicated but can also be conversationally fluent with a perfect end-user experience.

4) Streamlined Workflow Utilizing Advanced Tools: The efficient data processing was achieved using pine FAISS tools, PyPDF2, and LangChain, an example being RecursiveCharacterTextSplitter of LangChain that chunks long documents into more digestible chunk-size pieces for accomplishing effective embedding and retrieval.

5) Scalability and Adaptability: Retail of Retailia proved that it could withstand heavy loads of request queries without providing any less than normal service level. Hence, it could be used in any quality of industrial applications.

C. Broader Implications of Retailia for the Retail Industry

Design and implementation of retailia symbolize a milestone in the advancement of customer service via AI. The system opens wide-ranging implications in different areas:

1) For Operations in Retail: Retailia is going to revolutionize the way it interacts with customers. It is going to be an omnipresent, complete, accurate, and swift solution. Structured and unstructured queries will be taken care of together whereby he will facilitate total query management boosting client loyalty and satisfaction. Process automation eliminates routine operations lowering operating costs, which enables the mankind agent capability to focus on strategic decision-making as well as complex inquiries.

2) Wider Uses: Retailia's modular design enables it to be used in a variety of industries, including finance, healthcare, and education. Real-time answers to questions must also be contextually relevant, further confirming that Retailia is flexible enough to be deployed in regions outside that of retail.

3) Academic Achievements: This study highlights the innovative potential of combining modular RAG frameworks with state-of-the-art LLMs. The study also highlights how tools such as FAISS and LangChain are important in the development of scalable, efficient, and flexible systems on AI.

D. Future Prospects and Potential of Retailia

1) Multi Lingual Support: Through fine translation layers and embedding multilingual information into the system, Retailia will very soon be able to cater to audiences anywhere in the world.

2) Real Time Feedback: Incorporating real-time feedback loops can improve the system's adaptability by refining responses based on user satisfaction and interactions.

3) Enhanced Personalized: Predictive analysis and microorganism user behavior data can now allow proactive interaction such as personalized recommendations for the client's needs.

4) Hybrid Deployment Models: Because they employ edge computing together with cloud-based scaling, they become relevant for several varied locales, allowing low-latency interactions.

5) Integration of latest Technologies: Enrich customer experience with voice-based interfacing capabilities and the immersive engagement that comes with AR/VR, providing much more interactive and user-friendly assistance

experiences.

Retailia is a prime example of future AI in customer service by filling the gaps in scalability, efficiency, and personalisation while opening the path for bigger innovations. The combination of the Agentic RAG framework along with Llama 3.1 has come together to create a scalable, flexible and resilient system which caters to the evolving requirements of modern e-commerce. It sets a benchmark for designing intelligent AI-enabled platforms and shows how technology can transform user experiences across the industry. Retailia is a path laying among the retail sector and beyond by increasing query processing accuracy, decreasing operational costs, and improving customer satisfaction. Future developments are certain to further enhance its capability and thus make it an indispensable part of intelligent human-AI collaboration in customer service.

REFERENCES

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

[2] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," in *Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations*, 2020, pp. 373–383, Springer, Cham, doi: https://doi.org/10.1007/978-3-030-49186-4_31.

[3] M. Arslan, H. Ghanem, S. Munawar, and C. Cruz, "A Survey on RAG with LLMs," in *Procedia Computer Science*, vol. 246, pp. 3781–3790, 2024, doi: https://doi.org/10.1016/j.procs.2024.09.178.

[4] G. Caldarini, S. Jaf, and K. McGarry, "A literature survey of recent advances in chatbots," in *Information*, vol. 13, no. 1, pp. 41, 2022, doi: https://doi.org/10.3390/info13010041.

[5] H. Das, "AI agents by Google," in *Scribd*, 2023, [Online]. Available: https://www.scribd.com/document/800487626/AI-Agents-by-Google.

[6] A. Følstad and M. Skjuve, "Chatbots for customer service: user experience and motivation," in *Proceedings of the 1st International Conference on Conversational User Interfaces*, August 2019, pp. 1–9, doi: https://doi.org/10.1145/3342775.3342784.

[7] S. Gupta, R. Ranjan, and S. N. Singh, "A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions," in *arXiv preprint arXiv:2410.12837*, 2024, doi: https://doi.org/10.48550/arXiv.2410.12837.

[8] M. C. Lee, Q. Zhu, C. Mavromatis, Z. Han, S. Adeshina, V. N. Ioannidis, and C. Faloutsos, "Agent-G: An Agentic Framework for Graph Retrieval Augmented Generation," in *arXiv preprint arXiv:2409.14924*, 2024, doi: https://openreview.net/pdf?id=g2C947jjjQ.

[9] C. V. Misischia, F. Poecze, and C. Strauss, "Chatbots in customer service: Their relevance and impact on service quality," in *Procedia Computer Science*, vol. 201, pp. 421–428, 2022, doi: https://doi.org/10.1016/j.procs.2022.03.055.

[10] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, and A. Mian, "A comprehensive overview of large language models," in *arXiv preprint arXiv:2307.06435*, 2023, doi: https://doi.org/10.48550/arXiv.2307.06435.

[11] A. Nawalgaria, G. Hernandez Larios, E. Secchi, M. Styer, C. Aniftos, and O. Petragallo, "Operationalizing Generative AI on vertex AI," in *Google Services*, 2023, [Online]. Available: https://services.google.com/fh/files/misc/operationalizing_generative_ai_on_vertex_ai.pdf.

[12] Natanaelfs, "Whitepaper_Foundational Large Language Models & Text Generation," in *Scribd*, 2023, [Online]. Available: https://www.scribd.com/document/793015356/Whitepaper-Foundational-Large-Language-Models-Text-Generation.

[13] "Prompt Engineering," in *Kaggle*, 2023, [Online]. Available: https://www.kaggle.com/whitepaper-prompt-engineering.

[14] C. Ravuru, S. S. Sakhinana, and V. Runkana, "Agentic Retrieval-Augmented Generation for Time Series Analysis," in *arXiv preprint arXiv:2408.14484*, 2024, doi: https://doi.org/10.48550/arXiv.2408.14484.

[15] S. San, "Newwhitepaper_Embeddings & vector stores," in *Scribd*, 2023, [Online]. Available: https://www.scribd.com/document/800205772/Newwhitepaper-Embeddings-vector-stores.

[16] "Solving Domain-Specific problems using LLMs," in *Kaggle*, 2023, [Online]. Available: https://www.kaggle.com/whitepaper-solving-domains-specific-problems-using-llms.

[17] S. Zhao, Y. Yang, Z. Wang, Z. He, L. K. Qiu, and L. Qiu, "Retrieval Augmented Generation (RAG) and Beyond: A Comprehensive Survey on How to Make your LLMs use External Data More Wisely," in *arXiv preprint arXiv:2409.14924*, 2024, doi: https://doi.org/10.48550/arXiv.2409.14924.