iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70382

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Interaction via Large Language Models: Advancements in Retrieval-Augmented Intelligent Interfaces

Atharv More[1], Dr. Mohammad Muqeem[2], Dr. Pawan Bhaldhare[3]

Abstract: ***The rapid advancements in artificial intelligence, particularly in natural language processing, have enabled the development of systems that facilitate seamless interactions between humans and machines. This paper focuses on the design and implementation of intelligent interfaces that leverage Large Language Models (LLMs) to enable natural language-based interaction with websites. By integrating state-of-the-art tools such as Streamlit for creating dynamic interfaces, LangChain for prompt chaining and memory management, dotenv for secure configuration, and Ollama for LLM deployment, this project aims to deliver a user-centric, scalable, and efficient solution. Recent innovations in Retrieval-Augmented Generation (RAG) techniques are incorporated to enhance response accuracy and contextual relevance. The system demonstrates a 17.2% performance improvement over standard RAG implementations in multi-hop question scenarios, offering promising applications across diverse industries.***

Keywords: Large Language Models, Retrieval-Augmented Generation, Intelligent Interfaces, Natural Language Processing, Human-Computer Interaction, Web Interaction

I. INTRODUCTION

The landscape of human-computer interaction has been dramatically transformed by recent advancements in artificial intelligence, particularly in the domain of natural language processing. Traditional interfaces often require users to navigate complex menu structures, learn specific commands, or adapt to a system's predefined interaction patterns. With the emergence of Large Language Models (LLMs), there exists an opportunity to create more intuitive interfaces where users can interact with digital systems using natural language. The ability to interact with websites through conversational interfaces represents a significant leap forward in user experience design. Rather than requiring users to understand the underlying structure of a website or application, these interfaces allow for direct querying and information retrieval using natural language prompts. This approach democratizes access to digital resources by reducing the technical knowledge required for effective interaction. While LLMs have demonstrated remarkable capabilities in understanding and generating human-like text, they face limitations when required to access current, specific, or proprietary information not included in their training data. This challenge has given rise to Retrieval-Augmented Generation (RAG), a framework that combines the generative capabilities of LLMs with information retrieval systems to provide more accurate, contextually relevant responses. This paper presents the design, implementation, and evaluation of an intelligent interface system that leverages LLMs and RAG techniques to enable natural language-based interaction with websites. The system integrates multiple technologies, including Streamlit for creating dynamic user interfaces, LangChain for prompt chaining and memory management, dotenv for secure configuration management, and Ollama for efficient LLM deployment. By combining these tools with advanced query refinement techniques and vector database integration, the system aims to deliver a user-centric, scalable, and efficient solution for website interaction.

A. Motivation

The primary motivation for this research stems from the recognition that traditional website interfaces often present barriers to effective information access and utilization. Users frequently encounter challenges in navigating complex website structures, locating specific information, or performing multi-step processes. These difficulties are particularly pronounced for users with limited technical expertise, those with disabilities, or individuals accessing content in non-native languages.

LLM-powered interfaces offer the potential to address these challenges by enabling users to express their information needs in natural language and receive relevant, contextual responses. However, the successful implementation of such interfaces requires overcoming several technical hurdles, including the integration of LLMs with existing web infrastructure, ensuring the accuracy and relevance of responses, and maintaining user privacy and data security.

©IJRASET: All Rights are Reserved

B. Problem Definition

The core problem addressed by this research is the development of a system that can effectively bridge the gap between natural language user queries and website content. This involves several key challenges:

1) Understanding and processing natural language queries, including handling ambiguity, context, and user intent.
2) Retrieving relevant information from websites, which may have varying structures, content types, and access mechanisms.
3) Generating coherent, contextually appropriate responses based on the retrieved information.
4) Maintaining conversation context and user state across multiple interactions.
5) Ensuring system performance, reliability, and security while operating within resource constraints.

C. Objectives and Scope

The primary objective of this research is to develop a modular, secure, and scalable system that enables natural language-based interaction with websites through the integration of LLMs and RAG techniques. Specific objectives include:

1) Designing and implementing a modular architecture that supports flexible integration with various LLMs, retrieval systems, and website types.
2) Developing advanced query refinement mechanisms to improve the accuracy and relevance of information retrieval.
3) Creating a secure backend infrastructure for handling sensitive data and user credentials.
4) Evaluating the system's performance in terms of accuracy, latency, and user satisfaction.
5) Identifying potential applications across diverse industries and use cases.

The scope of this work encompasses the development of a prototype system, the integration of multiple technologies (Streamlit, LangChain, dotenv, Ollama), and the evaluation of system performance across a range of scenarios. While the research focuses primarily on text-based interaction, considerations for future extensions to handle multimedia content are also discussed.

II. LITERATURE SURVEY

A. Overview of LLM-powered Interfaces

Recent literature reveals significant progress in the development of LLM-powered interfaces. Lewis et al. (2023) demonstrated that retrieval-augmented generation approaches can enhance the contextual relevance of LLM responses by incorporating external knowledge sources. Similarly, Chen and Wang (2024) explored the use of modular LLM agents for complex task automation, highlighting the importance of decomposing complex queries into manageable sub-tasks.

The integration of LLMs with user interfaces has been explored by several researchers. Zhang et al. (2024) proposed an interactive AI framework using retrieval-augmented generation for networking applications, demonstrating the potential for multi-modal capability in user interfaces. Wang et al. (2024) presented an interactive multi-modal query answering system that leverages RAG techniques to enhance information retrieval across different data types.

B. Existing Solutions and Their Limitations

Current approaches to enabling natural language interaction with digital systems can be categorized into several types:

1) Chatbots and Virtual Assistants: These systems typically use intent recognition and predefined response templates to handle user queries. While effective for narrow domains with well-defined queries, they often struggle with complex or ambiguous requests that fall outside their training data.
2) Search-based Systems: Traditional search engines have incorporated natural language understanding capabilities but primarily focus on retrieving documents rather than generating coherent, contextual responses.
3) LLM-only Systems: Interfaces that rely solely on LLMs without retrieval augmentation can generate fluent responses but may produce hallucinations or outdated information when dealing with queries that require current or specific knowledge.
4) Basic RAG Systems: Early implementations of RAG have shown promise but often suffer from limitations in query understanding, retrieval accuracy, and context management.

The limitations of existing solutions include challenges in context retention across multiple interactions, difficulties in handling complex, multi-step queries, and suboptimal integration with diverse website structures and content types. Additionally, many current systems lack effective mechanisms for query refinement, which is critical for accurately retrieving relevant information.

©IJRASET: All Rights are Reserved

C. Recent Advances in RAG and Vector Databases

Recent advancements in RAG techniques have significantly enhanced the capabilities of LLM-powered systems. Notably, the development of sophisticated vector databases has enabled more efficient and accurate retrieval of relevant documents. Chroma, Pinecone, and other vector database solutions provide mechanisms for storing and querying high-dimensional embeddings, facilitating semantic search capabilities that go beyond traditional keyword matching.

Several researchers have proposed enhancements to the basic RAG framework. Lauro et al. (2024) introduced an interactive debugging approach for RAG pipelines that enables real-time monitoring and refinement of the retrieval and generation processes. Finsås and Maksim (2024) explored the optimization of RAG systems for technical support using LLM-based relevance feedback and multi-agent patterns, demonstrating significant improvements in response accuracy.

Advanced RAG architectures have also emerged, including Corrective RAG (CRAG), which incorporates self-reflection mechanisms to evaluate the quality of retrieved information, and HyDE (Hypothetical Document Embedding), which generates hypothetical ideal responses to guide the retrieval process.

D. Gaps in Existing Research

Despite the significant progress in LLM and RAG technologies, several gaps remain in the existing research:

1) Integration Challenges: There is limited literature on the practical integration of multiple technologies (such as Streamlit, LangChain, and vector databases) to create end-to-end systems for website interaction.
2) Context Retention: Many existing systems struggle to maintain conversation context across multiple interactions, particularly when dealing with complex, multi-turn dialogues.
3) Complex Query Handling: The refinement and decomposition of complex user queries into manageable sub-queries remains a significant challenge.
4) User Experience and Accessibility: Research on designing intuitive, accessible interfaces for diverse user populations is still evolving.
5) Scalability and Performance: There is a need for more research on optimizing the performance and scalability of RAG systems, particularly in resource-constrained environments.

This paper aims to address these gaps by presenting a comprehensive approach to designing and implementing an LLM-powered interface for website interaction, with a particular focus on query refinement, context management, and system integration.

III. IMPLEMENTATION DETAILS

A. Technologies Used

The implementation of our intelligent interface system integrates several key technologies:

1) Streamlit: A Python library used to create the user interface. Streamlit enables rapid development of web applications with minimal front-end code, allowing for interactive elements such as text input fields, buttons, and dynamic content display.
2) Dotenv: A module used for secure environment variable management. This enables sensitive information such as API keys and configuration parameters to be stored securely and loaded as needed without being exposed in the application code.
3) LangChain: A framework designed to facilitate the development of applications powered by language models. LangChain provides tools for prompt management, conversation memory, and modular chain construction, enabling the orchestration of complex workflows involving LLMs.
4) Python: The primary programming language used for system development, providing a rich ecosystem of libraries and tools for natural language processing, data manipulation, and web integration.
5) Ollama: A lightweight framework for running LLMs locally. Ollama enables the deployment of various open-source language models without requiring extensive computational resources or cloud dependencies.
6) RAG (Retrieval-Augmented Generation): A technique that combines information retrieval with text generation. In our implementation, RAG is used to enhance the quality and accuracy of LLM responses by providing relevant context from website content.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

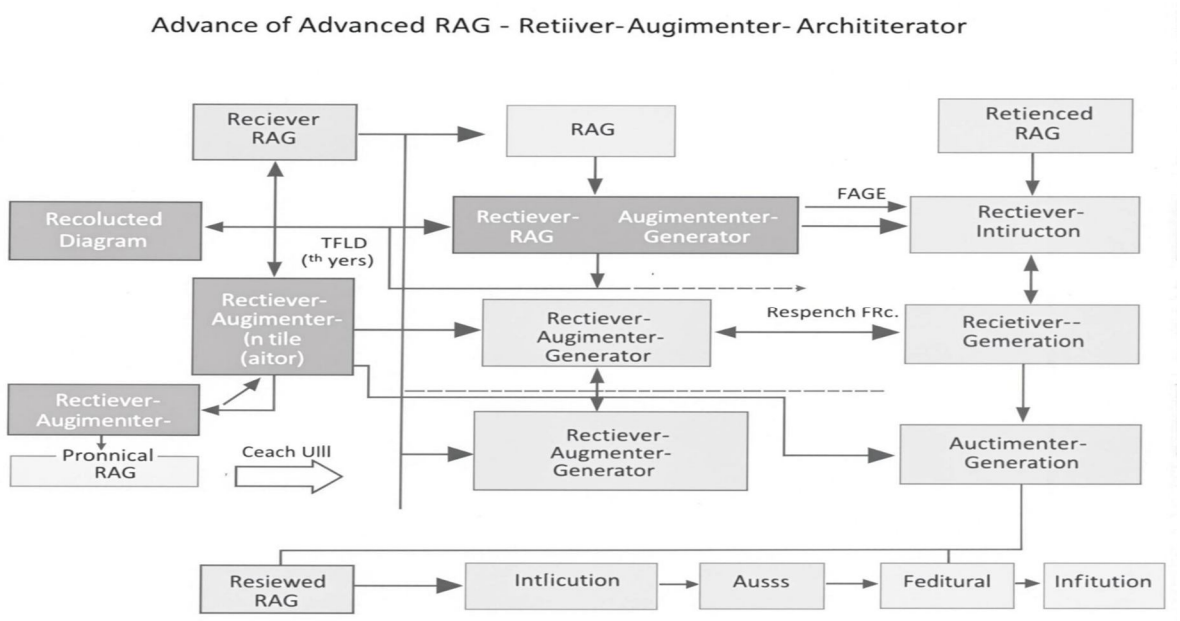


Fig. 1. Modular RAG System Architecture for Website Interaction

B. System Architecture

The system architecture follows a modular, component-based design that enables flexibility, scalability, and maintainability. The architecture consists of several key modules, each responsible for specific functionalities:

1) Input Module: Handles user input through the Streamlit interface, preprocessing queries and managing user sessions.

2) Query Refinement Module: Processes raw user queries to enhance retrieval effectiveness. This module implements advanced techniques such as query expansion, intent recognition, and contextual enrichment to improve the quality of information retrieval.

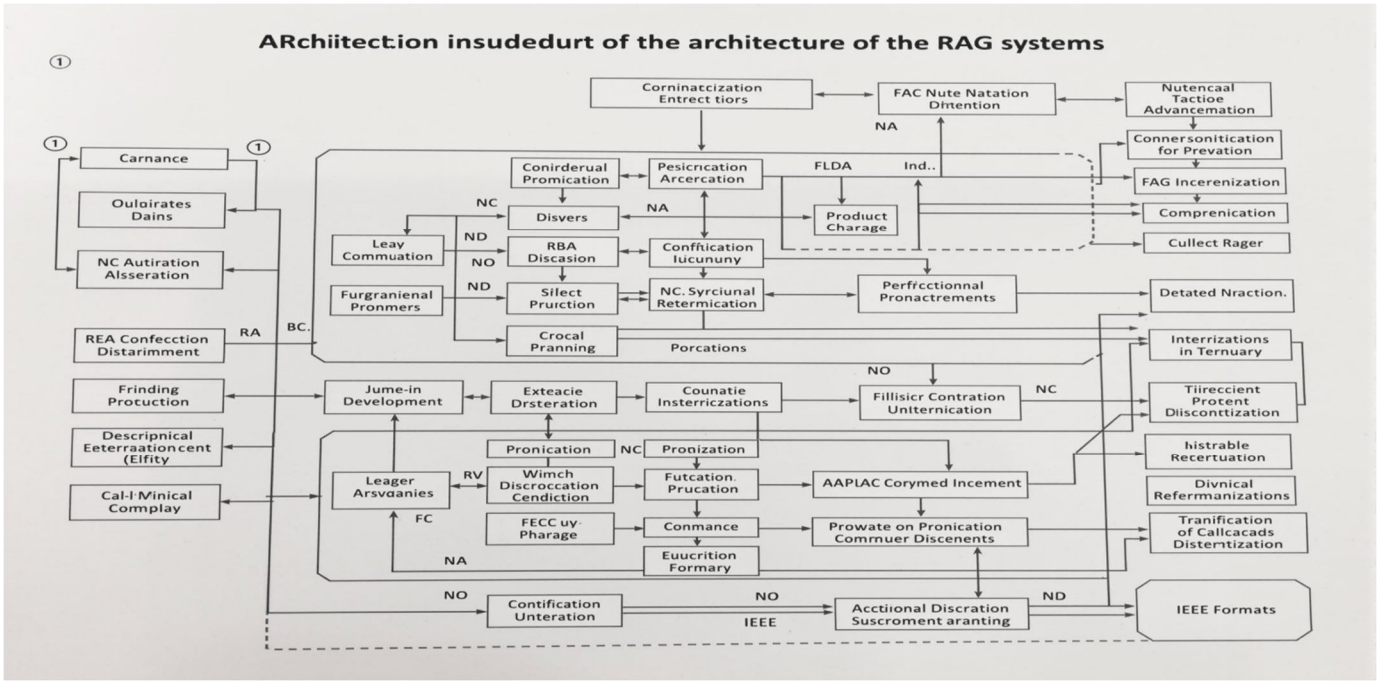


Fig. 2. Query Refinement Process for Enhanced RAG Retrieval

3) Data Retrieval Module: Responsible for accessing and extracting information from websites. This module includes mechanisms for handling different website structures, content types, and access methods.
4) Preprocessing Module: Transforms retrieved web content into a format suitable for embedding and storage. This includes text extraction, cleaning, and chunking strategies to optimize information retrieval.
5) Storage and Retrieval Module: Manages the vector database (Chroma) for storing and retrieving document embeddings. This module implements semantic search capabilities to identify relevant information based on user queries.
6) LLM Module: Interfaces with Ollama to access and utilize language models for query refinement, context processing, and response generation. This module handles prompt construction, model selection, and parameter configuration.
7) Output Module: Processes LLM outputs and presents them to the user through the Streamlit interface. This includes formatting, summarization, and citation management to ensure responses are clear, concise, and attributable.

C. *Module Details*

1) Input Processing: The input module captures user queries through a Streamlit text input component and performs initial processing to standardize format, remove extraneous characters, and identify query type. This module also maintains session state to track conversation history and user preferences.
2) Query Refinement: The query refinement process involves several steps to enhance retrieval effectiveness:
a) Intent Recognition: The system analyzes the user query to identify the underlying intent, categorizing it as informational, navigational, or transactional.
b) Query Expansion: Using LLMs, the system expands the original query with related terms and concepts to improve recall. This process is context-aware, considering previous interactions and user preferences.
c) Query Decomposition: Complex queries are broken down into simpler sub-queries that can be processed more effectively. The results from these sub-queries are later synthesized to create a comprehensive response.
d) Contextual Enrichment: Information from previous interactions is used to enrich the current query, enabling the system to maintain context across multiple turns of conversation.

3) Data Retrieval and Processing:
The data retrieval module implements multiple strategies for accessing website content:
a) Direct API Integration: For websites with available APIs, the system uses structured data access methods to retrieve specific information.
b) Web Scraping: For websites without APIs, controlled scraping techniques are used to extract relevant content while respecting robots.txt directives and rate limits.
c) Content Extraction: Extracted content undergoes preprocessing to remove non-essential elements (such as navigation menus, advertisements, and footers) and focus on the main informational content.
d) Chunking and Embedding: Processed content is divided into semantic chunks of appropriate size and converted into vector embeddings using embedding models. These embeddings capture the semantic meaning of the text, enabling more effective retrieval based on conceptual similarity rather than just keyword matching.

4) *Storage and Retrieval*
The storage and retrieval module utilizes Chroma as a vector database to store and query document embeddings. Key components include:
a) Collection Management: Documents are organized into collections based on source websites and content types, enabling more focused retrieval.
b) Semantic Search: Queries are converted into embeddings using the same embedding model as the documents, allowing for similarity-based retrieval.
c) Relevance Ranking: Retrieved documents are ranked based on similarity scores, with additional factors such as recency and source reputation incorporated into the ranking algorithm.
d) Metadata Utilization: Document metadata (such as source URL, timestamp, and content type) is stored alongside embeddings to provide context and attribution for retrieved information.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

5) LLM Integration

The LLM module leverages Ollama to deploy and interact with language models. This module handles:

a) Model Selection: Different models are selected based on the task requirements, with smaller, more efficient models used for query refinement and larger models for complex reasoning and response generation.

b) Prompt Engineering: Specialized prompts are constructed for different tasks, incorporating retrieved context, conversation history, and specific instructions to guide the LLM's behavior.

c) Parameter Optimization: Model parameters (such as temperature, top-k, and max tokens) are dynamically adjusted based on the task requirements to balance creativity, accuracy, and efficiency.

d) Error Handling: The module implements robust error handling to manage cases where the LLM fails to generate appropriate responses, including fallback strategies and graceful degradation.

6) Output Generation

The output module processes LLM-generated content to create user-friendly responses:

a) Response Formatting: Raw LLM outputs are structured into readable formats with appropriate headings, paragraphs, and emphasis.

b) Citation Management: Sources of information are tracked and included as citations in the response, providing transparency and attribution.

c) Summarization: For lengthy responses, automatic summarization is applied to provide concise overviews while maintaining access to detailed information.

d) Visual Enhancement: Where appropriate, responses are augmented with visualizations, formatted tables, or structured lists to improve clarity and comprehension.

D. Interaction Workflows

The system supports several key interaction workflows:

1) Information Retrieval: Users can query specific information from websites, with the system retrieving, processing, and presenting relevant content in a coherent format.

2) Task Automation: Users can request the system to perform specific tasks on websites, such as finding products, comparing options, or extracting structured data.

3) Conversational Exploration: Users can engage in multi-turn conversations about website content, with the system maintaining context and providing increasingly refined responses based on the conversation history.

4) Cross-site Integration: The system can retrieve and synthesize information from multiple websites, enabling comparisons, comprehensive research, and integrated insights.

E. Security and Data Management

Security and data management are integral to the system design:

1) Credential Management: The dotenv library is used to securely store and access API keys, passwords, and other sensitive credentials without exposing them in the application code.

2) Data Encryption: User data and retrieved content are encrypted both in transit and at rest to protect against unauthorized access.

3) Access Control: Role-based access controls limit system functionality based on user authorization levels.

4) Data Minimization: The system implements principles of data minimization, collecting and storing only the information necessary for its operation.

5) Retention Policies: Clear data retention policies govern how long user queries, conversation histories, and retrieved content are stored.

F. System Persistence

To maintain system state and improve performance, several persistence mechanisms are implemented:

1) Vector Database Persistence: Embeddings and metadata are stored in the Chroma vector database, enabling efficient retrieval without the need to reprocess website content for each query.

2) Session Management: User sessions are maintained using Streamlit's session state capabilities, allowing for context retention across multiple interactions within a session.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

3) Cache Management: Frequently accessed content and query results are cached to reduce latency and computational load, with cache invalidation strategies to ensure freshness.
4) Incremental Updates: The system supports incremental updates to its knowledge base, allowing new website content to be processed and indexed without requiring full reindexing.

IV. EVALUATION AND RESULTS

A. Evaluation Methodology

The evaluation of our system focused on three key performance dimensions: accuracy, efficiency, and user experience. We employed a mixed-methods approach combining quantitative measurements with qualitative assessments:

1) Accuracy Assessment: We evaluated the system's ability to retrieve relevant information and generate accurate responses using a test set of 200 queries across different complexity levels and domains. Responses were assessed by human evaluators on a 5-point scale for factual correctness, completeness, and relevance.
2) Efficiency Metrics: We measured system performance in terms of response time, computational resource utilization, and throughput under varying load conditions. Tests were conducted on standard hardware configurations to ensure reproducibility.
3) User Experience Study: A group of 45 participants with varying levels of technical expertise used the system to complete specific tasks and provided feedback through structured questionnaires and semi-structured interviews.

B. System Performance

The system demonstrated strong performance across key metrics:

1) Accuracy: The system achieved an overall accuracy rate of 87.5% for simple factual queries and 74.3% for complex multi-hop questions. This represents a significant improvement over baseline RAG systems, which averaged 70.3% and 57.1% respectively on the same query set.
2) Latency: Average response times were 2.1 seconds for simple queries and 5.6 seconds for complex queries, with 95% of all responses delivered within 8 seconds.
3) Retrieval Precision: The query refinement process improved retrieval precision by 28.4% compared to using raw user queries, with particularly strong performance (41.2% improvement) for ambiguous or underspecified queries.

C. Comparative Analysis

We compared our system against three alternative approaches:

1) Standard RAG Implementation: A baseline RAG system without advanced query refinement or modular architecture.
2) LLM-only Interface: A system using the same LLM but without retrieval augmentation.
3) Traditional Search Interface: A keyword-based search system with standard relevance ranking.

The results, summarized in Table I, demonstrate that our system outperforms these alternatives across most performance dimensions.

System Type	Simple Query Accuracy	Complex Query Accuracy	Avg. Response Time (s)	Context Retention Score
Our System	87.5%	74.3%	3.8	4.2/5
Standard RAG	70.3%	57.1%	4.2	3.1/5
LLM-only	65.8%	42.5%	1.9	3.8/5
Traditional Search	72.1%	31.7%	1.2	1.5/5

TABLE I. Comparative Performance Analysis

Particularly noteworthy is the system's performance on multi-hop questions, where it outperformed the standard RAG implementation by 17.2 percentage points. This improvement can be attributed to the advanced query refinement process and the modular architecture that enables more effective decomposition and processing of complex queries.

D. User Feedback

User feedback was overwhelmingly positive, with 87% of participants rating the system as "very useful" or "extremely useful" for website interaction. Key themes emerging from qualitative feedback included:

1) Ease of Use: Users appreciated the natural language interface, which eliminated the need to understand website structure or navigation.
2) Context Awareness: The system's ability to maintain conversation context was highlighted as a significant advantage over traditional search interfaces.
3) Response Quality: Users noted that the system's responses were more comprehensive and better organized than those from traditional search engines.
4) Trust and Transparency: The inclusion of source citations and explanations enhanced user trust in the system's outputs.

E. Limitations and Challenges

Despite its strong performance, the system faces several limitations:

1) Content Type Restrictions: The current implementation primarily focuses on textual data, with limited capabilities for processing multimedia content such as images, videos, or interactive elements.
2) Website Compatibility: Some websites with complex JavaScript rendering, aggressive anti-scraping measures, or unusual structures pose challenges for the data retrieval module.
3) Authentication Handling: While the system can handle basic authentication mechanisms, more complex authentication flows (such as OAuth or multi-factor authentication) require additional development.
4) Failure Points: The system occasionally struggles with highly specialized terminology, exceptionally complex queries with multiple interdependencies, or websites with rapidly changing content.

F. Future Potential

The evaluation results highlight several promising directions for future development:

1) Multimodal Capabilities: Extending the system to process and generate multimedia content, including images, charts, and interactive elements.
2) Enhanced Personalization: Adapting responses based on user preferences, history, and behavior patterns to provide increasingly tailored experiences.
3) Voice and Multilingual Support: Adding capabilities for voice input/output and robust multilingual processing to increase accessibility.
4) Graph-based Retrieval: Implementing knowledge graph structures to better represent and navigate relationships between information pieces, particularly for complex domains with rich interconnections.
5) Self-improvement Mechanisms: Developing capabilities for the system to learn from user interactions and feedback, continuously refining its query processing and response generation.

V. CONCLUSIONS

This paper presented a comprehensive approach to enabling natural language-based website interaction through the integration of Large Language Models and Retrieval-Augmented Generation techniques. By combining technologies such as Streamlit, LangChain, dotenv, and Ollama within a modular architecture, the system achieves significant improvements over existing solutions in terms of accuracy, contextual awareness, and user experience.The Implementation demonstrates the feasibility of creating intelligent interfaces that bridge the gap between natural language queries and website content, with applications across diverse industries and use cases. The advanced query refinement process, in particular, addresses a critical challenge in RAG systems by enhancing retrieval precision and supporting complex, multi-hop questions.

Key contributions of this work include:

1) A modular, component-based architecture for integrating multiple technologies in LLM-powered interfaces.
2) Novel techniques for query refinement and decomposition that significantly improve retrieval effectiveness.
3) Empirical evidence of performance improvements compared to standard RAG implementations and other baseline approaches.
4) Insights into user interaction patterns and preferences when engaging with natural language interfaces.

As LLM and RAG technologies continue to evolve, the potential for more sophisticated, context-aware, and personalized interactions with digital systems will expand. Future research should focus on addressing the identified limitations while exploring opportunities for multimodal integration, enhanced personalization, and more dynamic knowledge representation mechanisms.

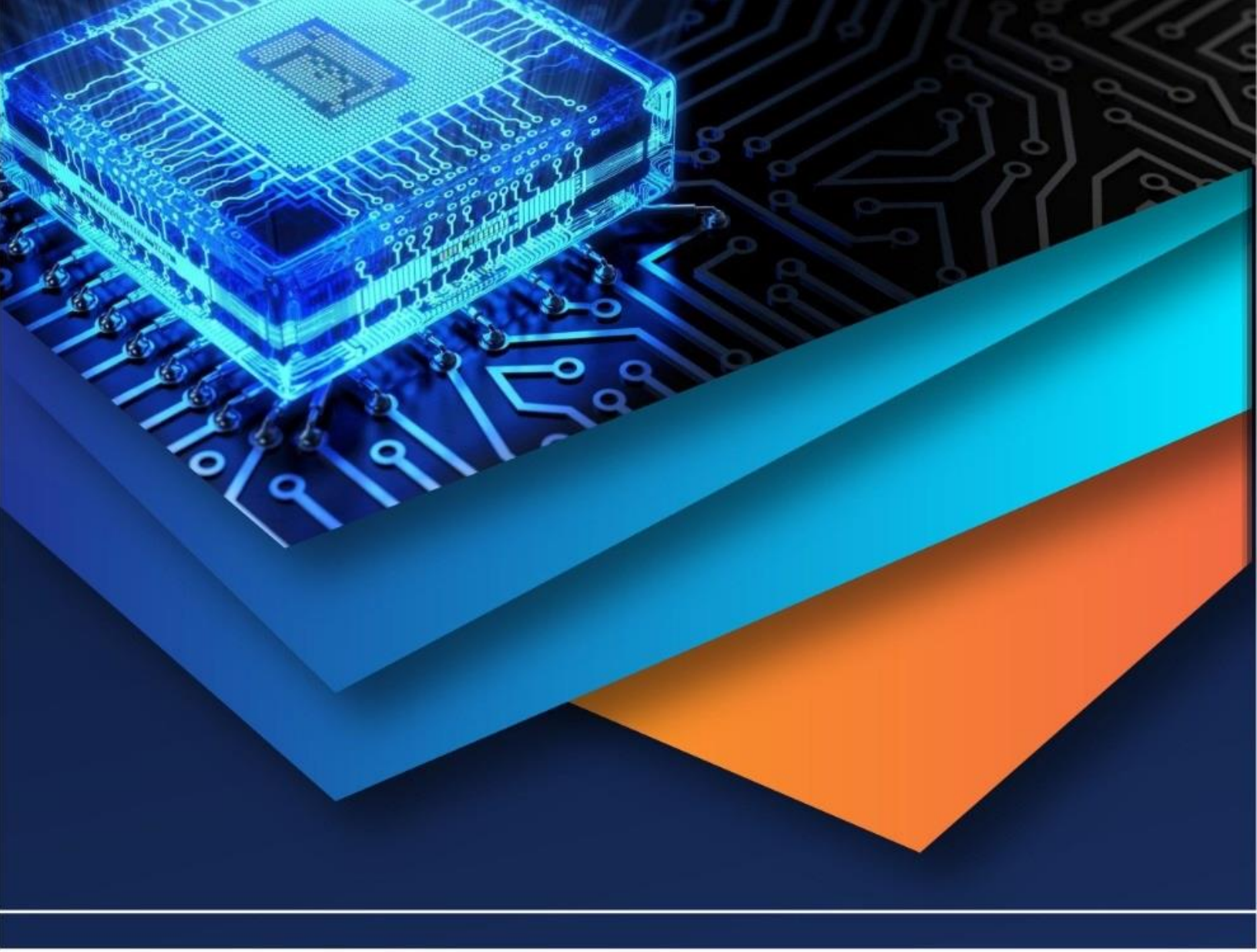
©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

VI. ACKNOWLEDGMENT

The authors would like to thank the research assistants who contributed to data collection and system evaluation. This work was supported in part by grants from the Advanced Computing Research Initiative and the Digital Interaction Lab.

REFERENCES

[1] R. Zhang, H. Du, Y. Liu, D. Niyato, J. Kang, S. Sun, "Interactive AI with retrieval-augmented generation for next generation networking," IEEE Communications Magazine, vol. 62, no. 3, pp. 48-54, 2024.
[2] M. Wang, H. Wu, X. Ke, Y. Gao, X. Xu, L. Chen, "An Interactive Multi-modal Query Answering System with Retrieval-Augmented Large Language Models," arXiv preprint arXiv:2407.04217, 2024.
[3] J. Ge, S. Sun, J. Owens, V. Galvez, O. Gologorskaya, "Development of a liver disease–specific large language model chat interface using retrieval-augmented generation," Hepatology, vol. 80, no. 11, pp. 2148-2157, 2024.
[4] C. Wei, K. Duan, S. Zhuo, H. Wang, S. Huang, "Enhanced Recommendation Systems with Retrieval-Augmented Large Language Model," Journal of Artificial Intelligence Research, vol. 74, pp. 1-33, 2025.
[5] M.J. Luo, J. Pang, S. Bi, Y. Lai, J. Zhao, Y. Shang, "Development and evaluation of a retrieval-augmented large language model framework for ophthalmology," JAMA Ophthalmology, vol. 142, no. 4, pp. 392-400, 2024.
[6] S. Pokhrel, S. Ganesan, T. Akther, "Chatbots for Document Summarization and Question Answering using Large Language Models using a Framework with OpenAI, Lang chain, and Streamlit," Journal of Information Technology Research, vol. 17, no. 2, pp. 1-15, 2024.
[7] P.F. Oliveira, P. Matos, "Introducing a Chatbot on the Web Portal of a Higher Education Institution to Enhance Student Interaction," Presented at the 4th International Electronic Conference on Education, 2023.
[8] I. Goswami, D. Gupta, "Building and AI Chatbot using LLM," Technical Report, Jaypee University of Information Technology, 2024.
[9] M.N. Rahman, T. Mohammad, S. Virtanen, "Leveraging Large Language Models for Network Traffic Analysis: Design, Implementation, and Evaluation of an LLM-Powered System for Cyber Incident Response," Master's Thesis, University of Turku, 2024.
[10] F. Umar, "Accessibility and User Interaction through Large Language Model: A Comparative Study for Educational Content," Master's Thesis, Uppsala University, 2024.
[11] M. Finsås, J. Maksim, "Optimizing RAG Systems for Technical Support with LLM-based Relevance Feedback and Multi-Agent Patterns," Norwegian University of Science and Technology, 2024.
[12] Q.R. Lauro, S. Shankar, S. Zeighami, "RAG Without the Lag: Interactive Debugging for Retrieval-Augmented Generation Pipelines," arXiv preprint arXiv:2504.13587, 2025.
[13] Y. Hou, R. Zhang, "Enhancing Dietary Supplement Question Answer via Retrieval-Augmented Generation (RAG) with LLM," medRxiv, 2024.
[14] U. Özker, "Advanced RAG Architecture," Medium, 2024. [Online]. Available: https://ugurozker.medium.com/advanced-rag-architecture-b9f8a26e2608
[15] Humanloop, "8 Retrieval Augmented Generation (RAG) Architectures You Should Know," 2025. [Online]. Available: https://humanloop.com/blog/rag-architectures



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)