Journal of
Electronics Technology Exploration

Homepage: htttps://shmpublisher.com/index.php/joetex
p-ISSN 3025-3470 | e-ISSN 3026-1066

# Implementation of Retrieval-Augmented Generation (RAG) and Large Language Models (LLM) for a Document and Tabular-Based Chatbot System

**Imam Chalish Rafidhul Haque**
Faculty of Mathematics and Science Education, Indonesia University of Education, Indonesia

**Article Info**

***Article history:***

Received Jun 20, 2025
Revised Jun 30, 2025
Accepted Jul 07, 2025

***Keywords:***

Retrieval-augmented Generation
Large language model
Text-to-SQL
Chatbot
ChromaDB

**ABSTRACT**

The challenge of accessing information from disparate sources—unstructured documents and structured tabular data—hinders efficiency in enterprise information systems. This study addresses this challenge by presenting the design, implementation, and validation of a unified chatbot system powered by Retrieval-Augmented Generation (RAG) and Large Language Models (LLM). For unstructured documents, the system implements a RAG pipeline utilizing ChromaDB for vector indexing and OpenAI embeddings. Meanwhile, for structured data, it leverages a Text-to-SQL engine to translate natural language queries into SQL commands, with results visualized via QuickChart. The architecture is built upon a modular FastAPI backend with role-based access control and was rigorously validated through blackbox functional testing. Results demonstrate 100% functional success across all endpoints, confirming the architecture's reliability. This study confirms the viability of a unified RAG and Text-to-SQL architecture, offering a practical blueprint for creating more intelligent and integrated data interaction systems in enterprise environments.



***Corresponding Author:***

Haque,
Faculty of Mathematics and Science Education,Indonesia University of Education,
Dr.Setiabudhi Street No.229 Bandung 40154, Jawa Barat, Indonesia.
Email: sekuniv.upi@upi.edu
https://doi.org/10.52465/joetex.v3i1.588

## 1. INTRODUCTION

Recent advancements in natural language processing (NLP) have established large language models (LLMs) like GPT-4 as powerful tools for understanding and generating human-like text [1], [2]. However, a primary limitation of these models in enterprise settings is their reliance on pre-trained, static knowledge. This makes them prone to generating outdated or factually incorrect informational phenomenon often referred to as "hallucination" [3] and, crucially, renders them unable to access or reason over private, domain-specific, or real-time documents created after their training cutoff date. To bridge this gap, Retrieval-Augmented Generation (RAG) was introduced as a paradigm to address this limitation by dynamically incorporating an external information retrieval step into the generation process [4]. RAG enables LLMs to ground their responses in specific, verifiable information from a designated knowledge base, thereby significantly improving the accuracy, relevance, and trustworthiness of their outputs [5], [6].

Concurrently, the application of LLMs has extended to structured data interaction, particularly through Text-to-SQL technology, which translates natural language questions into executable SQL queries [7], [8]. This innovation empowers non-technical users to query complex databases without needing to learn SQL syntax, effectively democratizing data access within organizations [9]. A significant body of research has focused on optimizing Text-to-SQL models for increasingly complex queries and diverse database schemas [10], [11]. However, while many researchers have explored RAG for document-based question answering [4], [12] and others have focused on Text-to-SQL for database querying [8], [13], these two powerful capabilities have largely evolved in parallel. This has resulted in a fragmented user experience where an employee might use a knowledge base search for internal documents but must turn to a separate business intelligence tool or a data analyst for insights from databases. This research addresses the limited scholarship on creating a unified, modular system that integrates both capabilities within a single, scalable backend architecture.

This research addresses the gap by designing and implementing a cohesive chatbot system that combines a RAG pipeline for unstructured document interaction with a Text-to-SQL module for structured data querying. While a few researchers have focused on building enterprise-level chatbots [14], [15] and others have utilized vector databases like ChromaDB for semantic search [16], the novelty of this work lies in its architectural integration. There have been limited studies concerned with developing an integrated backend system using a modern, high-performance framework like FastAPI [17] that provides modular, permission-based access (personal, departmental, and global) to both document and tabular data sources. Therefore, this research intends to develop and validate a robust backend architecture for a dual capability chatbot. The primary objectives of this research are to: (1) build a functional and reliable RAG pipeline for PDF documents; (2) implement an effective Text-to-SQL engine for CSV data; and (3) verify the system's modularity and functional reliability through blackbox testing, thereby demonstrating a practical path towards more versatile and explainable AI systems.

## 2. METHOD

The methodology for this research encompasses the system's architectural design, the technical implementation of its core components, and the validation procedures used to ensure its functional reliability.

### System Architecture

The system was designed as a modular backend using the FastAPI framework, selected for its high performance, native asynchronous support, and automatic API documentation capabilities, which streamline development and testing [17]. The architecture is logically divided into four primary modules: (1) a File Handling and Access Control Module responsible for uploads and permission management; (2) a RAG Core Module for processing queries against unstructured documents; (3) a Text-to-SQL Module for handling natural language queries over tabular data; and (4) a Utility Module for tasks like text summarization. This modular design allows for independent development, scaling, and maintenance of each functional component.

### RAG Pipeline Implementation

The document-based chatbot is powered by a RAG pipeline executed in several distinct steps:

1. Document Ingestion and Chunking: Upon a user uploading a document (e.g., PDF), the system utilizes a recursive character text splitter. This strategy segments the content into smaller, semantically coherent chunks of approximately 1000 characters with a 100-character overlap. This approach was chosen to maintain semantic context within each chunk while respecting the context window limitations of the downstream LLM.
2. Vector Embedding and Storage: Each text chunk is then converted into a high-dimensional vector representation using one of OpenAI's embedding models [18]. These vectors, which numerically capture the semantic meaning of the text, are indexed and stored in a ChromaDB vector database [16]. Each vector is stored with associated metadata, such as the document ID and chunk number, to enable precise source tracking.
3. Retrieval and Generation: When a user submits a query, it is also converted into a vector using the same embedding model. The system then performs a cosine similarity search in ChromaDB to retrieve the top-k (where k=4) text chunks most semantically relevant to the user's query. This retrieved context is then dynamically inserted into a structured prompt template before being passed to an LLM (e.g., GPT-3.5-Turbo, Deepseek R1). This method ensures that the generated response is explicitly "grounded" in the provided document, a core principle of the foundational RAG paper [4] that enhances factual consistency. The entire workflow is illustrated in Figure 1.

### Access Control and Testing

To ensure security and data segregation in a multi-user environment, the backend was structured with role-based access control (RBAC). The system defines three distinct roles ('user', 'operator', 'admin') to manage permissions for file access at personal, departmental, and global scopes. To validate the system's reliability, all API endpoints were subjected to positive path blackbox testing. This method involves creating a suite of tests that call each endpoint with valid payloads and authentication credentials. The HTTP response status and body content are then programmatically asserted against predefined expected outcomes to verify functional correctness without inspecting the internal code structure [19].

## 3. RESULTS AND DISCUSSIONS

This section presents the empirical results of the system's implementation, followed by a discussion of their significance and implications. The findings are organized according to the system's primary components: the RAG pipeline's performance, the Text-to-SQL module's efficacy, and the overall system reliability and architectural integrity.

### RAG Pipeline Performance

The RAG pipeline was validated through a series of qualitative tests where PDF documents of varying complexity and length were uploaded and queried. The results consistently demonstrated that the system could retrieve relevant document chunks using ChromaDB's semantic search capabilities and generate coherent, context-aware responses via the LLM. For example, when asked specific questions about numerical data or policy details located deep within a document, the chatbot correctly identified the relevant passage and formulated an answer based solely on that context. This grounding mechanism proved effective in mitigating the risk of factual hallucination [3], as the model's responses were tethered to the source material. Response latency was acceptable for interactive use, and the output remained faithful to the source document, demonstrating the practical effectiveness of the RAG architecture [4], [6] as illustrated in Figure 1.

The discussion of these results highlights the success of the chosen methodology. The chunking strategy was effective in preserving local context, while the semantic retrieval process was precise enough to locate the correct information. However, some limitations were observed. The performance of the RAG pipeline is highly dependent on the quality of the source document and the clarity of the user's query. Vaguely phrased questions sometimes led to the retrieval of broader, less specific chunks, resulting in more generic answers. This underscores the importance of prompt engineering and potential user guidance in a production-level system.

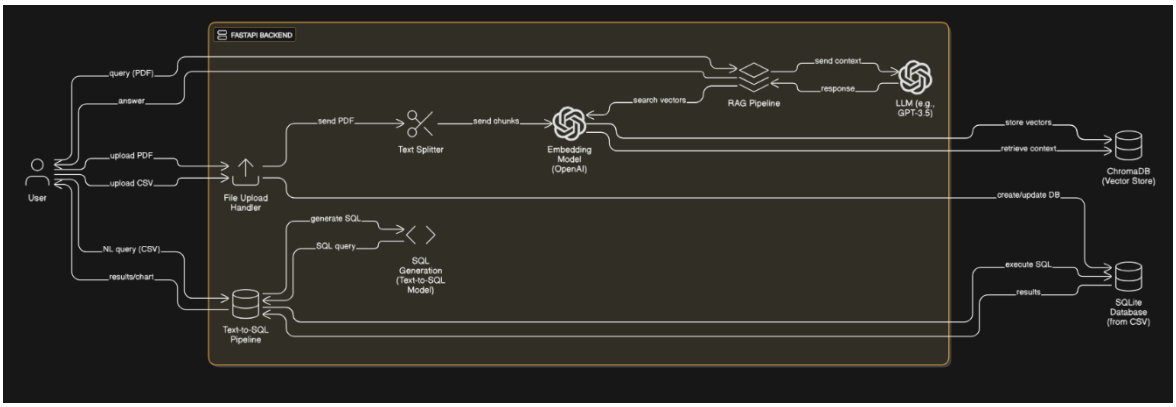


Figure 1. System architecture of the RAG and text-to-SQL chatbot

### Text-to-SQL Module Efficacy

For tabular queries, the Text-to-SQL engine successfully translated user intents into valid SQL statements. This capability enabled non-technical users to extract data insights without direct SQL knowledge, aligning with the project's goal of democratizing data analysis [9]. For instance, a natural language query like, "What were the average sales for each product last quarter?" was correctly converted into the corresponding SELECT AVG(Sales), Product FROM table GROUP BY Product; query. The system's ability to return both textual summaries and visual charts generated via an external API provided an intuitive and flexible way to interpret the data, as this dual-form output caters to users who prefer either raw numbers or visual representations.

In discussing these results, the Text-to-SQL module provides significant value by lowering the barrier to data analytics. However, the efficacy was most pronounced for queries of low-to-moderate complexity. The model sometimes struggled to correctly parse highly complex queries involving multiple sub-queries, complex joins, or ambiguous colloquial language known challenge in the broader Text-to-SQL research field. This

indicates that while the current implementation is highly functional for common business queries, further fine-tuning would be required to handle more advanced analytical scenarios.

**System Reliability and Architectural Integrity**

Blackbox testing was applied to all major API endpoints to verify their functional correctness and stability [19]. The testing focused on valid input scenarios (positive path) to ensure the reliability of core features like file processing, semantic search, and chat interaction. As shown in Table 1, all tested endpoints passed without failure, confirming the robustness of the backend logic.

The discussion of these findings centers on architectural choices. The 100% pass rate demonstrates that the modular design, built on the FastAPI framework [17], is not only stable but also well-structured. The clear separation of concerns between modules allowed for isolated testing and simplified debugging. This modularity also suggests a high potential for future scalability and ease of collaboration between development teams. Ultimately, the successful integration of these disparate technologies (RAG, Text-to-SQL, vector databases) into a single, reliable system represents the primary contribution of this work, addressing the notable gap in unified data interaction platforms for enterprises [14], [15].

Tabel 1. Summary of blackbox functional testing results

| Endpoint Module | Feature Tested | Input Type | Expected Output | Result |
|---|---|---|---|---|
| File Management | Document Upload (PDF) | Valid PDF file | Success message, file ID | Passed |
| RAG Chat | Query on document | Valid text query | Contextual text answer | Passed |
| CSV Management | CSV Upload | Valid CSV file | Success message, DB created | Passed |
| Text-to-SQL | Query on CSV data | Valid text query | Text summary and chart URL | Passed |
| Summarization | Summarize Document | Valid file ID | Concise summary text | Passed |

**Overall Implications and Limitations**

The successful development of this dual capability chatbot provides a practical blueprint for enterprises seeking to build unified data interaction platforms. Such systems can break down information silos by providing a single interface for employees to query diverse internal data sources. By grounding responses in verifiable documents and structured data, the system also enhances the trust and explainability of AI-driven insights.

However, studying has several limitations. First, the validation was primarily functional and qualitative; it did not include large-scale quantitative benchmarking of response accuracy or latency under heavy load. Second, the study did not involve real-world user testing to measure usability and satisfaction.

## 4. CONCLUSION

This study successfully demonstrated the design and implementation of an integrated chatbot system that leverages Retrieval-Augmented Generation (RAG) for querying unstructured documents and a Text-to-SQL engine for interacting with structured tabular data. The research achieved its primary objective of creating a unified, modular backend that bridges the gap between handling these two distinct data types. The document-based chatbot pipeline was able to retrieve semantically relevant information from user-provided content, offering meaningful and grounded responses. Concurrently, the Text-to-SQL module enabled intuitive interaction with CSV data by converting natural language into SQL, facilitating accessible data analysis and visualization.

As confirmed by positive path blackbox testing, all system components performed reliably, with endpoints consistently returning valid and expected results. The modular architecture, built with FastAPI, proves to be a scalable and maintainable approach for developing complex AI systems. In conclusion, combining RAG and LLM technologies with a backend-driven, file-centric chatbot architecture presents a promising direction for building explainable, adaptive, and user-friendly AI systems in document-intensive domains. Future work could focus on enhancing the Text-to-SQL engine to handle more complex, multi-turn conversational queries, expanding support for additional database types, and conducting large-scale user studies to measure performance and satisfaction in a real-world operational environment.

## REFERENCES

[1] A. Vaswani and others, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[2] T. B. Brown and others, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, 2020, pp. 1877–1901.

[3] J. Ji and others, “Survey of hallucination in natural language generation,” *ACM Comput. Surv.*, vol. 55,

no. 12, pp. 1–38, 2023, doi: 10.1145/3571730.
[4] P. Lewis and others, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, 2020, pp. 9459–9474.
[5] K. Guu and others, "Retrieval augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 3929–3938.
[6] A. Asai and others, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proceedings of the International Conference on Learning Representations*, 2024.
[7] T. Yu and others, "Spider: A large-scale human-labeled dataset for complex and cross-domain text-to-SQL task," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3911–3921.
[8] B. Li and others, "A survey on text-to-SQL: Concepts, methods, and future directions," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 6876–6888.
[9] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, "A decentralized privacy-preserving healthcare blockchain for IoT," *Sensors (Switzerland)*, vol. 19, no. 2, pp. 1–17, 2019, doi: 10.3390/s19020326.
[10] J. Scholak and others, "Picard: Parsing incrementally for constrained auto-regressive decoding from language models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 9546–9559.
[11] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.
[12] G. Nguyen *et al.*, "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 77–124, 2019, doi: 10.1007/s10462-018-09679-z.
[13] J. R. Saura, B. R. Herraez, and A. Reyes-Menendez, "Comparing a traditional approach for financial brand communication analysis with a big data analytics technique," *IEEE Access*, vol. 7, pp. 37100–37108, 2019, doi: 10.1109/ACCESS.2019.2905301.
[14] A. Følstad and P. B. Brandtzæg, "Chatbots and the new world of HCI," *Interactions*, vol. 24, no. 4, pp. 38–42, 2017, doi: 10.1145/3085558.
[15] C. Shang and F. You, "Data Analytics and Machine Learning for Smart Process Manufacturing: Recent Advances and Perspectives in the Big Data Era," *Engineering*, vol. 5, no. 6, pp. 1010–1016, 2019, doi: 10.1016/j.eng.2019.01.019.
[16] ChromaDB, "Chroma - The AI-native open-source embedding database." 2023.
[17] S. R. Montoliu, "FastAPI Documentation." 2023.
[18] OpenAI, "New and improved embedding model." 2022.
[19] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed. Hoboken, NJ, USA: John Wiley \& Sons, 2012.