An Intelligent Conversational Agent Using Self-Reflective Retrieval-Augmented Generation for Enhanced Large Language Model Support in National Accounts Learning

M Farhan[1*], Yunofri[2], E Tasriah[2], L H Suadaa[1], S Pramana[1]

[1] STIS Polytechnic of Statistics, East Jakarta, Indonesia
[2] BPS-Statistics Indonesia, Jakarta, Indonesia

*Corresponding author's email: farhan082002@gmail.com

Abstract. BPS Statistics Indonesia plays a strategic role in compiling balance sheet statistics as the foundation for national policy analysis. This role requires a deep understanding of the concepts, definitions, and compilation standards outlined in the System of National Accounts (SNA) manual. However, in practice, comprehending such complex technical documents is not always straightforward. To address this challenge, this study proposes the development of an intelligent conversational agent in the form of a chatbot that implements the Self-Multimodal RAG approach. This approach integrates self-reflection mechanisms to generate more accurate and relevant responses. The evaluation was conducted using the LLM-as-a-Judge framework across four metrics: answer correctness, answer relevancy, context relevancy, and context faithfulness. Experimental results demonstrate that the Self-Reflective RAG achieved a score of 80% on the answer correctness metric, with competitive performance in terms of relevancy and faithfulness. From the chatbot implementation perspective, black-box testing confirmed that all functionalities operated as expected, while system usability testing using the CSUQ instrument yielded a score of 74.704%, indicating that the chatbot is well-accepted by users.

Keyword: Chatbot, Large Language Model, Self-Reflective RAG, System of National Account.

1. Introduction

BPS Statistics Indonesia is a government institution that plays a strategic role in providing high-quality statistical data to support national development planning. One of BPS Statistics Indonesia's core responsibilities is compiling balance sheet statistics, which serve as the foundation for various policy analyses. In practice, this task requires a thorough understanding of the concepts, definitions, and compilation standards stipulated in the System of National Accounts (SNA) manual. However, comprehending such highly technical documents is not always straightforward. This challenge arises from the use of highly specialized terminology, dense document structures, and complex concepts that demand advanced analytical reasoning. Consequently, employees may encounter difficulties in interpreting these concepts and applying them consistently in the compilation of balance sheet statistics.

To address these challenges, one potential approach is the utilization of Artificial Intelligence (AI) technologies, particularly Large Language Models (LLM). LLM-based technologies have demonstrated remarkable capabilities in comprehending documents, generating contextual text with high accuracy, and enabling conversational or question–answering mechanisms [1]. Furthermore, LLMs have been adopted by various statistical organizations worldwide to enhance business process efficiency [2]. For example, the Australian Bureau of Statistics (ABS) employs LLMs to accelerate the updating of task lists within the Australian and New Zealand Standard Classification of Occupations (ANZSCO). Similarly, the Central Statistics Office (CSO) of Ireland leverages LLMs to translate code from SAS to R, supporting the organization's transition to a more modern and efficient programming ecosystem. The International Monetary Fund (IMF) has also developed StatGPT, a generative AI–based prototype that enables users to access statistical data through a natural language interface. Unlike these statistical organizations that have already integrated LLM technologies into their business processes, BPS Statistics Indonesia has not yet implemented them on a large scale. In this regard, this study proposes the development of an intelligent conversational agent, or chatbot, powered by LLMs, which would enable employees to learn about national accounts by interactively posing questions related to concepts, definitions, and compilation standards.

Chatbot technology has advanced significantly and demonstrated rapid progress across various domains, including education, healthcare, and public services. In the specific context of education and teaching, chatbots have proven effective in increasing availability and facilitating access to learning services [3]. Beyond that, they also enrich students' learning experiences by providing motivational support, enhancing accessibility, and offering direct online assistance [4]. The use of chatbots has also begun to be implemented in the government sector as an effort to improve the effectiveness of communication between governments and citizens. For instance, [5] developed a chatbot designed to bridge two-way interactions in the context of public services. The approach integrated natural language processing, machine learning, and data mining technologies to create richer and more expressive communication channels, drawing on regulatory documents, government operational data, and social media as data sources. In the field of training, chatbots are increasingly being utilized as interactive tools to enhance the effectiveness of learning processes. For example, [6] proposed an innovative solution in the form of an LLM-based chatbot functioning as a startup training simulator. This system is capable of addressing iterative challenges in startup development through a conversational interface powered by LLMs such as GPT-3.

The primary approach employed in the implementation of the chatbot in this study is Retrieval-Augmented Generation (RAG), which combines the natural language understanding capabilities of LLMs with an additional mechanism for retrieving relevant information from external documents, such as the SNA manual. The use of RAG has been shown to effectively reduce factual errors or hallucinations that may arise from LLMs when performing tasks requiring specialized knowledge [7], [8]. Consequently, RAG has become the predominant approach for generating contextual answers grounded in domain-specific knowledge to mitigate factual inaccuracies or hallucinations [9]. For example, [10] developed a RAG-based medical chatbot to help the public access reliable health information, particularly on infectious diseases, by leveraging medical documents in PDF format from trusted sources such as Elsevier and The New York Times. Meanwhile, [11] explored the application of RAG in the context of financial report analysis with the aim of assisting individual investors in reviewing and interpreting quarterly or semiannual bank reports to support more informed investment decisions.

Although RAG can enhance response generation, in practice it may still provide irrelevant contextual information to the LLM, which can lead to inaccurate and low-quality responses [12]. This issue arises because the retrieval process by default does not consider whether the retrieved information truly contributes to generating the correct answer [13]. To address this problem, several Advanced RAG frameworks have been developed. Among them are Self-Improve [14] and Self-Refine [15], which employ feedback loop mechanisms to autonomously correct errors in generated responses. Other approaches such as Self-Correction [16] and Self-Reasoning [17] integrate reasoning processes to improve the effectiveness of information retrieval. In addition, Self-Reflective [18] introduces reflection tokens as explicit signals that help the model evaluate the quality of retrieved information based on relevance, as well as assess the utility and factual support of the generated responses. With this mechanism, Self-Reflective RAG has been proven to outperform RAG-based ChatGPT and Llama2-Chat across various tasks, including open-domain question answering, reasoning, and fact verification, particularly in terms of accuracy. This approach was subsequently adopted in this study as the foundational framework to improve the quality of generated responses.

Building upon this foundation, this study aims to implement Self-RAG as the primary approach in an LLM-based chatbot for generating final answers to employees' questions regarding national accounts. Through its reflective mechanism, the chatbot is expected to provide more accurate and relevant information while simultaneously offering BPS Statistics Indonesia employees easy, flexible, and adaptive access to learning in understanding national accounts concepts and methodologies.

2. Research Method

2.1. Dataset

The primary data sources used as the knowledge base in this study consist of the SNA 2008 manual and six national accounts training modules developed by the Training and Education Center (Pusdiklat) of BPS Statistics Indonesia in 2024. To support the evaluation of the RAG approach, a test dataset comprising 105 question–answer pairs was constructed, all of which directly reference the content of the knowledge base. These questions were designed to reflect various types of information relevant to the context of national accounts, covering conceptual, quantitative, analytical, and methodological aspects. The information regarding the token statistics of the test dataset is presented in table 1.

Table 1. Token statistics of the test dataset.

Statistics	Question tokens	Answer tokens
Average	26.67	64.66
Minimum	5.00	7.00
Maximum	116.00	275.00

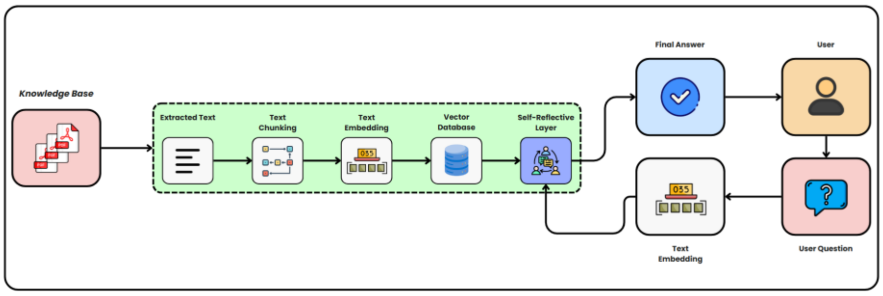
2.2. Document Ingestion and Indexing

To process data from the knowledge base in PDF format, a parsing stage is required as the initial step. In this study, parsing was carried out using the gptpdf package, which leverages the PyMuPDF library to extract text elements from documents with the assistance of large visual models such as GPT-4o, and subsequently converts the extracted text areas into markdown format. This was followed by a chunking process, in which lengthy extracted texts were divided into smaller segments to enable efficient processing by the LLM. The resulting text chunks were then embedded and stored in a vector database after undergoing the embedding process using text-embedding-3-small.

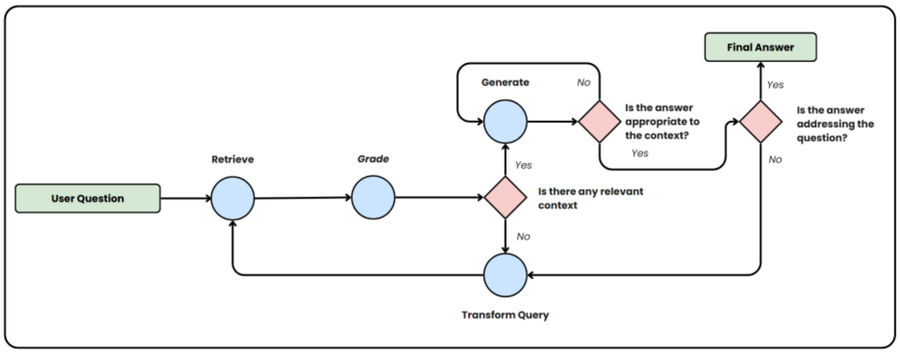
In this study, ChromaDB was selected as the vector database due to its open-source nature and seamless integration with the LangChain framework, which serves as the primary foundation for system development. The text-embedding-3-small model was chosen because it has been optimized to work synergistically with the main LLM employed in this research, namely GPT-4o-mini, thereby ensuring that retrieved contexts used during inference are more relevant and semantically meaningful.

2.3. Self-Reflective RAG

The components and workflow of the Self-Reflective RAG approach are illustrated in figure 1. In general, the workflow consists of four main stages: context retrieval, relevance verification, answer generation, and answer verification. All of these stages were implemented using LangGraph, with each stage represented as a node corresponding to an LLM object.



(a)


Final Answer
Generate
No
Yes
Is the answer appropriate to the context?
Yes
Is the answer addressing the question?
Retrieve
Grade
User Question
Yes
Is there any relevant context
No
No
Transform Query

(b)

Figure 1. Components and workflow of Self-Reflective RAG, showing: (a) the Self-Reflective RAG components, and (b) the detailed workflow inside the Self-Reflective Layer.

The explanation of each process in the Self-Reflective RAG workflow is outlined as follows.

2.3.1. Document Retrieval. The document retrieval process employed cosine similarity as the metric for measuring semantic closeness between documents. The cosine similarity value between two text representation vectors, $\vec{u}$ and $\vec{v}$, ranges from -1 to 1. A value closer to 1 indicates a high degree of semantic similarity between the two texts. Conversely, a value of 0 denotes no semantic similarity, while a value of -1 indicates that the two texts are semantically opposite in meaning [19]. The formula for cosine similarity is as follows.

$$cos\ cos\ (\vec{u}, \vec{v}) = \frac{\vec{u}.\vec{v}}{||\vec{u}||\ ||\vec{v}||} = \frac{\sum_{i=1}^{n} \quad u_i . v_i}{\sqrt{\sum_{i=1}^{n} \quad u_i^2}\sqrt{\sum_{i=1}^{n} \quad v_i^2}} \tag{1}$$

In this process, for each query q, the contexts C within the corpus are ranked in descending order based on their cosine similarity scores with respect to the query. Subsequently, only the top four contexts with the highest cosine similarity values are selected for further processing in the next stage.

2.3.2. Relevance Verification. To monitor the context retrieval process, a dynamic state object was used as the medium for data transmission between nodes. The relevance verification stage was specifically executed at the grade node (see figure 1b). This node receives as input the user's question and the retrieved contexts transmitted via the state object from the retrieve node. The grade node then checks the relevance of each received context against the user query and generates a binary flag of either "yes" or “no”. If the flag is yes, the process proceeds to the generate node to produce an answer. Conversely, if the flag is no, the process is redirected to the transform query node, where the query is reformulated for greater optimization before the retrieval process is repeated at the retrieve node.

2.3.3. Answer Generation. The answer generation stage is executed at the generate node, which receives as input the user’s question along with the contexts deemed relevant by the grade node, transmitted through the state object. In practice, the generate node produces an initial answer based on the provided document contexts. This answer is then evaluated in the answer verification stage before being delivered to the user as the final response.

2.3.4. Answer Verification. The answer verification process is carried out through two conditional edges that serve as sequential evaluation pathways. In the first conditional edge, the system verifies whether the generated answer is consistent with and supported by the relevant retrieved contexts. In other words, this verification assesses the alignment between the answer and its supporting contexts. The outcome of this process is a binary flag of either “yes” or “no”. If the flag is no, the system repeats the answer generation process at the generate node. Conversely, if the flag is yes, the process proceeds to the second conditional edge.

In the second conditional edge, verification focuses on the correspondence of the answer to the user’s question, ensuring that the generated response truly addresses the essence of the query. This process also produces a binary flag of either “yes” or “no”. If the flag is yes, the answer is considered valid and delivered to the user as the final response. However, if the flag is no, the system redirects the process to the transform query node. A summary of the verification stages in the Self-Reflective RAG workflow is presented in table 2.

Table 2. Summary of the verification stage in the Self-Reflective RAG workflow.

Type of verification	Input	Output	Action
Relevance of context to the user’s query	q, d	YES	Proceed to generate node
		NO	Redirect to transform query node
Consistency of the answer with the relevant retrieved context	A, C	YES	Proceed to verification stage between the answer and the query
		NO	Return to generate node
Consistency of the answer with the user’s query	q, A	YES	Deliver final answer to the user
		NO	Redirect to transform query node

In this context, q represents the user query or question, d denotes the set of retrieved contexts, A refers to the answer generated by the LLM, and C corresponds to the context from the relevant documents.

2.4. Evaluation
To evaluate the effectiveness and performance of the implemented Self-Reflective RAG, a series of experiments were conducted against several comparative approaches. In this case, two variations were tested in addition to Self-Reflective RAG itself, namely the baseline and vanilla RAG. In the baseline experiment, questions from the test dataset were sent directly to the LLM without going through a retrieval process. This approach served as an initial benchmark to compare the effectiveness of the RAG approach in the national accounts case study. Meanwhile, the vanilla RAG approach represents the basic form of Retrieval-Augmented Generation, which only combines the retriever and generator modules

without an additional reflective mechanism. In this study, the vanilla RAG was executed by omitting the self-reflection process embedded in the Self-Reflective RAG. The workflow began with text extraction, followed by chunking. The resulting text chunks were then converted into vector representations using the text-embedding-3-small model, with the outputs stored in a vector database as the search basis. For each question in the test dataset, the system retrieved the most relevant text context using the cosine similarity metric. These retrieved context snippets were subsequently passed to the LLM as augmented context to generate the final answer.

The performance evaluation process of the RAG approach in this study was carried out using the LLM-as-a-Judge method with four main metrics [20]. The first is answer correctness, which evaluates the accuracy of the LLM-generated answer relative to the reference answer. The second metric is answer relevancy, which assesses whether the generated answer is relevant to the given question. The third metric is context faithfulness, which examines the consistency and alignment between the generated answer and the provided textual context. Lastly, context relevancy measures the degree of relevance of the textual context to the user's question. The LLM-as-a-Judge method was selected because LLMs have been proven capable of mimicking human-like reasoning and decision-making processes. Moreover, this approach offers a more cost-efficient solution compared to using human evaluators and can be easily scaled to accommodate large-scale evaluation needs [21]. In practice, this evaluation method employs specific evaluators for each metric, which construct prompts to be executed by the LLM, subsequently producing binary assessment outputs (0 or 1) along with the rationale behind each decision. A value of 1 indicates that the answer meets the evaluation criterion, while a value of 0 indicates otherwise. The final score for each metric is obtained by calculating the average of all binary evaluation results across the test dataset. A summary of the evaluation metrics used is presented in table 3, while the general formula for calculating the final score of each metric is shown as follows.

$$\underline{S} = \frac{1}{n}\sum_{i=1}^{n} s_i \tag{2}$$

with:

n	: The total number of question–answer pairs in the test dataset.
s_i	: The binary value (0 or 1) assigned by the evaluator for the i-th question.
$\underline{S}$	: The average binary score obtained from the evaluation of all questions in the test dataset.

Table 3. Evaluation metrics and required inputs.

Metric	Required input
Answer Correctness	q, A, RA
Answer Relevancy	q, A
Context Faithfulness	A, Ctx
Context Relevancy	q, Ctx

In this case, RA refers to the reference answer in the test dataset, and Ctx denotes the relevant context obtained from the retrieval process.

2.5. Chatbot Application Construction

In this study, the chatbot application was developed in an end-to-end manner, covering both the backend and the user interface (frontend), with Self-Reflective RAG serving as the main approach employed in the inference or answer generation process. To support communication between system components, the Self-Reflective RAG pipeline was deployed using FastAPI to provide API endpoints that enable integration and interaction with the user interface. The chatbot application itself was developed using the Next.js framework with JavaScript as the programming language and Tailwind CSS as the styling framework, ensuring a responsive and visually appealing user interface. User data and conversation history were stored in a MongoDB database due to its fast read–write performance, making it highly suitable for real-time applications such as chatbots. Overall, the architecture of the developed chatbot application is illustrated in figure 2.

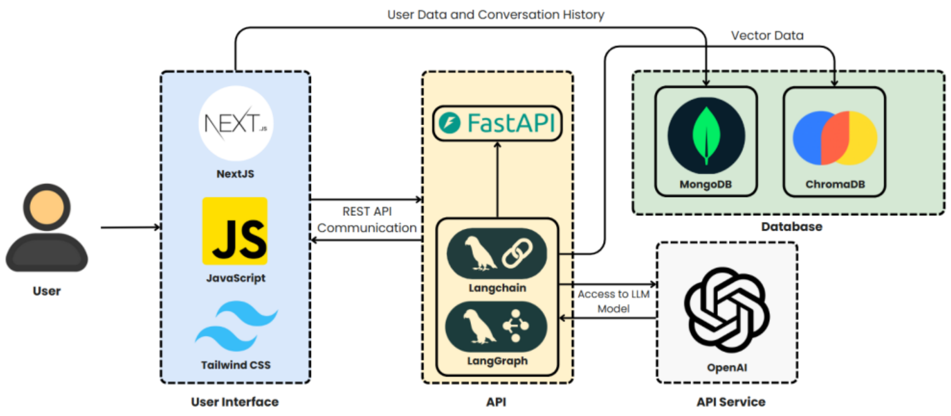


Figure 2. Chatbot application architecture.

To assess the quality and feasibility of the developed chatbot application, evaluations were conducted on two main aspects: system functionality and system usability. For the functionality aspect, testing was performed using black-box testing by examining each core feature of the chatbot and verifying whether the output aligned with the predefined expectations. Meanwhile, for the usability aspect, the evaluation employed the Computer System Usability Questionnaire (CSUQ), which is designed to measure users' perceptions regarding interface quality, ease of use, and overall satisfaction in interacting with the system. The details of the questionnaire items used in this instrument are presented in table 4.

Table 4. CSUQ questionnaire questions.

Question no.	Question content
1	Overall, I am satisfied with the ease of use of this chatbot application.
2	It is very easy to use this chatbot application.
3	I can complete my tasks (finding answers to questions related to the National Balance System) quickly using this chatbot application.
4	I feel comfortable using this chatbot application.
5	It is very easy to learn how to use this chatbot application.
6	I am confident that I can quickly become productive using this chatbot application.
7	The chatbot provides clear error messages that help me understand how to fix problems.

8	When I make a mistake while using the chatbot, I can recover and continue using it easily and quickly.
9	The information provided (on-screen response messages) is clear and easy to understand.
10	I can easily find the information I need when using this chatbot.
11	The information provided by this chatbot helps me complete my tasks (finding answers to questions related to the System of National Account).
12	The organization of information on the chatbot interface is neat and clear.
13	The interface design of this chatbot application is pleasant.
14	I like the interface design of this chatbot application.
15	This chatbot has all the functions and capabilities I expect.
16	Overall, I am satisfied with this chatbot.

There are four main types of scores generated from the CSUQ questionnaire [22], namely: (1) the overall score, calculated as the average of all questions (items 1 to 16); (2) the system usability score, which is the average of items 1 to 6; (3) the information quality score, based on the average of items 7 to 12; and (4) the interface quality score, obtained from the average of items 13 to 15. Each item is rated on a Likert scale of 1 to 7, with lower scores indicating a higher level of user satisfaction. In addition, CSUQ assessments can also be converted into percentages to facilitate the quantitative interpretation of results using the following formula [23].

$$CSUQ = 100 - \left(\frac{\sum_{n=1}^{16} CSUQ_n}{16} - 1\right) \times \frac{100}{6} \tag{3}$$

with:

$CSUQ$: The overall CSUQ score expressed as a percentage.

$CSUQ_n$: The score of the n-th CSUQ question.

In this context, the evaluation scores converted into a percentage scale can be interpreted as indicators of the level of user acceptance of the system. A score above 70 indicates that the system is deemed feasible and acceptable to users, as it has met expectations in terms of usability, information quality, and interface design. Conversely, a score below 50 suggests that the system is considered unacceptable, as it fails to provide an adequate user experience. Meanwhile, scores ranging between 50 and 70 reflect a marginal or unstable level of user satisfaction, thereby requiring further improvements to achieve an optimal level of acceptance.

3. Result and Discussion

3.1. Main Results

The experimental performance of the Self-Reflective RAG approach and its comparative variations is presented in table 5. Based on table 5, it can be observed that each pipeline configuration yields different performance across the evaluation metrics. In the answer correctness metric, the Self-Reflective RAG approach demonstrates the best performance with a score of 0.80, higher than both the Baseline (0.66) and Vanilla RAG (0.56). This indicates that the reflective mechanism implemented in Self-Reflective RAG is capable of improving the accuracy of the model's generated answers. In the answer relevancy metric, all three approaches show consistent results with a high score of 0.98, indicating that whether without retrieval or with different RAG variations, the generated answers are relatively aligned with the given questions.

Meanwhile, in the evaluation dimension that involves context, a significant difference is observed. On the context faithfulness metric, Vanilla RAG only recorded a score of 0.42, whereas Self-Reflective RAG achieved 0.70, indicating that the reflection mechanism helps the model maintain consistency between the generated answers and the context retrieved from the knowledge base. The Baseline, however, has no value on this metric since it does not involve any context retrieval process. For the context relevancy metric, both Vanilla RAG and Self-Reflective RAG obtained the same score of 0.93, showing that the retrieved documents are consistently relevant to the given questions. As with the previous case, this metric is not applicable to the Baseline. Overall, these results demonstrate that Self-Reflective RAG provides a significant improvement on the answer correctness and context faithfulness metrics compared to Vanilla RAG, confirming that the addition of a reflective mechanism contributes positively to answer quality, particularly in terms of accuracy and alignment with the available context.

Table 5. Performance evaluation results.

Approach type	Answer corretness	Answer relevancy	Context faithfulness	Context relevancy
Baseline	0.66	0.98	–	–
Vanilla RAG	0.56	0.98	0.42	0.93
Self-Reflective RAG	0.80	0.98	0.70	0.93

To estimate the resources required for the inference process in each evaluated approach, a systematic observation was conducted on cost and time consumption through direct measurements using the Application Programming Interface (API) service provided by OpenAI as the inference model provider. This observation was carried out during the testing phase with the test dataset for each approach. Before and after the execution of testing, cost consumption data was monitored via the OpenAI developer dashboard to identify the difference in resource usage. The difference was then divided by the number of questions in the test dataset, which on average contained 26.6 tokens per question, to obtain the estimated average cost per query. Meanwhile, the estimation of time consumption was calculated by recording the total duration required by each approach to complete the inference process for all questions in the test dataset. The recorded time was then divided by the total number of questions, resulting in the estimated average inference duration per query. This method was chosen because it can represent the actual cost and time requirements of the entire inference process, including additional overheads caused by the dynamic and iterative reflection mechanism in the Self-Reflective RAG. Detailed results regarding the estimated operational time and cost for each RAG approach are presented in table 6.

Table 6. Average inference time and cost per question.

Approach Type	Average inference time per question (seconds)	Average inference cost per question (USD)
Baseline	5.36	0.000190
Vanilla RAG	3.86	0.000095

Self-Reflective RAG	10.71	0.001714

Based on the information presented in table 6 above, it is evident that Vanilla RAG demonstrates the most efficient performance in terms of both time and cost. Conversely, the Self-Reflective RAG approach records the highest levels of time and cost consumption. The elevated inference overhead in the Self-Reflective RAG approach is primarily attributed to the iterative nature of the reflection process. These results indicate that while the reflection mechanism can improve answer quality, it also entails significant overhead in terms of inference time and cost.

3.2. Analysis

Although the Self-Reflective RAG approach has been proven to improve the quality and accuracy of answers, further analysis shows that this method is not always superior in every case. It was found that approximately 9.5% of the questions could not be answered correctly by Self-Reflective RAG but were successfully answered by other approaches. Two examples are presented in table 7.

Table 7. Examples of questions that could not be answered correctly by the Self-Reflective RAG approach but were answered correctly by other approaches.

Question	Reference answer	Correctly answered by
Produk yang masih tersedia dalam penguasaan produsen dikateogrikan sebagai apa? *Products that remain under the control of the producer are categorized as what?*	Produk yang masih tersedia dalam penguasaan produsen dikateogrikan sebagai persediaan (inventori). *Products that remain under the control of the producer are categorized as inventories.*	*Baseline, Vanilla RAG*
Apa syarat seorang individu dikatakan sebagai anggota rumah tangga (ART)? *What are the criteria for an individual to be considered a member of a household?*	Seorang individu dianggap sebagai Anggota Rumah Tangga (ART) jika menetap atau berniat untuk menetap pada bangunan tempat tinggal rumah tangga tersebut. *An individual is considered a member of a household if they reside or intend to reside in the dwelling occupied by that household.*	*Baseline*

In the first question in table 7 above, the Self-Reflective RAG approach produced an answer stating that products available under the control of producers are categorized as cultivated biological resources. The evaluator assessed this answer as incorrect because the concept of cultivated biological resources differs from inventory, which was the correct reference answer. This error occurred due to the stringent verification process in the Self-Reflective RAG workflow, which places excessive emphasis on literal alignment with the context, thereby overlooking correct information that is not explicitly reflected in the text of the question. In the second question, the Self-Reflective RAG approach produced an answer that an individual is considered a household member if they are an individual or a group of individuals who share the same residence, pool part or all of their income and assets, and collectively consume certain goods and services, particularly food and housing. Although this answer includes several relevant elements, the evaluator deemed it incorrect because it does not directly address the fundamental criterion stated in the reference answer, namely residing or intending to reside in the household dwelling. These two examples demonstrate that while the self-reflection mechanism in Self-Reflective RAG is effective in reducing hallucination, it may also expand the scope of answers or overlook the core concept that should be used to correctly respond to the question. Furthermore, when the analysis is extended to the combination of evaluation metrics used, two notable findings emerge. The first is cases where the answer was judged correct, but the context used was deemed irrelevant. In the Self-Reflective RAG approach, such cases occurred at a proportion of 3.8%. The second is cases where the answer was also judged correct, but the context used failed to meet the criterion of faithfulness. The proportion of such cases in the Self-Reflective RAG approach was 22.8%. These findings indicate that the internal knowledge of the LLM contributed to producing a final answer that was still judged correct.

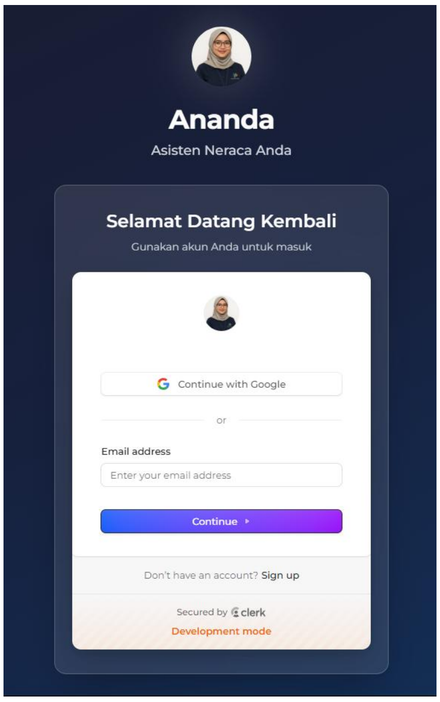
3.3. Results of Chatbot Application Development

The presentation of the chatbot application development results in this study is divided into two main parts: the final implementation and the evaluation. The implementation section focuses on the preview of the application's interface, while the evaluation section outlines the results of application testing, both in terms of system functionality and usability level. A detailed explanation of each part is presented as follows.

3.3.1. Final Implementation. In the developed chatbot application, there are three main pages that serve as the core components of the user interaction flow with the application, namely the login/register page, the chat page, and the admin page. These three pages are designed to support the overall functionality of the application, ranging from user authentication and authorization, conversational interaction, to data management by the administrator. The details of each page are explained as follows.

A. Login/Register Page

This page serves as the main entry point for every user before they can access and further interact with the chatbot application. On this page, users are required to log in using an already registered account, or to register first if they do not yet have one. The authentication and authorization processes in this application are supported by Clerk services, which provide a secure and efficient identity management system. The interface of the login and register pages can be seen in figures 3 and 4 below.



Ananda

Asisten Neraca Anda

Selamat Datang Kembali

Gunakan akun Anda untuk masuk

Continue with Google

or

Email address

Enter your email address

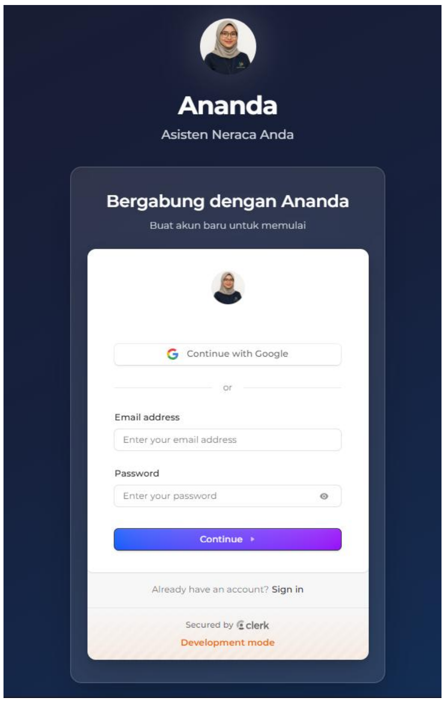
Continue

Don't have an account? Sign up

Secured by clerk

Development mode

Figure 3. Login page.



Ananda

Asisten Neraca Anda

Bergabung dengan Ananda

Buat akun baru untuk memulai

Continue with Google

or

Email address

Enter your email address

Password

Enter your password

Continue

Already have an account? Sign in

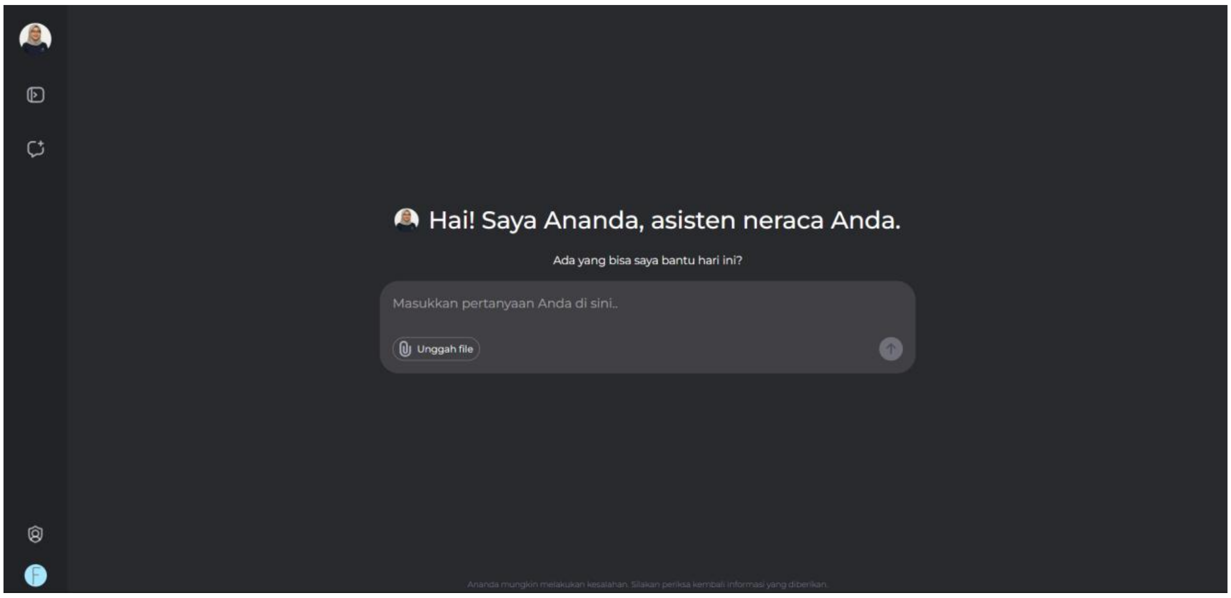
Secured by clerk

Development mode

Figure 4. Register page.

B. Chat Page

This page serves as the main interface used by users to engage in conversations with the developed chatbot application. On this page, users can send questions and receive response answers based on the queries submitted. The initial display of the chat page can be seen in figure 5 below.



Hai! Saya Ananda, asisten neraca Anda.

Ada yang bisa saya bantu hari ini?

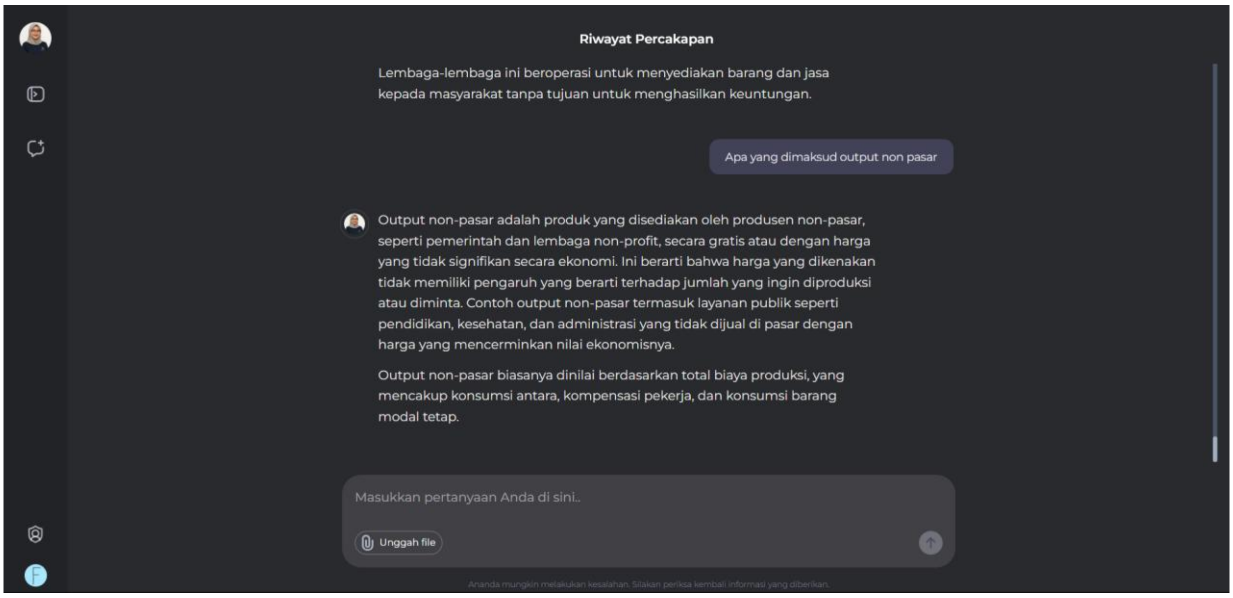
Masukkan pertanyaan Anda di sini..

Unggah file

Ananda mungkin melakukan kesalahan. Silakan periksa kembali informasi yang diberikan.

Figure 5. Initial display of the chat page.

After the user submits a question and receives a response, the chat page will automatically display the conversation history between the user and the chatbot, as shown in figure 6. This history is presented in a dialog format, making it easier for users to revisit previous interactions.



Riwayat Percakapan

Lembaga-lembaga ini beroperasi untuk menyediakan barang dan jasa kepada masyarakat tanpa tujuan untuk menghasilkan keuntungan.

Apa yang dimaksud output non pasar

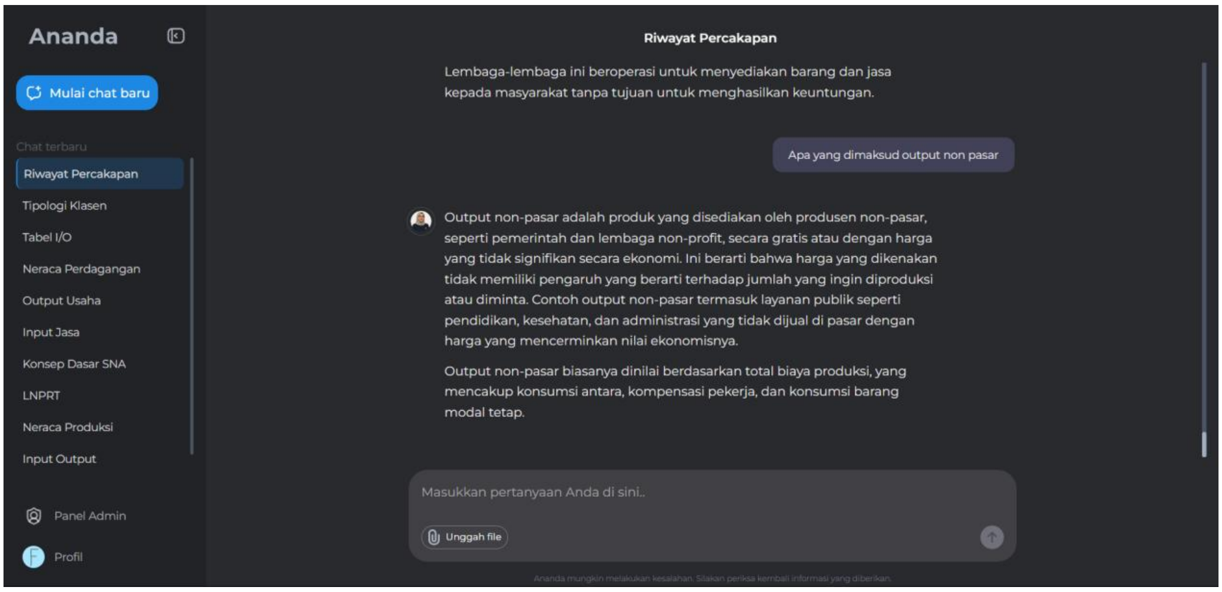
Output non-pasar adalah produk yang disediakan oleh produsen non-pasar, seperti pemerintah dan lembaga non-profit, secara gratis atau dengan harga yang tidak signifikan secara ekonomi. Ini berarti bahwa harga yang dikenakan tidak memiliki pengaruh yang berarti terhadap jumlah yang ingin diproduksi atau diminta. Contoh output non-pasar termasuk layanan publik seperti pendidikan, kesehatan, dan administrasi yang tidak dijual di pasar dengan harga yang mencerminkan nilai ekonomisnya.

Output non-pasar biasanya dinilai berdasarkan total biaya produksi, yang mencakup konsumsi antara, kompensasi pekerja, dan konsumsi barang modal tetap.

Masukkan pertanyaan Anda di sini..
Unggah file
Ananda mungkin melakukan kesalahan. Silakan periksa kembali informasi yang diberikan.

Figure 6. Chat page display with conversation history.

On this chat page, users can also utilize the sidebar to perform various additional functions. For instance, they can start a new conversation session, browse and select previous conversations, rename existing conversations, or delete those that are no longer needed. The display of the chat page with the sidebar can be seen in figure 7 below.



Riwayat Percakapan

Lembaga-lembaga ini beroperasi untuk menyediakan barang dan jasa kepada masyarakat tanpa tujuan untuk menghasilkan keuntungan.

Apa yang dimaksud output non pasar

Output non-pasar adalah produk yang disediakan oleh produsen non-pasar, seperti pemerintah dan lembaga non-profit, secara gratis atau dengan harga yang tidak signifikan secara ekonomi. Ini berarti bahwa harga yang dikenakan tidak memiliki pengaruh yang berarti terhadap jumlah yang ingin diproduksi atau diminta. Contoh output non-pasar termasuk layanan publik seperti pendidikan, kesehatan, dan administrasi yang tidak dijual di pasar dengan harga yang mencerminkan nilai ekonomisnya.

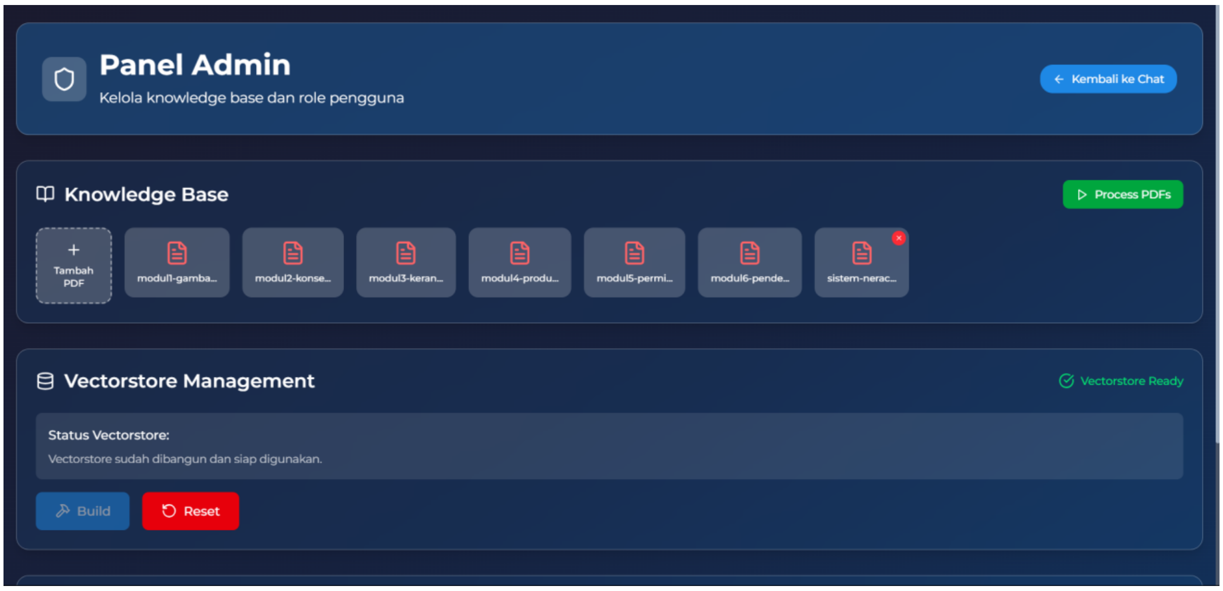
Output non-pasar biasanya dinilai berdasarkan total biaya produksi, yang mencakup konsumsi antara, kompensasi pekerja, dan konsumsi barang modal tetap.

Masukkan pertanyaan Anda di sini..
Unggah file
Ananda mungkin melakukan kesalahan. Silakan periksa kembali informasi yang diberikan.

Figure 7. Chat page display with conversation sidebar.

C. Admin Page

The admin page is specifically designed to manage the knowledge base and vector store. In terms of accessibility, this page can only be accessed by users with the admin role to ensure system security and guarantee that data management is carried out by authorized personnel. Through this page, the admin has the authority to add, modify, or delete the knowledge base according to the needs and dynamics of the available information. In line with this, the admin can also update the vector store, especially when changes or additions occur in the knowledge base, so that the system continues to provide relevant and contextual retrieval results. The display of the admin page in the developed chatbot application can be seen in figure 8 below.



Panel Admin

Kelola knowledge base dan role pengguna

Kembali ke Chat

Knowledge Base

Process PDFs

Tambah PDF | modul1-gamba... | modul2-konse... | modul3-keran... | modul4-produ... | modul5-permi... | modul6-pende... | sistem-nerac...

Vectorstore Management

Vectorstore Ready

Status Vectorstore:

Vectorstore sudah dibangun dan siap digunakan.

Build | Reset

Figure 8. Knowledge base and vector store management page display.

3.3.2. Evaluation. After the final implementation was completed, a comprehensive evaluation was carried out on the developed chatbot application. Two main types of evaluation were applied, namely functionality testing and system usability testing, detailed as follows.

A. System Functionality Testing

System functionality testing was conducted to ensure that each feature in the developed chatbot application operates according to the specifications and expectations. The method used in this testing is black-box testing, in which the tester provides input as a user without accessing or knowing the application's source code, and then observes the system's output. The features tested in this process include registration, login and logout, profile management, interaction and session management, knowledge base management, vector store management, and user role management. Based on the testing conducted, it was found that all functional requirements of the system for these features were successfully met.

B. System Usability Testing

The system usability testing was conducted to evaluate the extent to which the developed chatbot application can be accepted and effectively used by end users. In this study, the primary instrument employed was the CSUQ, which is designed to measure users' perceptions of system usability quality. The testing process involved 18 respondents, all of whom were employees of Statistics Indonesia (BPS) working in the field of national accounts. The evaluation results are summarized in table 8, which presents the average scores for each assessment indicator. Based on these results, it can be concluded that the developed chatbot has generally met user expectations and is considered suitable for use. This finding is further supported by an overall average score that exceeds the feasibility threshold of 70%, achieving a final score of 74.07%. Moreover, when examined by individual assessment criteria, it was found that the system usability criterion obtained a score of 2.435, information quality scored 2.657, and user interface quality scored 2.592. Overall, these results indicate that the aspects of system usability, information quality, and interface design were well received by end users. Nevertheless, some respondents expressed concerns regarding the inaccuracy of certain answers as well as the relatively long inference process. The issue of inaccurate answers is essentially caused by the limited coverage of information in the knowledge base used, as well as frequent errors in the relevant context filtering process within the Self-Reflective approach. Meanwhile, the slow inference process is indeed a consequence of the complexity of the iterative workflow in the Self-Reflective RAG approach, which involves multiple stages.

Table 8. Results of usability testing using CSUQ

Question no.	Criteria	Score	Criteria score
1	System usability	2.444	2.435
2		1.777	
3		2.833	
4		2.500	
5		2.277	
6		2.77	
7	Information quality	2.833	2.657
8		2.666	
9		2.555	
10		2.944	
11		2.833	
12		2.111	
13	User interface quality	2.222	2.592
14		2.388	
15		3.166	
16		2.555	
Average score		2.555	
Percentage conversion		74.074%	

4. Conclusion

This study specifically implements and evaluates Self-Reflective RAG as an approach to support the learning process of national accounts through LLM-based question answering tasks. The approach integrates an internal reflection mechanism that enables the model not only to generate answers but also to critically assess the relevance of the retrieved context and the quality of the responses produced. Evaluation results in the domain of the national accounts system show that Self-Reflective RAG achieved an answer accuracy rate of 80% and an answer relevance rate of 98%, while also demonstrating significant improvements compared to both the baseline approach and Vanilla RAG. Nevertheless, further analysis revealed that this approach still faces challenges in improving accuracy in certain cases as well as in computational efficiency due to the iterative nature of the reflection process. Therefore, the findings of this study open new avenues for further investigation, including the application of more precise context selection techniques, optimization of the inference pipeline, and the exploration of more diverse knowledge bases to broaden the coverage and flexibility of the system.

References

[1] M. Shanahan, "Talking about Large Language Models," Commun. ACM, vol. 67, no. 2, pp. 68–79, Jan. 2024, doi: 10.1145/3624724.

[2] C. Curtin et al., "Large Language Models for Official Statistics," Dec. 2023.

[3] S. Wollny, J. Schneider, D. Di Mitri, J. Weidlich, M. Rittberger, and H. Drachsler, "Are We There Yet? - A Systematic Literature Review on Chatbots in Education," Front Artif Intell, vol. 4, 2021, doi: 10.3389/frai.2021.654924.

[4] S. Cunningham-Nelson, W. Boles, L. Trouton, and E. Margerison, "A Review of Chatbots in Education: Practical Steps Forward," in 30th Annual Conference for the Australasian Association for Engineering Education (AAEE 2019): Educators Becoming Agents of Change: Innovate, Integrate, Motivate, Brisbane, Queensland: Engineers Australia, 2019, pp. 299–306. [Online]. Available: https://search.informit.org/doi/10.3316/informit.068364390172788

[5] A. Androutsopoulou, N. Karacapilidis, E. Loukis, and Y. Charalabidis, "Transforming the communication between citizens and government through AI-guided chatbots," Gov Inf Q, vol. 36, no. 2, pp. 358–367, 2019, doi: https://doi.org/10.1016/j.giq.2018.10.001.

[6] J. B. Ilagan and J. R. Ilagan, "A Prototype Of A Chatbot For Evaluating and Refining Student Startup Ideas Using A Large Language Model," Dec. 2023. doi: 10.35542/osf.io/azhf9.

[7] O. Ram et al., "In-Context Retrieval-Augmented Language Models," 2023. [Online]. Available: https://arxiv.org/abs/2302.00083

[8] A. Asai, S. Min, Z. Zhong, and D. Chen, "Retrieval-based Language Models and Applications," in Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts), Y.-N. (Vivian) Chen, M. Margot, and S. Reddy, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 41–46. doi: 10.18653/v1/2023.acl-tutorials.6.

[9] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[10] S. K. S, J. W. K. G, G. M. K. E, M. R. J, R. G. Singh A, and Y. E, "A RAG-based Medical Assistant Especially for Infectious Diseases," in 2024 International Conference on Inventive Computation Technologies (ICICT), 2024, pp. 1128–1133. doi: 10.1109/ICICT60155.2024.10544639.

[11] I. Iaroshev, R. Pillai, L. Vaglietti, and T. Hanne, "Evaluating Retrieval-Augmented Generation Models for Financial Report Question and Answering," Applied Sciences, vol. 14, no. 20, 2024, doi: 10.3390/app14209318.

[12] F. Shi et al., "Large Language Models Can Be Easily Distracted by Irrelevant Context," in Proceedings of the 40th International Conference on Machine Learning, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., in Proceedings of Machine Learning Research, vol. 202. PMLR, Apr. 2023, pp. 31210–31227. [Online]. Available: https://proceedings.mlr.press/v202/shi23a.html

[13] T. Gao, H. Yen, J. Yu, and D. Chen, "Enabling Large Language Models to Generate Text with Citations," 2023. [Online]. Available: https://arxiv.org/abs/2305.14627

[14] J. Huang et al., "Large Language Models Can Self-Improve," in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 1051–1068. doi: 10.18653/v1/2023.emnlp-main.67.

[15] A. Madaan et al., "Self-Refine: Iterative Refinement with Self-Feedback," in Advances in Neural Information Processing Systems, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., Curran Associates, Inc., 2023, pp. 46534–46594. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf

[16] S. Welleck et al., "Generating Sequences by Learning to Self-Correct," in The Eleventh International Conference on Learning Representations , 2023. [Online]. Available: https://openreview.net/forum?id=hH36JeQZDaO

[17] Y. Xia, J. Zhou, Z. Shi, J. Chen, and H. Huang, "Improving Retrieval Augmented Language Model with Self-Reasoning," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, no. 24, pp. 25534–25542, Apr. 2025, doi: 10.1609/aaai.v39i24.34743.

[18] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," 2023. [Online]. Available: https://arxiv.org/abs/2310.11511

[19] C. Yao and S. Fujita, "Adaptive Control of Retrieval-Augmented Generation for Large Language Models Through Reflective Tags," Electronics (Basel), vol. 13, no. 23, 2024, doi: 10.3390/electronics13234643.

[20] X. Zhang et al., "GPT-4V(ision) as a Generalist Evaluator for Vision-Language Tasks," ArXiv, vol. abs/2311.01361, 2023, [Online]. Available: https://api.semanticscholar.org/CorpusID:264935635

[21] J. Gu et al., "A Survey on LLM-as-a-Judge," 2025. [Online]. Available: https://arxiv.org/abs/2411.15594

[22] J. Lewis, "Measuring Perceived Usability: The CSUQ, SUS, and UMUX," Int J Hum Comput Interact, vol. 34, pp. 1–9, Aug. 2018, doi: 10.1080/10447318.2017.1418805.

[23] M. Santórum G. et al., “A Digital Platform for Respiratory Rehabilitation in Patients with post-COVID19: Design and Usability Evaluation,” Aug. 2023. doi: 10.54941/ahfe1004289.