

## APPLICATION OF GPT IN CHATBOTS TO FACILITATE KNOWLEDGE MANAGEMENT SYSTEM INTERACTION USING LANGCHAIN (CASE STUDY: PT SOFTBLESS SOLUTIONS)

Bagus Tri Mahardika<sup>1</sup>, Albantani Meiditya Hasan<sup>2</sup>

Universitas Darma Persada, Jakarta, Indonesia

Email: [bagusunsada@gmail.com](mailto:bagusunsada@gmail.com)<sup>1</sup>, [Figri177@gmail.com](mailto:Figri177@gmail.com)<sup>2</sup>

### ABSTRACT

*Searching the knowledge base on the system running at PT Softbless Solutions has the disadvantage of only being able to use keywords. Search results are also often irrelevant to what employees want. With the widespread use of AI in chatbots, it can improve information searches that are more interactive for employees. In this research, the methodology used in system development was the RAD (Rapid Application Development), and for bot development was the RAG (Retrieval Augmented Generation) method, which makes it possible to develop LLM such as ChatGPT to use existing knowledge base data in finding the desired information. This RAG implementation uses the LangChain framework and ChromaDB database to store data in embedding form. In the final result, the chatbot can be accessed in web applications and Telegram bots, and it achieves a User Acceptance Test value of 96.4%, which shows that the system and bot that have been built have a high level of usability. This research is expected to improve employee interaction in searching the knowledge base and become a reference for similar research.*

**KEYWORDS** Chatbot; ChromaDB; Knowledge Base; LangChain; Retrieval Augmented Reality



*This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International*

### INTRODUCTION

PT Softbless Solutions already has a Knowledge Management System (KMS) application to manage the knowledge base in the form of articles owned by the company. However, the interaction between employees and the existing knowledge base is still not good (Ali et al., 2016; Anwar et al., 2024; Buntoro et al., 2023; Dei, 2021; Natek & Lesjak, 2021). This is because the Knowledge Management System (KMS) can only search for knowledge in the form of articles based on keywords only. The search results obtained show many articles that are not relevant to the employee's wishes. With these limitations, it will certainly add more time to find relevant knowledge to existing questions or problems (Cerchione & Esposito, 2017; Choi et al., 2020; Montoya-Quintero et al., 2022; Nova et al., 2023; Zanker & Bureš, 2022).

In the digital era, efficient knowledge management is critical for organizational success, yet many companies struggle with ineffective systems that hinder productivity (Choi et al., 2020; Haass et al., 2023). According to a report by McKinsey & Company (2023), employees spend nearly 20% of their workweek searching for information, leading to significant time wastage and reduced operational efficiency. This global challenge underscores the need for innovative solutions to streamline knowledge retrieval and enhance workplace productivity.

Data from existing knowledge can be used as a data source for ChatGPT to produce responses such as questions and answers to problems or employee needs. This can be done by the RAG (Retrieval Augmented Generation) method. The RAG method is a combination of pre-trained language models by accepting external databases to expand knowledge and produce dynamic answers. The implementation of the RAG method can be done using the LangChain

framework. LangChain is a framework for developing applications that use the Large Language Model and its goal is to allow developers to easily leverage other data sources and interact with other applications.

A study by Gartner (2024) revealed that 65% of organizations face difficulties in managing their knowledge bases, with employees often encountering irrelevant search results. For instance, PT Softbless Solutions, the case study in this research, reported that their keyword-based Knowledge Management System (KMS) frequently delivers unrelated articles, exacerbating employee frustration and delaying problem resolution. These findings highlight the limitations of traditional search mechanisms in modern workplaces.

The specific issue addressed in this research is the inefficiency of keyword-based searches in KMS, which fail to accommodate natural language queries or contextual understanding. At PT Softbless Solutions, this limitation has resulted in prolonged search times and diminished user satisfaction, as employees cannot easily access relevant knowledge. This problem is particularly acute in multilingual environments where queries may be posed in Indonesian or English.

Recent studies have explored the potential of AI-driven chatbots to improve knowledge management. For example, Topsakal & Akinci (2023) demonstrated the effectiveness of LangChain in developing Large Language Model (LLM) applications, while Radeva et al. (2024) highlighted the benefits of Retrieval-Augmented Generation (RAG) for enhancing search accuracy. However, these studies primarily focused on technical implementations without addressing real-world organizational challenges, such as multilingual support and user acceptance.

Despite advancements in AI and chatbot technologies, there remains a gap in research on their practical application within specific organizational contexts, particularly in non-English-speaking environments. Existing studies often overlook the integration of these technologies with legacy systems and the measurement of their usability through empirical testing. This research aims to bridge this gap by focusing on PT Softbless Solutions as a case study.

The urgency of this research lies in the growing demand for interactive and efficient knowledge management tools. With the rapid adoption of AI technologies, organizations must adapt to remain competitive. Addressing the inefficiencies of traditional KMS through AI-driven solutions can significantly enhance employee productivity and organizational performance, making this research timely and relevant.

This study introduces novelty by combining the LangChain framework and ChromaDB for embedding storage to create a multilingual chatbot capable of understanding and responding to queries in both Indonesian and English. Unlike previous works, this research emphasizes user-centric design and evaluates system performance through a comprehensive User Acceptance Test (UAT), providing actionable insights for real-world deployment.

The purpose of this research is to develop and implement a GPT-powered chatbot using the RAG method and LangChain framework to facilitate efficient knowledge management at PT Softbless Solutions. The study aims to enhance search accuracy, reduce response times, and improve user satisfaction by leveraging AI technologies tailored to the organization's specific needs.

This research contributes to both academia and industry by demonstrating the practical application of RAG and LangChain in a real-world setting. It provides a blueprint for

integrating AI-driven chatbots with existing KMS, offering a scalable solution for other organizations facing similar challenges. Additionally, the study contributes empirical data on the usability and effectiveness of such systems through UAT results.

The findings of this research have significant implications for knowledge management practices. By showcasing the potential of AI to transform traditional systems, the study encourages organizations to adopt innovative technologies. Furthermore, the success of the implemented chatbot at PT Softbless Solutions serves as a model for other companies, paving the way for broader adoption of AI-enhanced knowledge management tools in diverse linguistic and operational contexts.

## RESEARCH METHOD

The data collection for this research involved multiple methods to ensure comprehensive and accurate information. First, a literature study was conducted to review previous research related to ChatGPT, LangChain, and chatbots, providing theoretical support for the study. Additionally, observation was carried out by monitoring how employees at PT Softbless Solutions interact with the existing Knowledge Base. Interviews with these employees were also conducted to gather deeper insights and firsthand information relevant to the research objectives.

For system development, the research employed the Retrieval Augmented Generation (*RAG*) method to enable GPT to utilize data from existing Knowledge Bases effectively. The web application was designed and built using the Rapid Application Development (*RAD*) model, known for its fast development cycle, typically completed within 60 to 90 days. The *RAD* model follows three main stages: requirements gathering, system design, and implementation, facilitating an efficient and iterative approach to software development.

## RESULT AND DISCUSSION

### System Needs Analysis

Based on the results of data collection on employees at PT Softbless Solutions' research site, the following system needs include:

1. Needs of IT Administrators/Employees
  - a. Manage User data
  - b. Added Open API Keys, Telegram Tokens and Webhooks via UI
  - c. Admins can import knowledge data from SQL files that have been exported from running applications
2. Employee Needs
  - a. Manage article data or knowledge in web-based applications
  - b. Can change account information such as username, firstname, lastname, email and password
  - c. Can access Chatbots on Telegram applications and bots
  - d. No need for chatbot history on the app
  - e. Chatbot must be able to accommodate Indonesian and English multilingual
  - f. Can ask questions or prompts on chatbots
  - g. The chatbot can only retrieve answers to questions given from existing knowledge data and added by employees to the application

- h. Employees cannot register on the application considering that the employees at the research site are small
- i. The app account is created by the IT staff as the application administrator

## System Design

The system design stage describes how the system as a whole will be created.

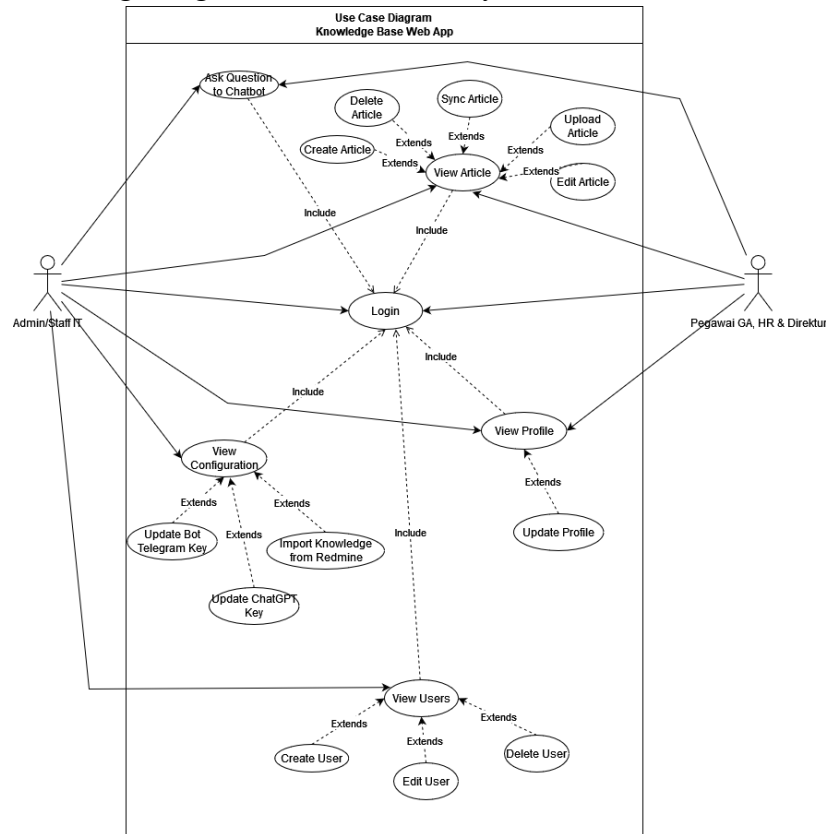


Figure 1. Use Case Diagram

Figure 2 shows how articles are saved into *embedding* so that they can be used by *chatbots*.

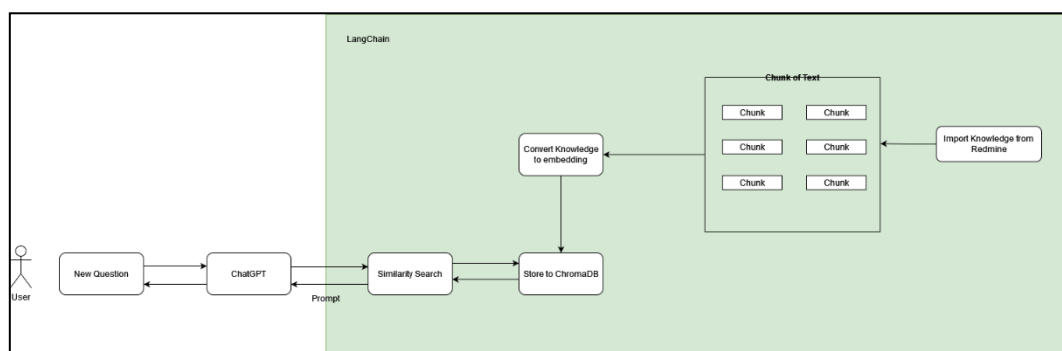


Figure 2. Flow Chart RAG

## Interface Display Design

In this stage of the user interface design is created to simplify the implementation process and ensure that it is in accordance with the needs.

### Login View Design

In figure 3, the login display design has a *username* and *password field*. so that you can log in.

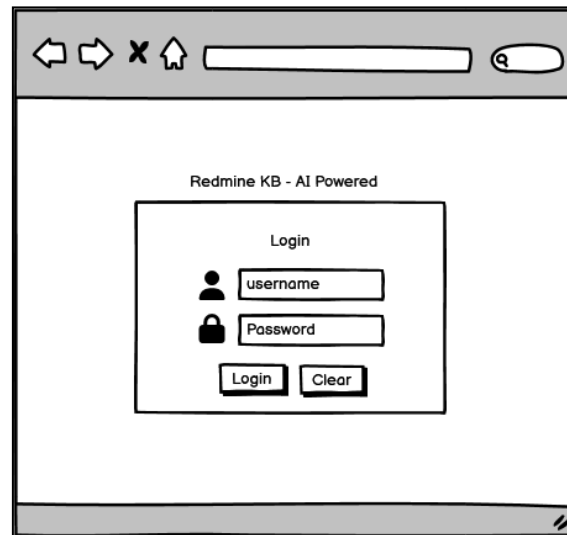


Figure 3. Login View Design

### Bot Display Design

After the user successfully logs in, the system will immediately redirect to the main page displaying the Chatbot. In figure 4, users can ask for information from the chatbot.

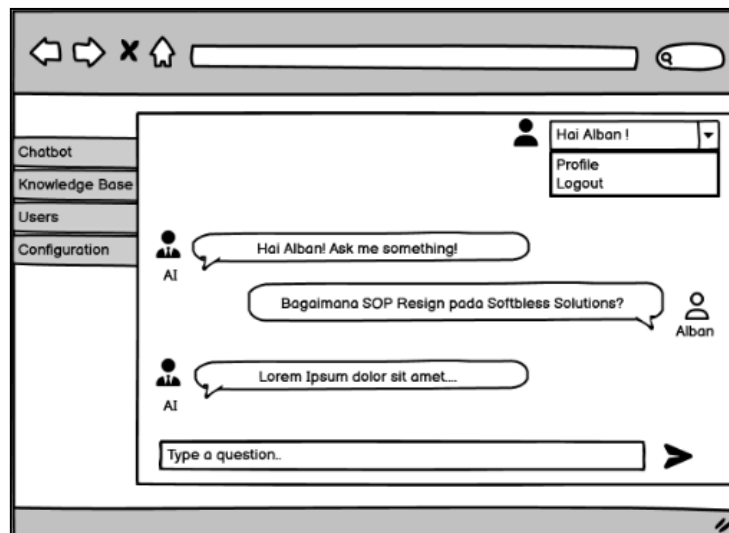


Figure 4. Bot Display Design

### Knowledge Page Display Design

Figure 5 shows a table display containing data from *knowledge* in the form of articles. Figure 6 is the design of the display when adding a new article and in figure 7 the display when the user sees the content of the article.

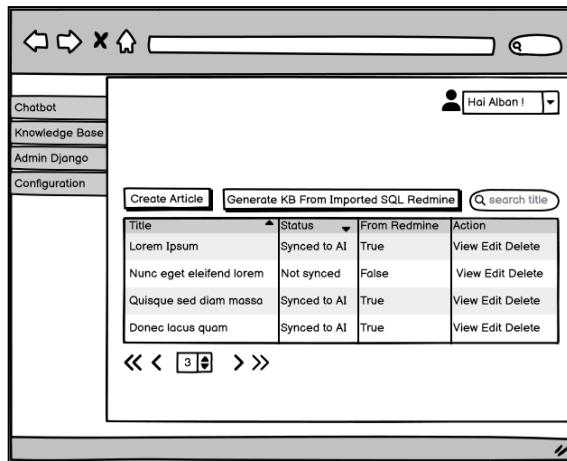


Figure 5. Knowledge List Display Design

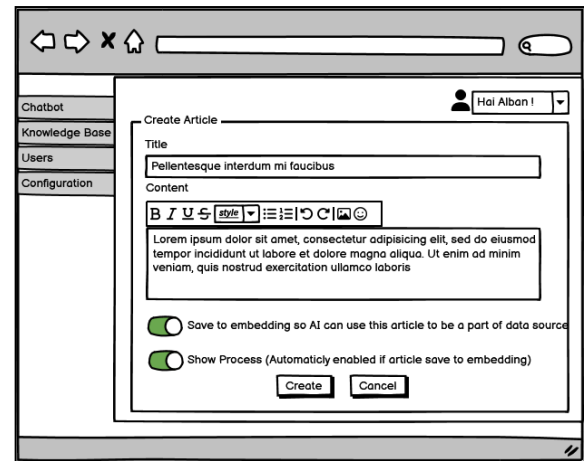


Figure 6. Display Design Add Article

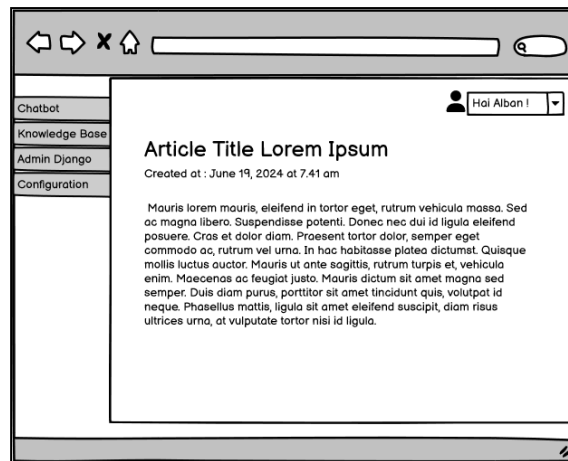


Figure 7. Display Design View Article Details

### Configuration Page View Design

In figure 8, the configuration page layout design has 3 fields to store OpenAI Key, Telegram Token, and Telegram WebHook URL

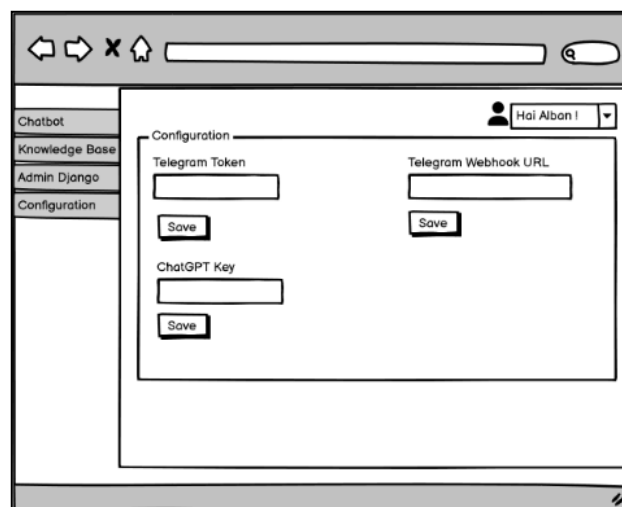
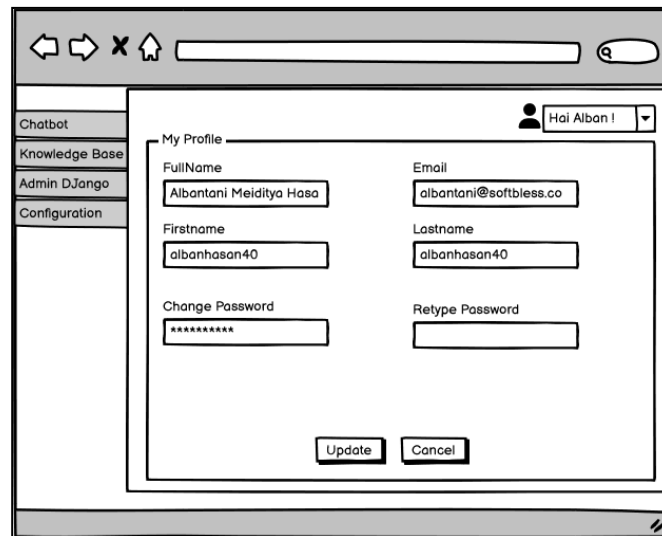


Figure 8. Configuration Page View Design

## User Profile Page Display Design

In figure 9, users can change their respective account profiles.



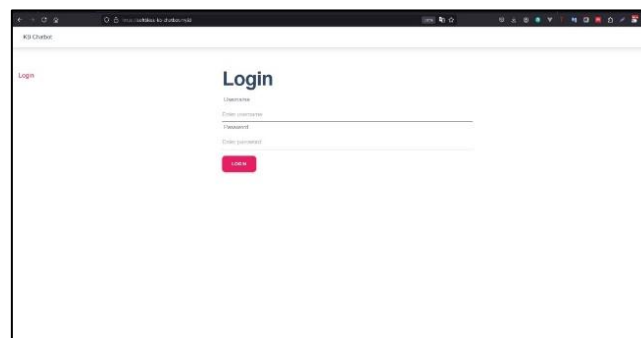
The image shows a web browser window with a sidebar menu on the left containing 'Chatbot', 'Knowledge Base', 'Admin DJango', and 'Configuration'. The main content area is titled 'My Profile' and features a user greeting 'Hai Alban!'. The profile form includes fields for 'FullName' (Albantani Meiditya Hasa), 'Email' (albantani@softbless.co), 'Firstname' (albanhasan40), and 'Lastname' (albanhasan40). There are also fields for 'Change Password' (masked with asterisks) and 'Retype Password'. At the bottom of the form are 'Update' and 'Cancel' buttons.

Figure 9. User Profile Display Design

## Implementation

### System Implementation

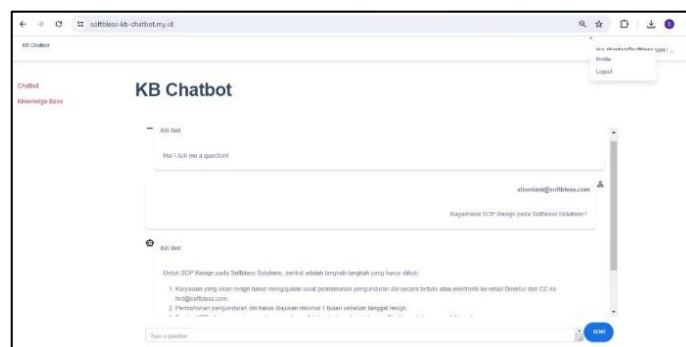
#### 1. Login View



The image shows a web browser window displaying a 'Login' page. The page has a sidebar with 'KB Chatbot' and 'Knowledge Base' links. The main content area is titled 'Login' and contains a form with fields for 'Username' and 'Password', and a 'Login' button.

Figure 10. Login View

#### 2. Bot View



The image shows a web browser window displaying the 'KB Chatbot' interface. The sidebar includes 'Chatbot' and 'Knowledge Base' links. The main content area shows a chat window with a message from 'albanhasan40@softbless.co' asking 'Bagaimana SOP design pada Softbless Solusistem?'. Below the chat window is a text input field and a 'Send' button. A sidebar on the right contains a 'Profile' button and a 'Logout' button.

Figure 11. Bot View

### 3. Knowledge Base View

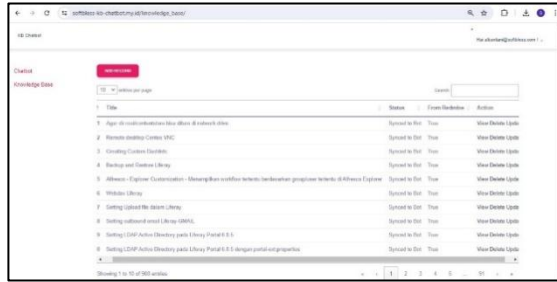


Figure 12. Knowledge List

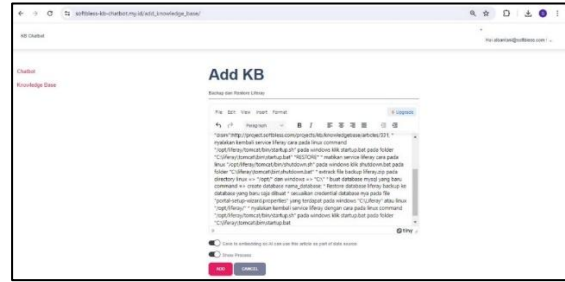


Figure 13. Add Articles

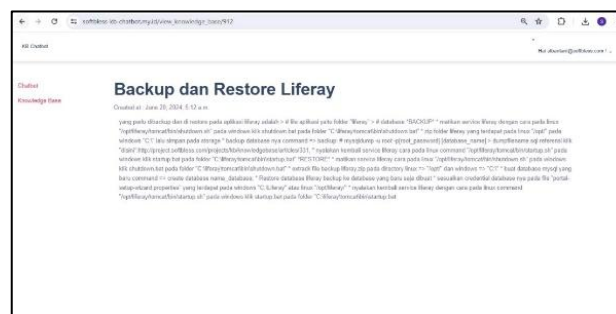


Figure 14. View Article Details

### 4. Configuration View

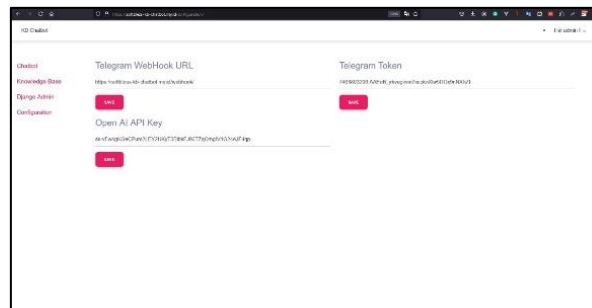


Figure 15. Configuration View

### 5. Profile User View

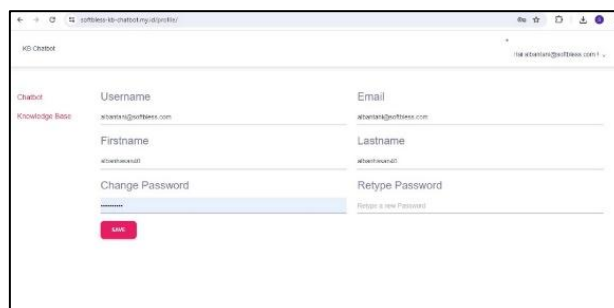


Figure 16. Profile User View



## b. Implementation RAG

### 1. Stages of Creating a Collection

A *collection* is a place or in other words a *database* that stores an *object document* that contains *text embedding*. At this stage, it will create a *collection* as an *embedding database* for the *chatbot*.

```
#Get ChromaDB Client
def get_chroma_db_client():
    client = chromadb.PersistentClient(
        path=DATABASE_PATH,
        settings=Settings(),
        tenant=DEFAULT_TENANT,
        database=DEFAULT_DATABASE,
    )

    return client

#Create or get collection
def get_collection(open_ai_key):
    client = get_chroma_db_client()
    collection = client.get_or_create_collection(COLLECTION_NAME, embedding_function=getEmbeddingFunction(open_ai_key))
    return collection

def getEmbeddingFunction(open_ai_key):
    openai_ef = CustomOpenAIEmbeddings(
        openai_api_key=open_ai_key,
    )

    return openai_ef
```

Figure 17. Source Code creates a collection

### 2. Stages of data preparation

At this stage, *cleansing* is carried out on the text or content of the article to remove *unused HTML tags*. After that in figure 19, the content will be cut into small pieces.

```
def clean_text(text):
    CLEANR = re.compile('<.*?>')
    cleaned_text = re.sub(CLEANR, '', text)
    cleaned_text = re.sub(r"[\r\n]", "", cleaned_text)
    return cleaned_text
```

Figure 18. Source Code Clean Text

```
def split_text(documents: list[Document]):
    text_splitter = RecursiveCharacterTextSplitter(
        length_function=len,
        add_start_index=True,
    )
    chunks = text_splitter.split_documents(documents)
    print(f"Split {len(documents)} documents into {len(chunks)} chunks.")

    return chunks
```

Figure 19. Source Code Split Text

```
def create_document(content, id_article):
    doc = Document(page_content=content, metadata={"id_article": id_article})

    return doc
```

Figure 20. Source Code Create Document Object

### 3. Steps To Save The Document

Once cut, the content will be converted into a document form so that it can be stored in the embedding database.

```
def add_collection(ids, documents, open_ai_key):  
    collection = get_collection(open_ai_key)  
    collection.add(ids=ids, documents=documents)  
  
    return collection
```

Figure 21. Source Code add document

#### 4. Stages of Save Retrieval

Figure 22 is a function to search for embedding that is relevant to the text or question given.

```
# Step 4: Implement Retrieval System Using LangChain and ChromaDB  
def retrieve_documents(query, open_ai_key):  
    retriever = Chroma(collection_name=COLLECTION_NAME, embedding_function=getEmbeddingFunction(open_ai_key), persist_directory=DATABASE_PATH)  
    results = retriever.similarity_search_with_relevance_scores(query, k=1)  
    return results
```

Figure 22. Is A Function To Search For Embedding That Is Relevant To The Text Or Question Given.

#### 5. ChatGPT Prompt

After finding the context of the existing article content to the given question, it will send prompt to ChatGPT to process the answers to make them easier for users to read.

```
PROMPT_CONTEXT_TEMPLATE = """  
Question Given: {question}  
  
Translate the context below using the same language as question given.  
  
Context: {context}  
  
Answer the question using translated context.  
  
Format the answer in html way  
---  
  
PROMPT_TEMPLATE = """  
Question : {question}  
  
Answer the question based only on the following context  
and format the context so that user can easily read.  
translate the context below using the same language as question given.  
  
Context : {context}  
  
---  
  
Answer the question using the context given  
  
Generate the answer in html way  
---  
  
PROMPT_QUESTION_TEMPLATE = """  
Give the result of translate the question below, if the question  
below using english then translate first to indonesian and if the question  
is using bahasa indonesia translate the question into english  
  
{question}  
---
```

Figure 23. Prompt Template

#### Test Result

The test results of the research were carried out by conducting a User Acceptance Test on 5 respondents and providing 9 questions related to the functionality of the system created.

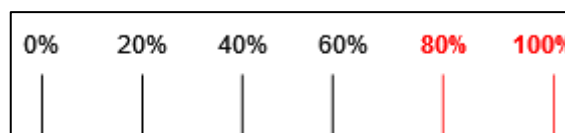
**Table 1. Total Number of Respondent Results**

No	Statement	1	2	3	4	5	Total	Total scores	Percentage
1	Do you agree that chatbots respond quickly enough?				2	3	5	23	92%
2	Do you agree that the chatbot's response to the question given is interactive?				1	4	5	24	96%
3	Do you agree with the answers that have been given by the chatbot according to the questions asked				1	4	5	24	96%
4	Do you agree that the chatbot has been able to understand questions in Indonesian and English well?				2	3	5	23	92%
5	Overall, does the chatbot perform well?				2	3	5	23	92%
6	Is the use of chatbot and telegram bot apps easy to use?					5	5	25	100%
7	Do you think that chatbot applications and telegram bots have exceeded the search feature in finding the Knowledge base on the current redmine application?				1	4	5	24	96%
8	Can the use of chatbot and telegram bot applications improve your performance?				1	4	5	24	96%
9	Do you agree that chatbot applications and telegram bots can search for the answers needed quickly and accurately?				1	4	5	24	96%

The following is the overall calculation of the UAT results from 5 respondents:

$$(92\% + 96\% + 96\% + 92\% + 92\% + 100\% + 96\% + 96\% + 96\%)/9 = 95.1\%$$

From the results of the calculation above, the average score is 95.1% which can be interpreted as a high score. This can be seen in the following image :



**Figure 18. UAT Scale**

## CONCLUSION

Based on the research presented, it can be concluded that the application of GPT in chatbots was successfully implemented using existing knowledge base data sources through the Rapid Application Development (RAD) method. The User Acceptance Test, conducted with five respondents, demonstrated a high usability level with a score of 95.1%. For future research, it is suggested to expand the testing to a larger and more diverse group of users to further validate the system's effectiveness and explore the integration of additional AI capabilities to enhance chatbot responsiveness and multilingual support.

## REFERENCES

- Ali, A., Nor, R. N. H., Abdullah, R., & Murad, M. A. A. (2016). Developing Conceptual Governance Model For Collaborative Knowledge Management System In Public Sector Organisations. *Journal Of Information And Communication Technology*, 15(2). <https://doi.org/10.32890/Jict.15.2.2016.6702>
- Anwar, M., Asmawati, A., Putri, R. R., Anantyo, R., Triyanto, A., Paramarta, V., Magister, M. P., & Rumah, M. (2024). Penerapan SIMRS Dengan Knowledge Management System: Solusi Peningkatan Mutu Pelayanan Kesehatan. *Jurnal Kesehatan Tradisional*, 2(1).
- Buntoro, G. A., Astuti, I. P., Widhianingrum, W., Arifin, R., Winangun, K., & Selamat, A. (2023). Knowledge Management System For Handcrafted Reog Ponorogo Products. *Electronic Journal Of Knowledge Management*, 21(2). <https://doi.org/10.34190/EJKM.21.2.3026>
- Cerchione, R., & Esposito, E. (2017). Using Knowledge Management Systems: A Taxonomy Of SME Strategies. *International Journal Of Information Management*, 37(1). <https://doi.org/10.1016/J.Ijinfomgt.2016.10.007>
- Choi, H. J., Ahn, J. C., Jung, S. H., & Kim, J. H. (2020). Communities Of Practice And Knowledge Management Systems: Effects On Knowledge Management Activities And Innovation Performance. *Knowledge Management Research And Practice*, 18(1). <https://doi.org/10.1080/14778238.2019.1598578>
- Dei, D. G. J. (2021). Perspectives Of Knowledge Management Systems Implementation. *Library Philosophy And Practice*, 2021.
- Gartner. (2024). *Challenges In Knowledge Base Management*. <https://www.gartner.com/>
- Haass, O., Akhavan, P., Miao, Y., Soltani, M., Jan, T., & Azizi, N. (2023). Organizational Citizenship Behaviour On Organizational Performance: A Knowledge-Based Organization. *Knowledge Management And E-Learning*, 15(1). <https://doi.org/10.34105/J.Kmel.2023.15.005>
- Mckinsey & Company. (2023). *The State Of Knowledge Management In 2023*. <https://www.mckinsey.com/>
- Montoya-Quintero, D. M., Bermudez-Ríos, L. F., & Cogollo-Flórez, J. M. (2022). Model For Integrating Knowledge Management System And Quality Management System In Industry 4.0. *Quality - Access To Success*, 23(189). <https://doi.org/10.47750/QAS/23.189.03>
- Natek, S., & Lesjak, D. (2021). Knowledge Management Systems And Tacit Knowledge. *International Journal Of Innovation And Learning*, 29(2). <https://doi.org/10.1504/IJIL.2021.112994>

- Nova, N. A., González, R. A., Beltrán, L. C., & Nieto, C. E. (2023). A Knowledge Management System For Sharing Knowledge About Cultural Heritage Projects. *Journal Of Cultural Heritage*, 63. <https://doi.org/10.1016/J.Culher.2023.07.013>
- Radeva, I., Popchev, I., Doukovska, L., & Dimitrova, M. (2024). Web Application For Retrieval-Augmented Generation: Implementation And Testing. *Electronics (Switzerland)*, 13(7). <https://doi.org/10.3390/Electronics13071361>
- Topsakal, O., & Akinci, T. C. (2023). Creating Large Language Model Applications Utilizing Langchain: A Primer On Developing LLM Apps Fast. *International Conference On Applied Engineering And Natural Sciences*, 1(1). <https://doi.org/10.59287/Icaens.1127>
- Zanker, M., & Bureš, V. (2022). Knowledge Management As A Domain, System Dynamics As A Methodology. *Systems*, 10(3). <https://doi.org/10.3390/Systems10030082>