

# **Generative AI-Based Chatbot for Customer IT Support Automation**

**Group ID: R25-036**



## **Project Final Report**

**Gamage U.R - IT21807480**

**Supervisor: Prof. Nuwan Kodagoda**  
**Co-supervisor: Dr. Lakmini Abeywardhana**

**B.Sc. (Hons) Degree in Information Technology**  
**Specialization in Software Engineering**

**Sri Lanka Institute of Information Technology**  
**Faculty of Computing Department**  
**of Software Engineering**

**August - 2025**

# Generative AI-Based Chatbot for Customer IT Support Automation

Group ID: R25-036

Final Report

B.Sc. (Hons) Degree in Information Technology  
Specialization in Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

August - 2025

Supervisor: Prof. Nuwan Kodagoda

Co-supervisor: Dr. Lakmini Abeywardhana

Student: Gamage U.R - IT21807480

## Declaration

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Gamage U.R	IT21807480	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

.....  
Signature of the Supervisor:

(Prof. Nuwan Kodagoda)

Date:

## Abstract

This research project focuses on developing a generative AI - based chatbot for LOLC Company, designed to streamline and enhance their IT support systems. The chatbot integrates a range of features that enable employees to efficiently resolve IT-related queries, gain insights into HR policies, and access business-related information. By leveraging an advanced knowledge base, the chatbot provides precise and context-specific responses to user prompts, significantly improving operational efficiency and decision-making within the organization.

A central component of the system is its intelligent knowledge base, which allows dynamic updates through a user-friendly interface. This feature empowers staff to upload new documents, update existing policies, and refine content, ensuring that responses remain accurate and relevant over time. The knowledge base also enables personalized answers by learning from user interactions and retaining episodic memory of previous queries. This ensures a tailored experience for each user while continuously improving response accuracy based on feedback.

In addition to text-based support, the chatbot features an immersive 3D avatar equipped with voice-to-text and text-to-voice capabilities, enabling seamless interaction with employees. The system addresses knowledge gaps by auto-generating answers and providing guidance for completing forms, thus reducing manual effort and improving user efficiency. An agent-based automation system further enhances productivity by executing repetitive tasks based on instructions derived from the knowledge base.

This innovative solution is built to adapt dynamically to changing organizational needs while maintaining scalability and reliability. By combining personalized support, streamlined processes, and dynamic knowledge management, the chatbot sets a new benchmark for intelligent enterprise support systems, driving efficiency and improving the overall employee experience at LOLC.

## Table of Contents

<b>Declaration .....</b>	<b>3</b>
<b>Abstract .....</b>	<b>4</b>
<b>Table of Figures .....</b>	<b>6</b>
<b>List of Abbreviations .....</b>	<b>7</b>
<b>1. Introduction .....</b>	<b>8</b>
<b>1.1 Background.....</b>	<b>8</b>
<b>1.2 Literature survey .....</b>	<b>9</b>
<b>1.3 Research gap.....</b>	<b>11</b>
<b>1.4 Research Problem.....</b>	<b>13</b>
<b>1.5. Research Objectives.....</b>	<b>15</b>
1.5.1 main objective .....	15
1.5.2 specific objectives.....	15
<b>2. Methodology.....</b>	<b>17</b>
<b>2.1 System overview .....</b>	<b>17</b>
Requirement gathering .....	19
Data Ingestion & Construction of Knowledge Base.....	19
Vector Retrieval Layer.....	20
Re-Ranking Mechanism .....	20
Conversational episodic memory .....	21
Caching level .....	22
Policy Knowledge Graph.....	22
Chaining-of-Thought (CoT) Prompting.....	23
Safety & Guardrails.....	23
The API Layer.....	24
Frontend Interaction Layer .....	24
Evaluation Framework .....	25
Technology & Technique Justification .....	25
Efficiency Measures .....	26
Risks & Mitigations .....	27
<b>2.2 Commercialization aspect of the project.....</b>	<b>31</b>
Motivation Market and (limiting users) & Users .....	31
Value Proposition .....	31
Market Research and Demand .....	31
Revenue Model .....	31
The development and deployment.....	32
Growth Potential and Scalability.....	32
<b>2.3 Testing and Evaluation .....</b>	<b>33</b>
Logging and error tracking .....	33
Error Handling/Monitoring.....	33
Chainlit Functional Testing.....	33
Dataset Optimization .....	34
Caching Layer Testing.....	34
Iterative Improvements.....	34
Implementation process .....	37
Security of the backend .....	40
Project requirements.....	40

<b>3 Result &amp; Discussions.....</b>	<b>43</b>
<b>3.1 Results .....</b>	<b>43</b>
<b>3.2 Research Findings .....</b>	<b>46</b>
<b>3.3 Discussion .....</b>	<b>47</b>
<b>3.4 Summary.....</b>	<b>49</b>
<b>4. Gantt Chart.....</b>	<b>50</b>
<b>Conclusion .....</b>	<b>51</b>
<b>Reference.....</b>	<b>53</b>

## Table of Figures

Figure 1 - Architecture of the system.....	9
Figure 2 - ProxyLLM Background process.....	10
Figure 3 - Knowledgebase Architecture .....	17
Figure 4 - Internal process of generating answers .....	18
Figure 5 - data processing and insertion process .....	20
Figure 6 - Reranking process.....	21
Figure 7 - chat history retrieval with MongoDB indexing.....	22
Figure 8 - how knowledge graphs are created .....	23
Figure 9 -Guardrail validations .....	24
Figure 10 -Data set .....	30
Figure 11 -Test cases for RAGAS Metrics .....	35
Figure 12 - Final evaluation result JSON .....	36
Figure 13 - visualize monitoring to detect errors in each phase .....	37
Figure 14 - docker file .....	39
Figure 15 - query results.....	45
Figure 16 - Radar map .....	45
Figure 17 - Gantt chart .....	50

## List of Abbreviations

RAG - Retrieval-Augmented Generation

LLM - Large Language Model

Gen AI - Generative Artificial Intelligence

BERT - Bidirectional Encoder Representations from Transformers

NLP - Natural Language Processing

TTS - Text-to-Speech

API - Application Programming Interface

LOLC - Lanka ORIX Leasing Company

AZ - Azure (Microsoft cloud platform)

UI - User Interface

CLI - Command-Line Interface

HTTPS - Hypertext Transfer Protocol Secure

TLS - Transport Layer Security

# 1. Introduction

## 1.1 Background

In the modern business environment that continues to develop rapidly in terms of technologies, the generative AI technology rapidly revolutionizes how businesses should work. Routine tasks that had to be manually completed by employees are becoming automated, using AI-based solutions. This has necessitated the need to upgrade current systems that are in use especially in large multinational companies so as to keep in touch with latest technological trends.

In this regard, we have joined hands with LOLC companies in order to promote their existing IT services. Namely, we will modernize their present IT ticketing system to add the latest generative AI features to this system. We aim to enhance productivity and effectiveness of the employees through the employment of a chatbot solution with an internal knowledge base.

The ultimate aim of this project is to develop a chatbot that can give response to the employee related to HR policies, IT support, and any other business processes. This solution will automate the internal communication, save time that is used in making routine inquiries as well as improve on the productivity of the employees. With the help of generative AI, we will develop a personalized ticketing process that can be faster and more efficient, where less human intervention is required to address the ticket and where the process of resolving a ticket does not take an excessive amount of time.

The modern systems of manual ticketing are likely to involve a lot of efforts dedicated to human personnel, can be subject to delays, and cannot offer an individual approach to a problem resolution. Employees are often confronted with difficulties in getting the right answers promptly thus affecting their efficiency. It will be achieved through the implementation of a knowledge base designed specifically around the internal company information in order to ensure that employees can refer to the information immediately when they will need them. Moreover, the system will enable people that are not tech-savvy to add and correct the information, thus keeping the platform current and updated.

The proposed research project is an attempt to design a state-of-the-art AI-based tool that can be used to resolve these inefficiencies to an extent as well as help financial organisations such as LOLC deploy AI in their current framework. Such integration can help the firms optimize operations and employee satisfaction and stay in the competitive landscape of the future driven by AI.

## 1.2 Literature survey

Academic researchers have been making comprehensive investigations in the previous decades regarding the phenomena of big language models (LLMs) and chatbots-related technologies. These initiatives have led to the creation of transformer-based LLMs that have led to the breakthroughs in natural language understanding and generation. When it comes to areas such as IT support and HR policies, a lot of research efforts have been put to enhance purposefulness and dependability of automated response.

Retrieval-Augmented Generation (RAG) [1] integration with LLMs has become a topic with a lot of interest due to increasing accuracy and contextual relevance through the combination of generative capabilities with real-time retrieval. Han et al. (2024) point out its utility in the area of automating systematic literature reviews (SLRs) [6], demonstrating how RAG makes easier such steps as literature search, screening, data extraction, and synthesis. The materialistic attitude of RAG goes against problems associated with hallucinations and makes sure that outputs in domains that require precise information, such as medicine and law, are accurate.

IT support, [8] point to the strength of RAG systems at ticket resolution of incidences. Their solution provides an integration of retrieval and generative components overcoming such issues as the issue of domain-specificity of retrieval models, or the limitations of smaller models. This methodology augments the backing, as the responses are based on validated documentation, and hence the improved trust and accuracy. Further, [13] show an innovative work integrating RAG and knowledge graphs in customer service, where structural and relationship data is also maintained to improve retrieval accuracy and cut down the resolution time by 28.6%.

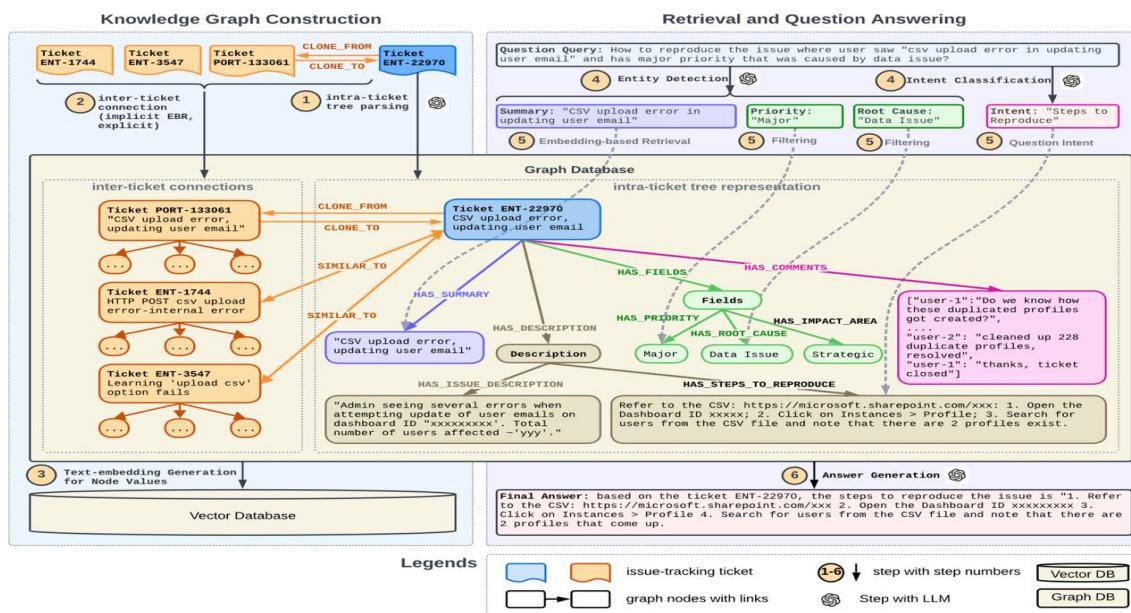


Figure 1 - Architecture of the system

Human-centered application attention is discussed by ProxyLLM in a work by Jo and Seo (2024), which translates unconscionable customer-to-text speech to neutral tones, using the text-to-style transfer framework. This technology combats the emotional burden on the agents and increases efficiency in customer service settings. The article of Shih-Ien Heng 2023, The Future of HR Chatbots, dwells upon the progress in vector search and the shift towards versatile LLMs, such as GPT-4. These enhancements stream chatbot responses to the organizational policies and proves how RAG systems can be flexible in an enterprise-related application.

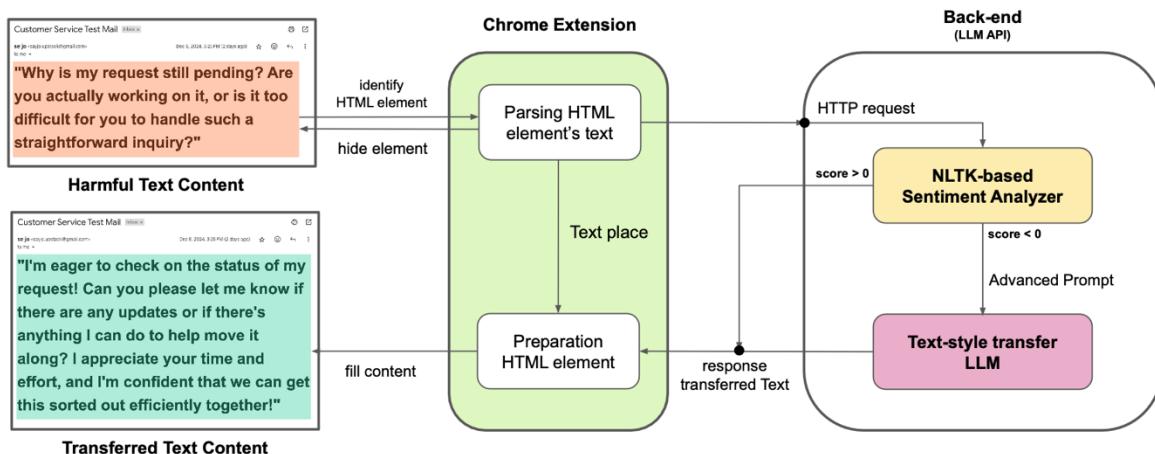


Figure 2 - ProxyLLM Background process

Although these advancements have happened, there are still a number of gaps. Existing systems have been known to not deal well with domain-specific issue and multimodal data processing and querying [12] issues as well as the efficient querying of complex queries [4]. Coupled with the fact that retrievers rely heavily on the availability is the problem of flexibility that can be imposed when the environment is highly volatile. Future directions could touch on the combination of adaptive retrievers with multimodal signals, the use of finetuned domain-specific LLMs, and the ability of models entering real-time learning. Moreover, the interaction between retrieval and generation could be enhanced by means of novel prompt engineering and teachable learning, providing more robust and context-sensitive prediction.

All in all, the deployment of RAG and LLMs has enhanced automation in the field of literature review, IT support system, HR and customer care. These systems can show how it can be revolutionary in the knowledge industries in providing accurate, reliable, and efficient solution.

## 1.3 Research gap

As per the literature survey, I have narrowed down to most important future implementations and research gaps regarding the knowledge bases. The details of persistence and efficacy of memory, efficiency of various tokens and mechanisms of successful retrievals are shared in most works. A common theme is that the majority of Retrieval-Augmented Generation (RAG) systems require a static document upload and are likely to require significant manual intervention to update the content. This shows that there is a necessity to utilise an automated system to dynamically populate existing vector databases with new contextual data and able to identify and update or replace similar existing data. An optimal data management and access would be achieved by such a system due to the high technology used.

## Research Gap

SYSTEMS	DYNAMIC RAG	EPISODIC MEMORY	NLP BASED DOCUMENT ANALYSIS	AGENT INSTRUCTIONS
RESEARCH PAPER [1]	✗	✗	✓	✗
RESEARCH PAPER [2]	✗	✗	✗	✗
RESEARCH PAPER [3]	✓	✗	✗	✓
PROXYLLM	✗	✗	✓	✗
OUR SYSTEM	✓	✓	✓	✓

Figure 3 - Research gap

Another important issue is that large language models (LLMs) tend to predict hallucinated answers to some user queries. Such answers are in most cases irrelevant to the questions asked and do not address user expectations, hence user dissatisfaction. They have previously studied AI chatbots mainly as a method of customer service and have not extensively been used yet in the same industry to assist the internal employee.

In addition, it is often necessary to convert contextual documents, like Word documents and PDFs, into an organized structured format. It is however inefficient in identifying the context of the content accurately and has an inadequate mechanism of indexing the information in the databases leaving less response time. The previous approaches do not smoothly upload this structured data to the vector databases or query optimization of indexes. In a similar vein, most chatbot solutions require the direct integration with OpenAI API and prompt engineering resulting in very basic responses that do not fulfill the requirements of informed Existing systems with large language models (LLMs) tend to provide all of the conversation text as a

prompt to the model as a part of response generation, which can be highly inefficient by wasting tokens and driving up model cost on every interaction. Although natural language processing has taken considerable strides, the major inefficiency has been on how to develop effective strategies of memory management in LLMs. In particular, prior systems did not have any mechanism of dynamically optimizing the context selection to minimize redundancy whilst not compromising accuracy of the responses. What is also needed is the approaches that involve user feedback to measure satisfaction, determine successful and unsuccessful interactions and customize responses to them. Another underlying aspect not dealt with is the identification of repeating queries and repeated queries to reduce redundant queries and enhance query recipe retrieval. Closing these gaps is critical to the cost-effectiveness and contextual sensitivity of the LLM-based systems.

## 1.4 Research Problem

With the identified research gaps and the weaknesses of the existing Retrieval-Augmented Generation (RAG) systems and large language models (LLMs) in enterprises, the research problem would be to work around the weaknesses of existing systems. Current systems are based on static uploads of documents, which require manual updating of the vector databases and thus, reduces the level of efficiency and scalability. Moreover, LLMs tend to produce hallucinogenic responses that jeopardizes user trust and satisfaction, especially in important spheres such as internal employee support to financial institutions. There are inefficiencies in memory operations like redundant token utilization and the high cost of operation that augment our problems.

Existing systems also do not have a dynamic context selection optimization capability as well as an efficient way of handling repeat queries and user feedback to create personalization and system refinement. Employees tend to have monotonous problems, and this results in their repetitive ticketing which causes more waste of time and hence needs to be attended by support staff. A support chatbot that uses an updated knowledge base has the capability to crunch this inefficiency by automatically finding relevant context and delivering steps or suggestions to answer user queries, saving much time.

In practice, the action of uploading contextual files, e.g. Word and PDF documents, may demand their conversion into the JSON format eventually. Using Natural Language Processing (NLP) such documents can be analyzed and their categories determined and their content dynamically structured. Current techniques however fail to provide the means by which such structured documents can be read into a vector database with the retrieval indexing being optimized. This gap constrains the possibility of having meaningful and contextually oriented answers to user questions. This research is geared to overcome this challenge by the integration of similarity-based retrievals, which ensures the organization of the documents effectively and enhancement in quality of the query results.

To further enhance the system the use of episodic memory can be introduced into the design of the chatbot to allow him to remember individual interactions with users and respond accordingly to the individual, which is a major weakness of current RAG-based systems. Moreover, certain queries generated by users might necessitate multimodal context- e.g. a combination of IT support and HR-related responses to collate a detailed response. The proposed system will serve this purpose by combining the relevant pieces of information pertaining to different documents and generating an appropriate response.

Last but not least, the knowledge base component will be able to help a support agent by providing them with actionable instructions and context-specific data and facilitate their tasks. The research objective is to build on a novel framework that will facilitate dynamic updating of knowledge bases, minimize hallucinations, maximization of token efficiency,

multimodal context and provide accurate, personalized, and context-relevant responses to enterprise requirements.

## 1.5. Research Objectives

### 1.5.1 main objective

The major goal is to create a dynamic Retrieval-Augmented Generation (RAG) system with the integration of the API of OpenAI to simplify IT and HR support of LOLC company. The system will use internal data to give specific and situation-specific advice and stepwise solution to IT support problems. The system will provide customized responses based on the historical interactions of the employees by maximizing the efficiency of the tokens and memory using episodic memory and apply a cost-efficient knowledge base. It will also employ the use of several documents to produce composite answers, so that the answers to the questions asked by the users are accurate and correct. The system will also use several cloud technologies to facilitate its implementation to make it scalable, reliable and accessible. This strategy will focus on increasing accuracy, decreasing response time and the general satisfaction of the employees by making the process of solving problems easier and more manual labor intensive.

### 1.5.2 specific objectives

#### **Knowledge Base Development**

- create the knowledge base server by based on a microservice architecture that helps the server to act as an independent component that is accessed through APIs.
- Make use of timely engineering processes to give accurate and context based responses to employee queries.
- Create easily understandable step-by-step guidelines and materials to be used by agents to effectively perform automation activities.

#### **Developing a RAG System**

- Develop and deploy a Retrieval-Augmented Generation (RAG) system to respond to user queries using the existing context of a query to the vector database.
- Search mechanisms are used to get answers to several documents and sources to specialize the knowledge base on IT support and HR policies.
- Use a document identification and classification method that is based on NLP to translate content into structured JSON before embedding.

## **Dynamic Context Update in Vector DB**

- Build a dynamic system to be able to add new documents or text content through the UI and automatically embedding and storing them to the database.
- 
- Similar existing contexts in the vector database will automatically be detected by the system and modified to reflect the relevant, newly added context.

## **Adopting Episodic Memory of the RAG System.**

- Harness episodic memory to use prior chat experience and give more contextually valid answers.
- Integrate prior experiences with current context, enhancing accuracy and memory efficiency to avoid the wastage of unnecessary tokens and to maximize the performance of the system.
- The response system was built on the input of the users to develop a pipeline to determine what works and what does not work. Response successes were stored in the database to enhance the provision of contextual responses in the system.

## **The deployment of Systems using Cloud Technologies.**

- Engage the use of the Azure cloud technologies to create a reliable, accessible, and scalable system.
- Support the knowledge base with high availability and distributed performance to enable the knowledge base to satisfy the needs of a growing enterprise.

## **Implementing automated filetype identifier and converter**

- The data form can change when uploading the documents to update the knowledgebase. To manage this, there is need to have an automated file identifier and converter to extract the data of various types of documents.

## 2. Methodology

### 2.1 System overview

The development of the knowledge base is the most important element in this Gen AI-based chatbot because it is central to producing accurate responses to users. This knowledge base has many features which are shown in the component diagram below.

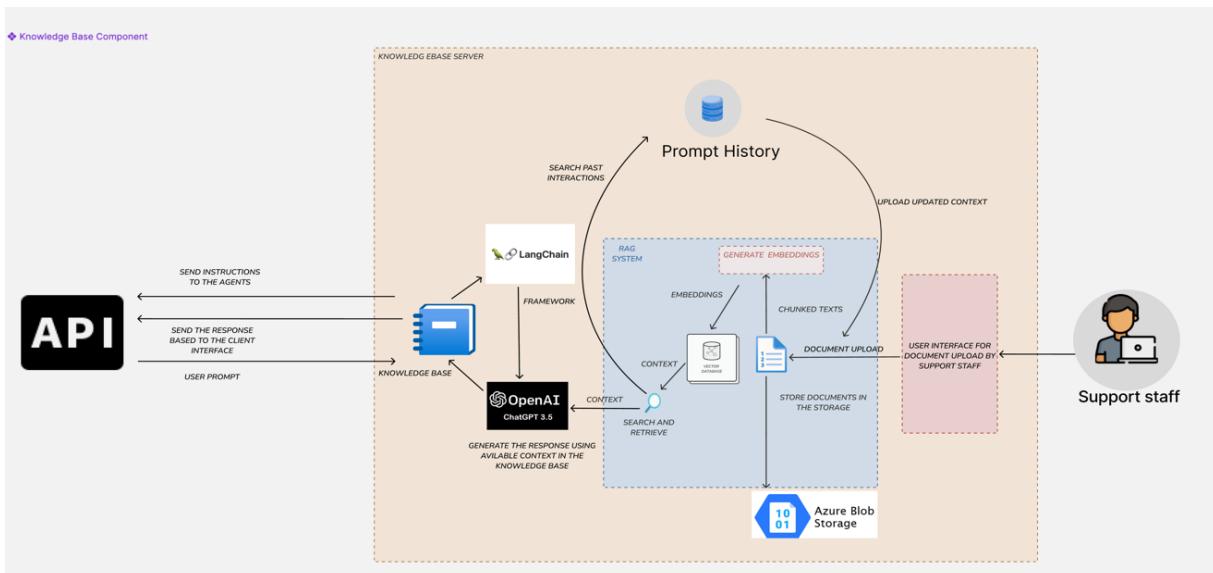


Figure 4 - Knowledgebase Architecture

The first exercise in establishment of this knowledge base will be to accumulate the necessary internal data of LOLC and to demystify the system requirements. A microservice architecture will be adopted to build the knowledge base component because it is a standalone component and can be scaled. The information gathered and data will require updating in a vector database using embedding text models to apply a powerful searching and retrieval system. This will enable the system to recover pertinent data in the database and give more precise responses to user queries by enriching the context through the assistance of large language models (LLMs).

## Knowledge base process

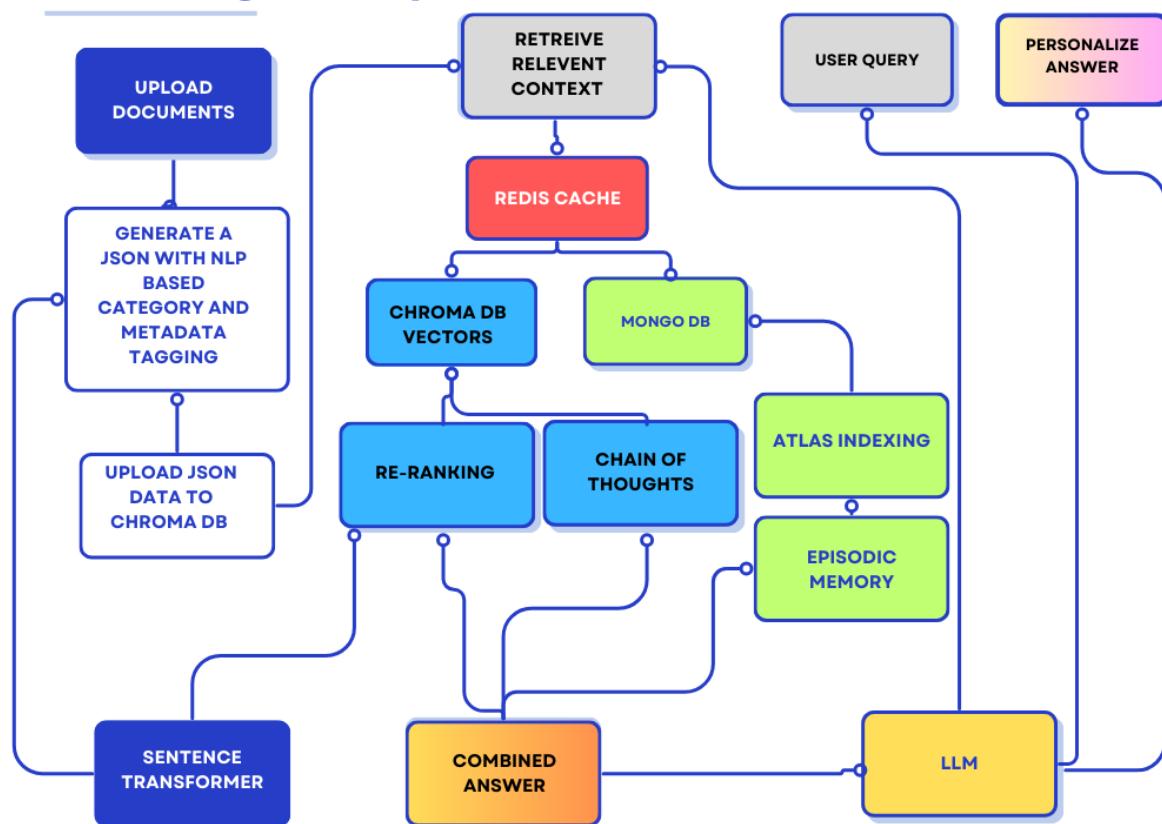


Figure 5 - Internal process of generating answers

A system will be used to dynamically update the content in order to keep the knowledge base current. This involves the addition of documents or policies and the up to date information replacing old information. To enable staff to manage these updates, a user interface (UI) will be created to provide staff with an opportunity to monitor and appropriate the knowledge base. The system will also be able to spot and substitute the old information with new information and keep the knowledge base complete and relevant. Additionally, a pipeline will be developed to save prompt history and build an episodic memory system. Such a system will store and recognize historical interactions, such as keywords, timestamps, prompts, and user feedback. This historical information will allow creating individual answers depending on the interactions of the users in the past. User feedback will be used too to determine which answers worked and which ones failed. Effective prompts and responses will get updated in the database so that the system can give correct and sophisticated responses when such queries are raised again.

The entire system will be hosted in the Microsoft Azure using Azure technologies, and the document management system will be based on a cloud environment. The APIs of OpenAI will be incorporated to make the chatbot give answers that are contextually accurate.

Moreover, the system will produce stepwise instructions to be followed by agents depending on the query made by the user and the context identified. As users submit questions using the

UI, the knowledge base shall classify query (e.g. whether query is support related) and shall pack the response accordingly using prompt engineering techniques. This will guarantee that the system will provide correct responses depending on the needs of the user.

## Requirement gathering

The requirement gathering was carried out by means of conducting the analysis in terms of the past research that was conducted over the last years. The primary data was obtained through Kaggle free available data set and close coordination with the company technical team to gain insight to the customer requirements of the application. Further details were retrieved using credible sites that deal with technical assistance. The process entailed interaction with the supervisor, co-supervisors as well as the internal research team of SLIIT besides sustained communication with the internal team of LOLC.

## Data Ingestion & Construction of Knowledge Base

The process begins by scraping IT support documents and restructuring them so that the system can effectively process and understand the content. Essentially, the documents are scanned and cleaned—all letters are converted to lowercase and broken into smaller units. While this may seem straightforward, it significantly reduces inconsistencies and makes searching far easier.

For example, if a user encounters a forgotten password issue, the system can detect it simply by analyzing the headings in the manuals. This ensures that different issues are not confused and can stand independently. Keywords like “Windows” or “login issue” are already tagged in the JSON, which helps the system quickly identify and match relevant cases without extra manual labeling.

Additionally, key information such as keywords and error codes is extracted and listed for correct data retrieval. The text is organized into overlapping chunks. Although some overlap may seem redundant, it is practical, as it preserves context without losing clarity.

Finally, all of this structured information, along with metadata, is stored in JSON format, making the knowledge base fast to train and easy to reuse. In the future, this process could be optimized further by adjusting chunk sizes according to content or by finding improved methods to handle duplicate information efficiently.

```

def extract_from_pdf(file_path):
    with pdfplumber.open(file_path) as pdf:
        for page in pdf.pages:
            text = page.extract_text()

            if not text:
                continue

            lines = text.split("\n")
            for line in lines:
                if re.match(r"^\d+[\.\;]?\s?", line): # Matches lines like "1. Forgotten Password"
                    if current_issue:
                        issue_text = " ".join(current_content).strip()
                        print(f"Processing Issue: {current_issue}") # Debugging output
                        print(f"Issue Content: {issue_text[:100]}...") # Print first 100 chars

                    issues_data.append({
                        "issue": current_issue,
                        "content": issue_text,
                        "related_category": predict_category(issue_text),
                        "keywords": extract_keywords(issue_text),
                        "issue_type": predict_issue_type(issue_text),
                        "operating_system": predict_os(issue_text),
                        "error_code": extract_error_code(issue_text),
                        "date_extracted": datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                    })

                    current_issue = line.strip()
                    current_content = []

```

Figure 6 - data processing and insertion process

## Vector Retrieval Layer

The vector retrieval layer is responsible for finding the most relevant information from the knowledge base. All content chunks are stored with unique, deterministic identifiers, making updates and replacements easy. When a query is run, the system identifies the most similar chunk using vector similarity, followed by a re-ranking step to improve accuracy. This two-step process ensures users receive precise answers within reasonable response times. Currently, keyword-based filtering is not combined with vector retrieval, but added metadata filters in the dataset have improved precision of the generating answers.

## Re-Ranking Mechanism

The re-ranking mechanism will improve search results prior to using them in the language model. By not altering the position of the best suited candidates large in the list, it will enhance the relevance of the initial results, this will lessen the possibility of irrelevant and misinformative answers. The system has been made to accommodate failures nicely where the original order will be followed in default. In the future, swapping the present model of scoring based on heuristics with some more powerful model would be able to make the results accurate without having a drastic effect on the response time.

```

reranker = CrossEncoder('cross-encoder/ms-marco-MiniLM-L-6-v2', max_length=512)

async def re_rank_documents(query, documents):
    """
    Re-rank documents based on their relevance to the query.

    Args:
        query (str): The user query
        documents (list): List of document strings to re-rank

    Returns:
        dict: Dictionary mapping document IDs to relevance scores
    """
    print(documents)
    try:
        # Prepare document pairs for scoring
        document_pairs = []
        for i, doc in enumerate(documents):
            document_pairs.append([query, doc])

        # Score the documents using the cross-encoder model
        def score_docs():
            return reranker.predict(document_pairs)

        loop = asyncio.get_event_loop()
        scores = await loop.run_in_executor(None, score_docs)

        # Create a dictionary mapping document IDs to scores
        result = {}
    
```

Figure 7 - Reranking process

## Conversational episodic memory

Conversational memory tracks past interactions, including problems, causes, and solutions, to provide consistent and personalized responses. Using multiple indexing techniques, the system searches the NoSQL database to find the most relevant past interactions, even for differently worded queries. This helps reuse proven troubleshooting guidance. Currently, sensitive information is not automatically filtered.

```

# Get chat history endpoint
@app.get("/history/{user_id}")
async def get_chat_history(user_id: str, limit: int = 10):
    try:
        history = list(chat_collection.aggregate([
            {
                "$search": {
                    "index": "default",
                    "queryString": {
                        "query": user_id,
                        "defaultPath": "user_id"
                    }
                }
            },
            {"$sort": {"timestamp": -1}}, # Sort by newest chats
            {"$limit": limit},
            {"$project": {"user_query": 1, "ai_responses": 1, "analysis": 1, "timestamp": 1, "_id": 0}
        ]))

        return {"history": history}
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error retrieving chat history: {str(e)}")

```

Figure 8 - chat history retrieval with MongoDB indexing

## Caching level

The caching layer enhances performance by storing query results, and frequently accessed information, e.g. policies or common troubleshoot steps, in a cache to be reused within a given time frame (24 hrs.). When starting up, the system may pre-cache historical data into the cache in order to accelerate the first responses. The system also has reliability because even when the cache is unavailable, it manages and still runs. Today the cache is queried on a per-query basis, although future revisions may cluster similar queries to offer more intelligent and responsive queries.

## Policy Knowledge Graph

The system uses a knowledge graph to support generating answers to user questions. If a user asks about a specific issue, such as a hardware problem, the system looks for matching policies in the graph and interprets what they mean. It then combines this information with the generated answer to provide relevant guidance.

The graph covers HR policies like access control, onboarding, remote work, and security. It only loads when needed, so database issues won't stop the system from running. Currently, relationships between policies are not weighted, and multi-step reasoning across connected policies is not implemented.

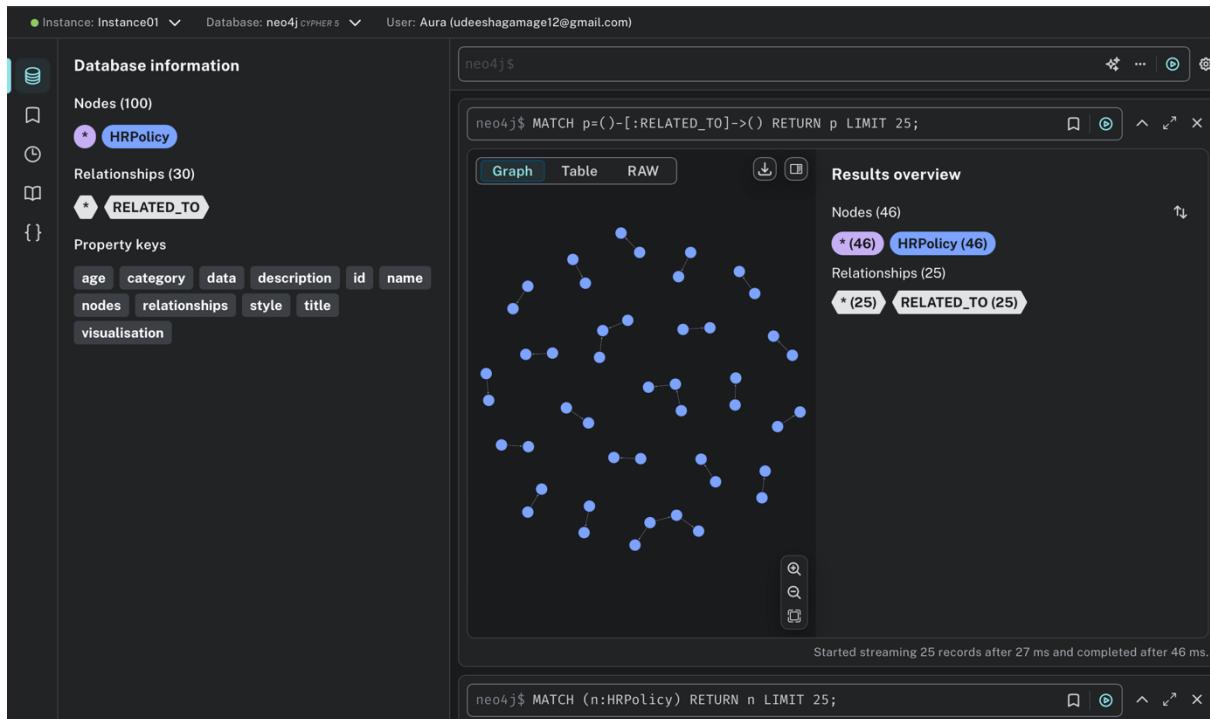


Figure 9 - how knowledge graphs are created

## Chaining-of-Thought (CoT) Prompting

Chain-of-Thought prompts breaks AI-thinking into intelligible, accountable steps. These answers are separated into parts like problem analysis, the possible causes of the problem, steps, and solutions, and prevention tips. This will provide a regular structure where the answers can be read more easily or be verified further on. The solution is consistent with the ITIL incident management lifecycle that serves to take the AI through steps of detecting the problem to offering preventive recommendations. Although it is useful to enhance simplicity, the process may get more workload when using longer prompts, but this problem is addressed by compositing only the highly pertinent information on each prompt.

## Safety & Guardrails

The system will provide numerous steps to safety to avoid the case of misusing the system. Deterministic filters on high-risk actions (credential access, destructive commands) are based on rulesets, and the ambiguous cases are fed to the lightweight AI-based classifier. The approach is not to make any unnecessary blocking, and the priority is to bear in mind availability rather than excessive limitation. The current system however does not throttle query rate per user and used a predominantly pattern-based detection strategy. Further developments can be based on higher level AI checks against malicious activities and reliance on similarity tests.

```

class GuardrailViolation(Enum):
    INAPPROPRIATE_CONTENT = "inappropriate_content"
    SECURITY_THREAT = "security_threat"
    PERSONAL_INFO_REQUEST = "personal_info_request"
    OUT_OF_SCOPE = "out_of_scope"
    MALICIOUS_INJECTION = "malicious_injection"
    EXCESSIVE_LENGTH = "excessive_length"      You, 2 months ago • latest implementation

You, 2 months ago | 1 author (You)
class ITSupportGuardrails:
    def __init__(self):
        self.max_query_length = 1000
        self.blocked_patterns = [
            r'(?i)\b(hack|hacking|crack|cracking|exploit|vulnerability|breach)\b',
            r'(?i)\b(password|pwd|passwd)\s+(for|of)\s+\w+',
            r'(?i)\b(personal|private|confidential)\s+(information|data)\b',
            r'(?i)\b(salary|wage|pay|financial|budget)\b',
            r'(?i)\b(delete|remove|drop|truncate)\s+(database|db|table)\b',
            r'(?i)\b(admin|administrator|root)\s+(password|access|credentials)\b'
        ]

        self.it_keywords = [
            'computer', 'laptop', 'software', 'hardware', 'network', 'wifi', 'email',
            'printer', 'password', 'login', 'install', 'update', 'error', 'crash',
            'slow', 'freeze', 'virus', 'malware', 'backup', 'recovery', 'vpn',
            'remote', 'access', 'device', 'monitor', 'keyboard', 'mouse', 'usb',
            'server', 'database', 'application', 'browser', 'internet', 'connection'
        ]

```

Figure 10 -Guardrail validations

## The API Layer

API layer is implemented with Fast API with endpoints that are chat, HR policies, user history, uploads, Knowledgebase and databases. API enforcement is required to support all the information that will go through the API, in expected formats. During start-up, the caches and graphs and retrieval clients will be established, and other aspects of defensive programming allow the failure in any one subsystem not to bring the whole system down. The core API is stateless and is easy to scale horizontally behind load balancers. Possible add-ons are request-level tracing to record times spent on various operations, which will allow optimizing performance.

## Frontend Interaction Layer

The frontend is geared towards a simple and easy to use chat interface. One can find the HR policies using the category buttons or perform dynamic search, responses, meanwhile, combine policy directives and structured reasoning as it requires less mental effort. The existing architecture is meant to be UI-agnostic, and in that way all future frontends which are built on frameworks such as React will be able to consume the same data contracts without modification.

## Evaluation Framework

The testing framework applies a combination of normal IT situations and difficult out-of-scope- or nonsensical requests. Measures entail answer faithfulness, relevancy and the extent to which the context of the knowledge base is remembered and used. The scores and the radar charts are then plotted in the graph and a heatmap to monitor the performance changes over a period. Limitations are that there is no evaluation of the temporal drift and that no calibrated human-annotated dataset exists.

## Technology & Technique Justification

Tool/Technology	Purpose/Use
OpenAI GPT-3.5/4	Used for natural language understanding and generating responses in the chatbot based on user queries.
Chain-of-Thought Strategy	we provide the AI clear and structured prompts. The approach assists it to perceive the question appropriately When the prompts are loosely defined you end up with improper or fabricated answers.
ChromaDB	It simple tool that stores our data and is written in Python.it's a vector database which can store vector embeddings Others such as FAISS do not save data, and Weaviate takes harder work to run.
MongoDB	NoSQL database for storing and managing data such as user queries, system responses, and knowledge base documents.
Fast API	Python-based web framework used for developing and hosting the backend API for the chatbot application.
Azure	Cloud platform for hosting the application, storing data, and providing computing resources for the system.
Huggingface	Provides pre-trained AI models and tools for natural language processing tasks, integrated into the chatbot system.
Redis	To store the memory in cache for faster data retrieval
Neo4j	To store the HR policies of the company which support to answer queries

RAGAS	is used to evaluate that everything is running smoothly. It executes automated checks on a basis of standard scores.
Guardrails	Restrict the LLM to provide unrelated answers to the users queries

## Efficiency Measures

A number of optimizations were made to increase speed and lower computational cost:

- **Precomputed Embeddings:** Embedding of all text is performed offline when ingesting so as to minimize CPU load at query time.
- **Top-N Re-Ranking:** The model receives only the most pertinent results, while useless computations are avoided.
- **Redis TTL:** After 24 hours the entries are deleted and storing old data indefinitely is avoided.
- **Lazy Service Initialization:** Components such as Neo4j may be started as a need arises, thus eliminating an all-or-nothing startup failures.
- **Chunk Tuning:** Chunks of text are broken down to the most useful size in order to preserve context with minimal prompt bloat.
- **Parsed CoT storage:** the storage of only crucial reasoning parts minimizes token storage and permits analytics.

### *The simplistic baseline vs. observed impact*

The implemented system has more evident improvements over a simplified keyword search with direct responses of the direct LLM:

- **Baseline Risks:** Basic methods tend to create illusory processes, overlook policy conformity, and re-visit issues that were resolved.

System Gains:

- Episodic memory minimises redundancy and continuity.
- Guidance that is policy-conscious will lead to organizational compliance.
- Grounded retrieval increases factual accuracy.
- Steps should have a consistent format: this is more user friendly and enables easier training.

## Risks & Mitigations

Several possible risks were determined and minimized where possible:

- **Model Drift:** The changes in the language model may result in unstable responses. Mitigation is associated with correction of temperature, prompts archiving, and conducting regular regression tests.
- **Graph Sparsity:** Sparsity of relationships minimizes policy utility. Mitigation: This is in case of policies that are absent; this will include incremental pipelines and fallback paths of ingestion.
- **Cache Poisoning:** Evil queries may be written in Redis. Mitigation in the future aims at hash-based allowlists of authentication.
- **Sensitive Data Leakage:** Memory logs are not the most sensitive information that could be stored unknowingly. There is planned mitigation of regex and entropy-based pre-persistence filters.

### *Best Practices used*

In the system component development, some of the best practices and architecture styles were used to achieve efficiency, maintainability and scalability. To design the backend all database connections were created in a Singleton design pattern which ensures that there is only one and shared instance of the database connection in the application minimizing resource overhead and maintaining consistency. The Strategy pattern was used in order to manage the various business logic implementations, and the system selected one algorithm or processing strategy at a time, without being bound to the underlying core logic. On a broader scale, the entire system was developed on a microservices architecture that offers fault isolation, scalability and modularity by dividing the application into independently deployable, loosely coupled services. Such a mixture of design patterns and architecture styles promoted the flexibility and the strength of the system.

### *Knowledge on IT support process*

In our LOLC IT support system, having a clear understanding of support processes is essential to keep day-to-day operations running without interruptions. The support team must be capable of troubleshooting problems quickly, monitoring systems effectively, and reducing downtime that could impact business-critical services. For this reason, strong skills in areas such as incident management, ticket handling, escalation procedures, and proactive system monitoring are extremely important.

As part of this project, we took time to study how established technology companies manage their IT support functions. This helped us to identify best practices that could be applied to our own environment. At the same time, we carefully analyzed our existing processes to see

where efficiency could be improved. The ultimate goal was to design a cost-effective and scalable solution that could replace the older setup and meet the growing needs of LOLC. To make sure the system truly fit the organization, we held initial meetings with both management and technical staff. These discussions allowed us to identify the main requirements as well as the pitfalls that had previously been encountered during development and deployment. In addition, we continuously engaged with our supervisor and co-supervisor to review progress, gather feedback, and refine our approach. By also working closely with internal staff members, we were able to better understand the internal workflows of the company and fix issues as they appeared. Through this ongoing process of collaboration, feedback, and adjustment.

### *Knowledge on Python*

Python plays a major role in automating repetitive IT tasks, integrating tools, and supporting backend systems at LOLC. Our IT support teams use Python along with **FastAPI** to build scalable services such as automated document ingestion, log tracking servers, and API integrations with enterprise systems. Thanks to its simplicity and powerful libraries, Python allows us to quickly develop solutions that improve efficiency, reduce manual work, and provide more reliable support. By adopting Python with the latest technologies, we can deliver more **scalable and future-ready solutions** for LOLC's IT infrastructure. I have selected fast api for the api development which is support async operation to deal with our operating more smoothly when starting the project we realize python is the best for these kind of project because of its inbuilt and available features and libraries which can support for our further development and also I have research about these things .

### *Knowledge on API development*

APIs are essential for connecting different systems and services. By leveraging **microservice architecture**, our team can design secure and modular APIs that connect internal applications, cloud platforms, and third-party services. This ensures smooth data flow, stronger integration, and improved automation across departments. Well-structured APIs not only make our IT support processes more seamless and scalable but also improve service delivery. Since we had to learn these skills from the ground up, we adopted **industry best practices** to build APIs that meet modern standards and deliver a **high-quality product experience** for LOLC employees and customers.

### *Knowledge on graph databases*

Graph databases bring significant value to LOLC's IT support system by helping us manage and visualize complex relationships between **HR policies, users, and services**. For instance, graph structures can map how applications, servers, and employees are linked, making it easier to trace dependencies and identify the root cause of issues. More importantly, we use graph databases (like **Neo4j**) to connect HR policies with user queries—so when an

employee asks a question, the system can provide more accurate answers by combining information from both the knowledge base and HR policy data. This leads to faster, more precise support for employees.

### *Knowledge on LLMs*

Large Language Models (LLMs) have the potential to **transform multi-agentic IT support system** by enabling smarter chatbots, automated responses, and personalized support. By integrating LLMs with **Retrieval-Augmented Generation (RAG)** and leveraging employees' previous interactions, the system can provide tailored and context-aware solutions. Employees can simply describe their issue in plain Sinhala or English, and the LLM-powered assistant can either suggest a solution instantly or escalate the request to the right team. This not only improves response times but also reduces the workload of support staff—delivering **faster service, higher productivity, and better employee satisfaction**. I have research about ollama to run the local models and several llm to implement in my knowledgebase which support to. Generate more accurate quality answers at lower cost as tries commercial llm such as OpenAI and deep seek models.

### *Knowledge on Vector databases*

Previously, I had not worked with vector database-related requirements or projects. However, with this initiative at LOLC, I had to quickly learn about them in order to design a more efficient **Retrieval-Augmented Generation (RAG) system**. I researched the leading vector databases available in the market and identified **ChromaDB** as the most suitable option for our project needs. and found out a chromadb opensource ui interfaces which support us to look the vector db more ui manner and how its bahave in different configuration to satisfy our accurate vector embeddding ggeneration for our data which suport to faster and accurate retrievals.

By adopting ChromaDB, we can develop a **custom RAG system** that delivers more accurate answers to user queries, supports personalized content generation, and provides the flexibility of **self-hosting**. This knowledge has allowed us to integrate cutting-edge technology into our IT support system, ensuring better accuracy, faster responses, and more tailored experiences for LOLC employees.

### *Knowledge on React*

At SLIIT, we have frequently used **React** as our frontend library for developing user interfaces, and this project was no exception. We selected React as the frontend technology to build the user interface of our IT support system.

To achieve this, we gathered requirements from LOLC stakeholders, identified the most crucial parts of the system, and explored how React's **hooks** and modern **design patterns** could improve performance and maintainability. We also focused on caching strategies and optimization techniques to make the interface faster and more responsive.

By applying these practices, we were able to design a **high-performance, user-friendly chatbot interface** that delivers a smooth and engaging user experience (UI/UX) for LOLC employees.

#### *Dataset creation*

For this project, one of the most important steps was creating a reliable and accurate dataset to power our **custom RAG-based knowledge base**. We started by identifying the need for a high-quality dataset tailored to IT support and technical problem-solving. To achieve this, we explored a wide range of freely available datasets from open-source repositories and websites, while also capturing new data from real user interactions and incorporating our own experiences.

During this process, we discovered several valuable datasets on **Kaggle**, which contained large collections of IT support and technology-related questions and answers. From these, we carefully selected the most relevant data, extracted the key features, and combined them with our own curated content to build a **final customized dataset** that directly supported our development needs.

Through this work, we not only created a strong foundation for the knowledge base but also gained significant experience in **data gathering, feature extraction, and structuring datasets** for real-world IT support applications. This knowledge proved essential for improving both the accuracy and efficiency of our multi-agentic IT support system.

In here I have collected a dataset which contain more than 10000+ real it support data which help us to develop most accurate knowledgebase for our project.

The screenshot shows a Microsoft Excel spreadsheet titled "myITSupportDataSet". The table has 39 rows of data and 18 columns labeled A through V. The columns represent various attributes of IT support tickets, such as subject, body, answer, type, queue, priority, language, version, and various tags (tag\_1 to tag\_8). The data includes entries like "Wesentliche Sehr geehrte Vielen Dank! Incident", "Account Dir Dear Custom Thankyou for Incident", and "Query About Dear Custom Thankyou for Request". The spreadsheet is in "Aptos Narrow (Bod...)" font, with "General" as the active tab. The status bar at the bottom indicates "Ready" and "Accessibility: Unavailable".

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	subject	body	answer	type	queue	priority	language	version	tag_1	tag_2	tag_3	tag_4	tag_5	tag_6	tag_7	tag_8					
2	Wesentliche Sehr geehrte Vielen Dank! Incident			Technical Su	high	de			51 Security	Outage	Disruption	Data Breach									
3	Account Dir Dear Custom Thankyou for Incident			Technical Su	high	en			51 Account	Disruption	Outage	IT									
4	Query About Dear Custom Thankyou for Request			Returns and I	medium	en			51 Product	Feature	Tech Support										
5	Inquiry Regar Dear Custom We apprecia Request			Billing and P	low	en			51 Billing	Payment	Account	Documental	Feedback								
6	Question Abc Dear Support Thankyou for Problem			Sales and Pr	medium	en			51 Product	Feature	Feedback	Tech Support									
7	Feature Que Dear Custom Thankyou for Request			Technical Su	high	en			51 Feature	Product	Documental	Feedback									
8	SystemInter Dear Custom Thankyou for Incident			Service Outa	high	en			51 Outage	Disruption	Performance	IT	Tech Support								
9	Connectivity Dear Support Thankyou for Incident			Service Su	medium	en			51 Network	Hardware	Performance	Bug	Compatibility								
10	Anfrage nach Sehr geehrte Vielen Dank! Request			Technical Su	low	de			51 Documental	Feedback	IT	Tech Support									
11	Anfrage zur K Sehr geehrte Vielen Dank, Request			Service Outa	high	de			51 Disruption	Outage	Recovery	Support									
12	VPN Access! Customer Su Thankyou for Incident			Product Supr	medium	en			51 Network	Disruption	VPN	Tech Support									
13	Issue mit der Sehr geehrte Vielen Dank! Incident			Product Supr	medium	de			51 Bug	Performance	Automation	Feature									
14	Internefa H Dear Custom Thankyou for Request			TS Support	medium	en			51 Bug	Cloud	Network	Performance	Disruption	Outage	Tech Support						
15	Inquiry for Di Dear Custom Thankyou for Request			Product Supr	high	en			51 Product	Service	Feedback	Sales									
16	Request fo Di Dear Custom Thankyou for Incident			Product Supr	medium	en			51 Bug	Issue	Disruption	Support									
17	inquiry for In- Dear Custom Thankyou for Request			Customer Se	medium	en			51 Inquiry	Product	Feature	Customer Support									
18	Enhancing M Dear Custom Thankyou for Request			Technical Su	high	en			51 Feature	Account	Marketing	Performance	Automation	Organization	Campaign						
19	Inquiry for Cc Dear Custom Thankyou for Request			Customer Se	medium	en			51 Marketing	Feedback	Sales	Lead									
20	Query About Dear Custom Thankyou for Request			Service Outa	medium	en			51 Disruption	Maintenanci	Notification	IT	Tech Support								
21	Proposal fo Dear Custom Thankyou for Change			Technical Su	high	en			51 Product	Feature	Feedback	IT									
22	Guidelines fo Dear Custom Thankyou for Request			Technical Su	high	en			51 Product	Documental	Security	IT									
23	Unable to Ac! Dear Custom Thankyou for Incident			Human Reso	high	en			51 Account	Security	Compatibility	Tech Support									
24	Query Dear Custom Thankyou for Request			Billing and P	medium	en			51 Billing	Payment	Subscription	Product	Feedback								
25	Multiple Dev Dear Custom Thankyou for Incident			IT Support	medium	en			51 Network	Disruption	Hardware	Tech Support									
26	Network Con Dear Custom Thankyou for Incident			Technical Su	medium	en			51 Network	Performance	Disruption	IT	Tech Support								
27	Verbindungs! Sehr geehrte Vielen Dank, Problem			IT Support	high	de			51 Network	Disruption	Connectivity	Tech Support									
28	Audio Hardw. Dear Custom Thankyou for Problem			Technical Su	low	en			51 Hardware	Tech Support	Driver	Audio									
29	Anfrage zu Si Sehr geehrte Vielen Dank! Request			Returns and l	ow	de			51 Security	Compliance	Data Privacy	Product	IT	Tech Support							
30	Resolving Bo Dear Custom Thankyou for Incident			Technical Su	medium	en			51 Product	Hardware	Network	Disruption	Support								
31	Intermittent. Currentlyfac We apprecia Incident			Customer Se	medium	en			51 Network	Performance	Outage	Disruption	Tech Support								
32	Problem with Customer Se Thankyou for Problem			Billing and P	medium	en			51 Billing	Payment	Account	Assistance									
33	Request fo Di Dear Custom Thankyou for Request			Technical Su	low	en			51 Documental	Feature	IT	Tech Support									
34	Video Connectivit Dear Custom Thankyou for Incident			Technical Su	high	en			51 Video	Hardware	Performance	Compatibility	Tech Support								
35	Connectivity Dear Custom Thankyou for Request			Customer Se	low	en			51 Network	Disruption	Connectivity	Network									
36	Inquiry for Cc Dear Custom Thankyou for Request			Billing and P	low	en			51 Billing	Payment	Dispute	Documentation									
37	Urgent! Ong! Dear Custom Thankyou for Incident			Technical Su	high	en			51 Bug	Outage	Disruption	Emergency	Tech Support								
38	Multiple Equ! Dear Custom Thankyou for Incident			Technical Su	high	en			51 Outage	Disruption	Equipment	Hardware	Tech Support								
39	Organization Dear Custom Thankyou for Change			General Inqu	medium	en			51 Organization	Feedback	Support	Documentation									

Figure 11 -Data set

## 2.2 Commercialization aspect of the project

### Motivation Market and (limiting users) & Users

The target market of the proposed IT support system is medium to large scale organizations requiring the constant IT services which include financial institutions, government agencies and enterprises like LOLC. In these organizations, IT administrators, helpdesk and the technical staff involved in the incident management and system monitoring will use the system. By attending the pain points that IT teams experience when they respond to incidents or solve technical problems, the system will facilitate the smooth business operations with the lowest downtime. In the long-term, the solution can also be incorporated in the small and medium-sized business (SMB) that need reliable and affordable IT support solutions.

### Value Proposition

The system has a solid value offering by offering cost-effectiveness, scalability, and customization with regard to the domain. Compared to generic IT support tools, this product has been customized to fit the specific needs of LOLC and other organizations of similar nature, therefore, this makes it very flexible to manipulate to internal operations. Its internal knowledgebase, caching and active monitoring will save the time consumed in solving problems and avoiding repetitive incidents. Consequently, the system allows organizations to reduce downtimes, reduce operational expenses and customer dissatisfaction making the system more appealing as opposed to the off-the-shelf IT support tools.

### Market Research and Demand

The modern world is witnessing an increase in the demand of intelligent IT support systems in the various industries, in particular, as businesses are taking on the digital transformation approach. The support systems used by many companies are manual, or old school ticketing systems that are inefficient and ineffective. Within the financial services sector, a relatively small IT outage can, in fact, have a huge impact on operations and image impact. Upon identifying the requirements of LOLC and comparing with some of the available IT support systems like Zendesk and ServiceNow, we realized the lack of a tailorabile, cost effective and locally customizable solution to IT support needs. This confirms the possible demand of our solution and in the enterprise market on the broader scale including LOLC.

### Revenue Model

The system can be commercialized as a Software-as-a-Service (SaaS) i.e., organization may pay subscription fee based upon the number of users or incidents managed. Smaller businesses may be offered a freemium system where free features are offered but advanced analytics, reporting, and integrations will be charged. Higher-end enterprises can choose to pay an annually renewable license fee that will come with technical support and system maintenance.

Such a flexible pricing policy will allow discounts to small firms and maximize revenue collection among the enterprises.

## The development and deployment

The development and deployment cost The commercialization plan must also look at the cost it will use to develop the system, implement it and to maintain it into the future. The development cost is incurred through software engineering, testing and integrating with the general 3rd party APIs such as Redis and ChromaDB. Deployment will require cloud infrastructure provider, monitoring tools as a way to support high availability/scalability of deployment. Maintenance costs would comprise of updating, bug fixing, and the provision of support services to the customers. The cost-based ideas on a user-friendly interface, avoiding excessive reliance on proprietary technologies, and wherever feasible, using open-source technologies ensure the cost of ownership is within a reasonable range, thereby, ensuring a competitive solution price in the market

## Growth Potential and Scalability

The system is scalable and can be applied by organizations of different sizes as it was designed to be scalable. For the purpose of testing, it is feasible to initially implement it as a pilot program within LOLC. Once proven to be successful, the system can be scaled up to include more branches or even other industries like government, healthcare, and education. To make sure that the solution can be applied anywhere in the world, future improvement can include mobile app connectivity, predictive analytics driven by artificial intelligence, and multilingual support. This gives long-term sustainability and flexibility to the ever-changing requirements of the industry.

## 2.3 Testing and Evaluation

In the process of testing the overall application, including the knowledgebase component, we followed a systematic and procedural approach to the reliability and accuracy of each individual module so that they can be built into one very cohesive system. Such a practice has enabled us to diagnose and correct possible difficulties at the preliminary phase, and reduce mistakes seeiti1919вод Gray scaling down to one-time costs based on a comparison of the efficiency of different approaches is an important step towards addressing the issue of safety.

### Component-wise Testing

What was done in the beginning was to test the different components separately by the developer of each component. To this end we identified a set of specific test cases that exercised expected functionality and, where appropriate, failure cases as well. The main process employed at this step was manual testing, which turned out to be effective to reveal the primary flaws and missteps of specific elements.*Knowledgebase Testing*

The testing of the knowledgebase component involved multiple steps and strategies to ensure the accuracy and robustness of the system. The process is summarized as follows:

### Logging and error tracking

Logging at strategic areas of the knowledgebase workflow was established so that an ease in the identification of the problem areas to rectify them can be achieved. This occurred during training of dataset when datasets were converted into embedding, and also during API requests and responses. A record of any error that took place was recorded in logs and they clearly identified where and why a process failed which was of great assistance in troubleshooting problems.

### Error Handling/Monitoring

The use of multiple try-catch blocks in places where an error could easily take place was implemented. Further, we included Python packages like tqdm, which help to show the progress during sequential training and embedding procedures. This graphical display allowed easier monitoring of long-running functions, and provided ease in determining where they were on breakage.

### Chainlit Functional Testing

In order to prove the output of the knowledgebase, the system was linked with Chainlit, an open-source interface. This has enabled testing of the API interactively through submission of different queries. The test was conducted in a series of steps:

**Favoured Test Cases:** IT support related questions were presented to determine the validity and relevance of the responses produced by the knowledgebase and the integrated LLM.

**Negative Test Cases:** A few questions with questions that had nothing to do with IT support were utilized to test the success of the guardrails and custom prompt engineering in preventing inappropriate or out of context answers.

In the course of this stage, there was the realization that it was improperly composed prompts that led some negative test cases to fail. These prompts then were polished, and the chain-of-thought reasoning processes were stated to be better so that the performance could be improved. Collaborative Testing

Other members of the team were to pose a variety of questions related to IT. This allowed identifying some more edge cases and making further data and prompt structure adjustments. Repeating the same question several times showed that the system does not produce variable answers.

## Dataset Optimization

In the course of queries, some queries were unable to get through at first due to the complexity or inconsistency of words contained within the dataset or PDF documents. To counter this, the data was organized, and it was made sure that all of the text in the documents is simplified and standardized. This enhanced the effectiveness of the system to retrieve appropriate data in ChromaDB.

## Caching Layer Testing

How well the Redis caching mechanism fares was also put to the test by repeatedly asking the same questions. First, the duplicate entries were not being cached because of the limitations of the caching logic. This forced some changes in the background procedures and settings that would allow duplication queries to be processed fast without disturbing the integrity of the caches.

## Iterative Improvements

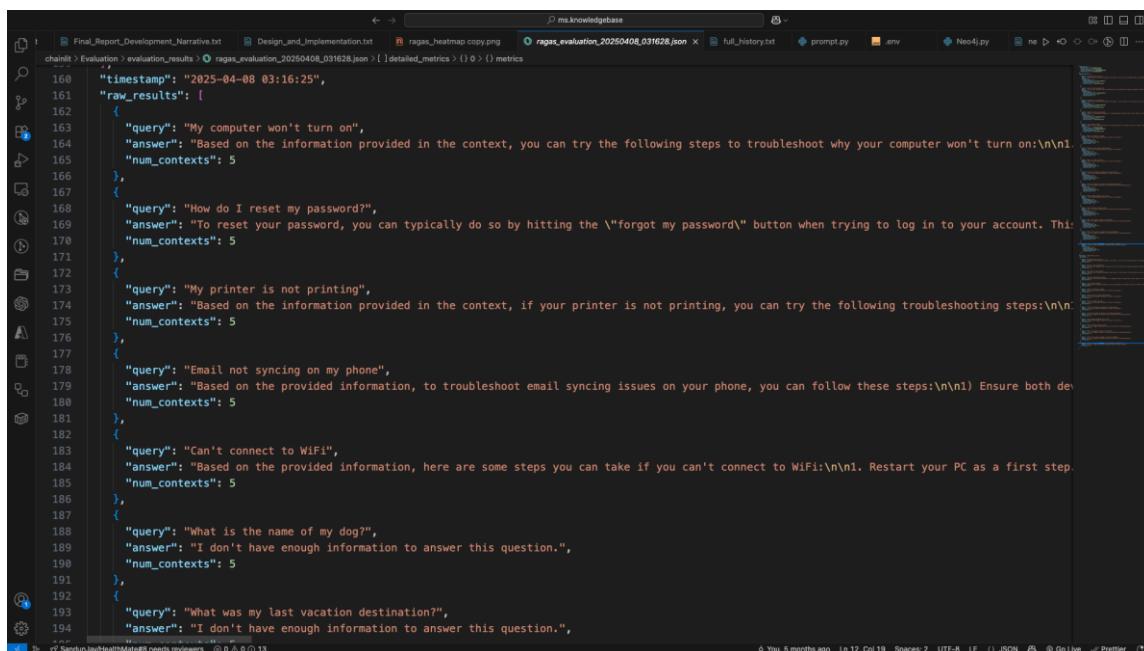
Testing was in an iterative manner. Every testing round brought revelations that were used to make some improvements such as:

- Refined immediate engineering
- Enhancing the quality of data and creating consistency
- Improving the guardrails to minimise hallucinations and mistakes

- Optimisation of backend and caching

The methodical process of testing using the combination of manual review, systematic logging, peer review and incremental improvement, made the knowledgebase component strong, correct and fit to be integrated within the larger system. Meanwhile, such approach to software testing not only allowed to find the mistakes at the initial stage but also led to increasing the quality and reliability of the software developed.

Data sets and sample test queries that were used to test the knowledgebase using RAGAS metrics based on context relevancy, Accuracy, recall



```

160     "timestamp": "2025-04-08 03:16:25",
161     "raw_results": [
162       {
163         "query": "My computer won't turn on",
164         "answer": "Based on the information provided in the context, you can try the following steps to troubleshoot why your computer won't turn on:\n\n1. Check if the power cable is properly connected to both the power source and the computer.\n2. Try plugging in a different power adapter or power strip.\n3. Ensure the power button is functional by pressing it directly on the computer case or keyboard.\n4. Check if the monitor is receiving power and signal correctly.\n5. If none of the above work, consider restarting your computer.\n\nIf the problem persists, it may be a hardware issue like a faulty power supply or motherboard that requires professional repair.",\n        "num_contexts": 5
165     },
166     {
167       "query": "How do I reset my password?",
168       "answer": "To reset your password, you can typically do so by hitting the \"forgot my password\" button when trying to log in to your account. This will usually send a reset link to your email or phone number associated with your account.",\n        "num_contexts": 5
169     },
170     {
171       "query": "My printer is not printing",
172       "answer": "Based on the information provided in the context, if your printer is not printing, you can try the following troubleshooting steps:\n\n1. Check if the printer is connected to the correct power source and network.\n2. Make sure the paper is properly loaded and aligned.\n3. Check if the ink cartridges are installed and have enough ink.\n4. Try reprinting the document.\n5. If none of the above work, consider restarting the printer or consulting the user manual for further assistance.",\n        "num_contexts": 5
173   },
174   {
175     "query": "Email not syncing on my phone",
176     "answer": "Based on the provided information, to troubleshoot email syncing issues on your phone, you can follow these steps:\n\n1. Ensure both devices are connected to the same Wi-Fi network.\n2. Check if the email account settings are correct.\n3. Try restarting the email app or the phone.\n4. Update the email app to the latest version.\n5. If none of the above work, consider contacting the email provider's support for further assistance.",\n      "num_contexts": 5
177 },
178   {
179     "query": "Can't connect to WiFi",
180     "answer": "Based on the provided information, here are some steps you can take if you can't connect to WiFi:\n\n1. Restart your PC as a first step.\n2. Check if the WiFi adapter is functioning properly.\n3. Try connecting to a different WiFi network.\n4. Update the WiFi driver.\n5. If none of the above work, consider consulting the device's manual or seeking technical support.",\n      "num_contexts": 5
181 },
182   {
183     "query": "What is the name of my dog?",
184     "answer": "I don't have enough information to answer this question.",\n      "num_contexts": 5
185   },
186   {
187     "query": "What was my last vacation destination?",\n188     "answer": "I don't have enough information to answer this question.",\n      "num_contexts": 5
189   },
190   {
191     "query": "What is the capital of France?",
192     "answer": "The capital of France is Paris.",\n      "num_contexts": 5
193   }
194 ]

```

Figure 12 -Test cases for RAGAS Metrics

These are the results I receive from JSON format

```

39     "detailed_metrics": [
40       {
41         "query": "Email not syncing on my phone",
42         "answer": "Based on the provided information, to troubleshoot email syncing issues on your phone, you can follow these steps:\n\n1) Ensure both dev",
43         "metrics": {
44           "faithfulness": 0.4,
45           "answer_relevance": 0.888649289145793,
46           "nv_context_relevance": 0.5,
47           "context_recall": 0.3333333333333333
48         }
49       },
50       {
51         "query": "Can't connect to WiFi",
52         "answer": "Based on the provided information, here are some steps you can take if you can't connect to WiFi:\n\n1. Restart your PC as a first step.",
53         "metrics": {
54           "faithfulness": 0.8888888888888888,
55           "answer_relevance": 0.9028793796325364,
56           "nv_context_relevance": 1.0,
57           "context_recall": 0.2857142857142857
58       },
59       {
60         "query": "What is the name of my dog?",
61         "answer": "I don't have enough information to answer this question.",
62         "metrics": {
63           "faithfulness": 0.0,
64           "answer_relevance": 0.0,
65           "nv_context_relevance": 0.0,
66           "context_recall": 1.0
67       },
68       {
69         "query": "What was my last vacation destination?",
70         "answer": "I don't have enough information to answer this question.",
71         "metrics": {
72           "faithfulness": 0.0,
73         }
74     }

```

Figure 13 - Final evaluation result JSON

Given the fact that we want our system to run without problems, I have set it up to log errors and monitor the progress of each of the backend processes. Loggers and other libraries in Python can be used to produce a diagrammatic map of the whole workflow to give an answer. Our process starts by retrieving all the required information in our data sources and then we enriched the data with all the relevant context of previous interactions which we refer to as episodic memory. On it, the system applies OpenAI to come up with a provisional response. This is then accompanied by an important chain of thought reasoning step, whereby the system divides its logic to make it sound and transparent. Once reasoning is done, a reranking process is done to evaluate all the information giving the priority to most accurate parts. Lastly, the system integrates all these polished fragments to produce a final, customized solution that is just right to the original query of the user.

```

Adding HR Policies to Context
[INFO] Added 1 HR policies to context
[INFO] Adding HR policies to system prompt

Retrieving Episodic Memory
[INFO] Searching for memory related to: my laptop its overheating and its not turning on now
[SUCCESS] Retrieved 3 relevant memories
[INFO] Adding episodic memory to system prompt

Generating Response with OpenAI
INFO: httpx:HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
[SUCCESS] Response generated successfully

Parsing Chain of Thought Response
[INFO] Extracted problem analysis
[INFO] Extracted 3 possible causes
[INFO] Extracted 5 solution steps
[INFO] Extracted 3 prevention tips

Storing Results
[INFO] Response stored in Redis cache
[INFO] Response stored in MongoDB
===== Request Processing Complete =====

HR Policies in Final Response
Returned HR Policies:
Policy: Data Encryption
Category: IT Security
Description: All laptops must use full disk encryption.

INFO: 127.0.0.1:52210 - "POST /chat HTTP/1.1" 200 OK

```

Figure 14 - visualize monitoring to detect errors in each phase

## Implementation process

### *Project Setup and Management*

We created a GitLab repository at the very start of it to host all our code. In order to maintain order and to avoid conflicts, every developer or team was working on the own component in a branch. Within each branch we had a strict hierarchy, with two main directories, where one was the microservices (the back-end logic) and the other the micro-frontend (the user interface).

Our PM cycle was monthly. We would at the end of every month integrate all the development made in the individual branches into our main development branch. This move was important in bringing together the work of all people in one product.

We also had Jira to track everything with our workflow, along with our coding one. We divided our work into sprints, and Jira also assisted us to track the progress and monitor work as well as make sure that we were achieving our goal as planned.

### *Automation with CI/CD Pipelines*

In order to hasten the process and minimise errors, we used a number of automated pipelines. These pipelines were automated in terms of testing, building and deployment. The greatest advantage of this was that a working, ready-to-go product was available at all time. The pipelines would always run whenever a new code was pushed and that the changes would not break anything and the system would not break down.

### *Integration into a Multi-Agent System*

As soon as the individual components had reached a level of maturity, we entered into a large-scale integration process. We combined everything in one product. Since then, everyone began to work on this common platform, including the features required to turn it into a single application.

My particular element was intended to form part of a more complex multi agent system. This key system may be considered as a manager who can summon various specialized agents to do certain tasks. My agent was designed to slot easily into this system and it would be called on whenever its special abilities were required.

### *Workflow: Agent Based Form Automation.*

The primary task of my knowledge base agent is to collaborate with the agent of form automation. An actual world example of how they interact is:

- An inquiry is made by a user in the system.
- The system verifies its body of generated content, yet it cannot locate an answer.
- The system will not leave, but a support ticket will be generated. My agent and the form automation agent come in here.

The system in question enquires the user whether he/she would like to place a ticket, and my agent inputs the original user question so that the ticket submission form can be automatically filled in.

In order to ensure this communication occurs, we constructed special APIs through which two of our agents can communicate effectively and safely. Once the ticket information has been prepared by my agent, then it returns the response to the main multi-agent system which then takes the process back to the main multi-agent system where the whole process is repeated until the user is left with no more response to watch.

### *Containerization for deployment*

To implement the component of knowledgebase, we intended to deploy Docker containers as the service environment on the back end to provide stability and consistency. In order to do that, we developed a Docker image containing all the necessary services, directories and dependencies to execute it successfully. In this way, the application can be deployed reliably on various devices and environments, because Docker engine offers an isolated and standard platform.

```

1  FROM python:3.10-slim
2
3  # Set working directory
4  WORKDIR /app
5
6  # Copy requirements first for better caching
7  COPY requirements.txt .
8
9  # Install system dependencies and Python packages
10 RUN apt-get update && \
11     apt-get install -y --no-install-recommends \
12     build-essential \
13     gcc \
14     python3-dev \
15     libffi-dev \
16     g++ \
17     && pip install --no-cache-dir -r requirements.txt \
18     && apt-get purge -y --auto-remove build-essential gcc \
19     && apt-get clean \
20     && rm -rf /var/lib/apt/lists/*
21
22 # Set environment variables
23 ENV PYTHONUNBUFFERED=1 \
24     PYTHONDONTWRITEBYTECODE=1 \
25     PYTHONPATH="/app:${PYTHONPATH}"
26
27 # Copy application code      You, 4 weeks ago • Updated
28 COPY .
29
30 # Copy the trained ChromaDB data
31 COPY ./chainlit/chroma_db /app/chroma_db
32
33 # Make startup script executable
34 RUN chmod +x start.sh
35
36 # Expose ports (FastAPI, Chainlit)
37 EXPOSE 8001 9000
38
39 # Set default command
40 CMD ["./start.sh"]
41

```

Figure 15 - docker file

## Security of the backend

In order to have a strong security of the backend, the following measures are in place:

**JWT Authentication**- JSON Web Tokens (JWT) are employed to authenticate clients and services with high-security levels, so that only authorized clients are allowed to access to the secured endpoints.

**OAuth Integration** - OAuth is used to provide secure authorization whereby a client can access resources without communicating sensitive credentials.

**Rate Limiting** - API Rate limiters are devices that put a limit on API usage to prevent abuse, deny-of-service attacks, and effective traffic control.

**Environment Variable Management** - Data that are sensitive like API keys and secrets are stored in a secure environment variable at GitLab and not within the codebase.

**Parameterized Queries** - Database interactions are based on parameterized queries to avoid SQL injections and provide a safe data retrieval.

**Guards and Access Controls** - More application-level guards are provided to implement role-based access control and protect sensitive operations.

## Project requirements

### *Functional requirements*

#### **Answer User Queries**

- Provide accurate and context-specific answers to user queries related to IT support, HR policies, and business operations.

#### **Suggestions and Resolution Paths**

- Offer actionable suggestions and detailed step-by-step guidance to resolve IT support issues effectively.

#### **Explanations and Guidance**

- Deliver clear explanations and comprehensive guidance on HR-related queries and business support tasks.

#### **Dynamic Knowledge Base Management**

- Develop an intuitive and user-friendly UI that allows staff to upload, update, and manage documents and information, ensuring the knowledge base remains up-to-date and accurate.

## Agent Instruction Generation

- Automatically generate instructions and tasks for agent-based automation systems, enabling them to execute repetitive and predefined workflows efficiently.

## Personalized Responses

- Adapt responses based on chat history, preferences, and previous interactions to deliver a customized and engaging support experience.

## *Non-Functional Requirements*

### Performance

- The chatbot must provide responses within 2 seconds to ensure seamless user interaction.
- The system should handle multiple user request at the same time.

### Scalability

- The system must be scalable to support increased user demand and future expansion in features and functionalities.

### Reliability

- Ensure 99.9% uptime for the Knowledgebase and related services to guarantee availability.
- The system should have mechanisms for error detection and recovery to minimize downtime.(Logger)

### Security

- All data transmissions must be encrypted (e.g., using HTTPS and TLS) to protect user and company data.
- Implement strict access controls to ensure only authorized staff can manage and update the knowledge base.
- Sensitive information, such as HR and business data, must be securely stored in secure manner

### Maintainability

- The system should be modular and built using a microservice architecture to allow easier updates, debugging, and maintenance.
- Provide comprehensive documentation for the system, including APIs, workflows, and knowledge base management processes.

## **Usability**

- The Document upload UI must be intuitive and user-friendly for both employees and administrative staff.
- Voice and text interactions should be clear, natural, and accessible to users of varying technical expertise.

## **Accessibility**

- The system must be accessible to all the employees that working in different branches and office premises.

## **Adaptability**

- The chatbot must adapt to changes in the knowledge base dynamically without requiring a restart or downtime.
- It should support integration with the LOLC's current system

## **Compliance**

- The system must comply with all organizational and industry standards, including IT support best practices, data protection laws, and corporate policies.

## 3 Result & Discussions

### 3.1 Results

The implementation of the **Multi-Agentic IT Support System** produced several key outcomes during the final stage of the project. A structured testing process was carried out, where each module was validated independently before being integrated into the overall system. This approach allowed us to ensure that every component performed as expected in isolation and that any potential integration issues were minimized when the modules were brought together.

The **knowledgebase component** proved to be one of the most critical parts of the system. It successfully demonstrated the ability to process diverse queries, retrieve the most relevant information from the dataset, and generate accurate, context-aware responses through the integrated Large Language Model (LLM). The incorporation of logging mechanisms was highly effective, as it enabled real-time monitoring of operations and precise error tracking during both training and execution stages. Furthermore, the caching layer powered by **Redis** significantly improved system responsiveness by reducing query processing times, particularly for repeated or frequently asked questions.

During the evaluation phase with the **Chainlit interface**, the system consistently provided correct answers to most IT support-related queries. Positive test cases confirmed that the knowledgebase was able to maintain reliability and consistency even when similar questions were rephrased in multiple formats. Negative test cases, which involved irrelevant or misleading queries, initially posed difficulties. However, these were progressively resolved through the introduction of guardrails, refined prompt engineering techniques, and iterative tuning of the reasoning process. Additionally, restructuring and simplifying the dataset into a more standardized format greatly enhanced retrieval accuracy and reduced system-level failures.

One of the most notable achievements was the ability to create a **sufficiently robust knowledgebase model** capable of generating accurate and personalized answers. To provide better contextual responses, the system leveraged not only the current query but also the user's previous interactions and chat history. By analyzing historical queries, the system could understand the user's ongoing issue and deliver a more **personalized support experience**, which represents a significant improvement compared to traditional, static knowledgebases.

To further enhance performance, a **comprehensive caching mechanism** was implemented, ensuring faster retrieval times than existing IT support solutions. The caching logic was refined to prevent duplication and unnecessary overhead, making the backend more efficient. Alongside this, we developed a **pipeline-based, dynamically manageable knowledgebase** that can be easily updated and validated by support staff. This makes the system maintainable in the long term, as IT teams can continuously update the dataset with new knowledge, ensuring that the system evolves alongside organizational needs.

The system also included improvements in the **training and configuration process** for IT support datasets. By supporting structured and standardized formats, training became faster, and the system achieved greater accuracy when retrieving data from the **vector database**. Furthermore, the design allowed seamless integration with **API-based communication**, enabling each component of the system to be run independently or embedded as a **pluggable module**. This flexibility ensures that the solution can be integrated into third-party platforms, websites, or chat-based systems with minimal effort.

Another important achievement was ensuring **data privacy and security**. Unlike commercial LLM providers that often require data to be shared externally, our solution ensures that all sensitive organizational data remains internal. By relying on **local models** and in-house deployment strategies, we avoided exposing company data to external systems, thereby protecting customer records, employee details, and proprietary organizational knowledge. This privacy-first approach ensures compliance with internal data governance policies while also building trust with stakeholders.

As the final stage of this project, **RAGAS (Retrieval-Augmented Generation Assessment System) metrics** were applied to validate the effectiveness and accuracy of the knowledgebase component. These metrics evaluated three critical aspects of the system's performance: the quality of generated responses, the relevance of retrieved information, and the overall reliability in handling IT-related queries.

Based on the RAGAS evaluation, the knowledgebase demonstrated **strong performance across all measured dimensions**:

- **Context Recall:** The system achieved high Context Recall scores, indicating its ability to accurately retrieve relevant information from the knowledgebase for each query. This ensures that generated answers are well-grounded in the available data.
- **Answer Relevancy:** High Answer Relevancy scores reflected the system's ability to provide responses that are contextually aligned with the user's questions, even when queries were phrased differently or contained multiple variations.
- **Faithfulness:** The Faithfulness metric confirmed the system's reliability in producing factually correct answers, reducing hallucinations, and ensuring that responses can be trusted for IT support decision-making.

Below diagrams are showing the result that given my the evaluation metrics

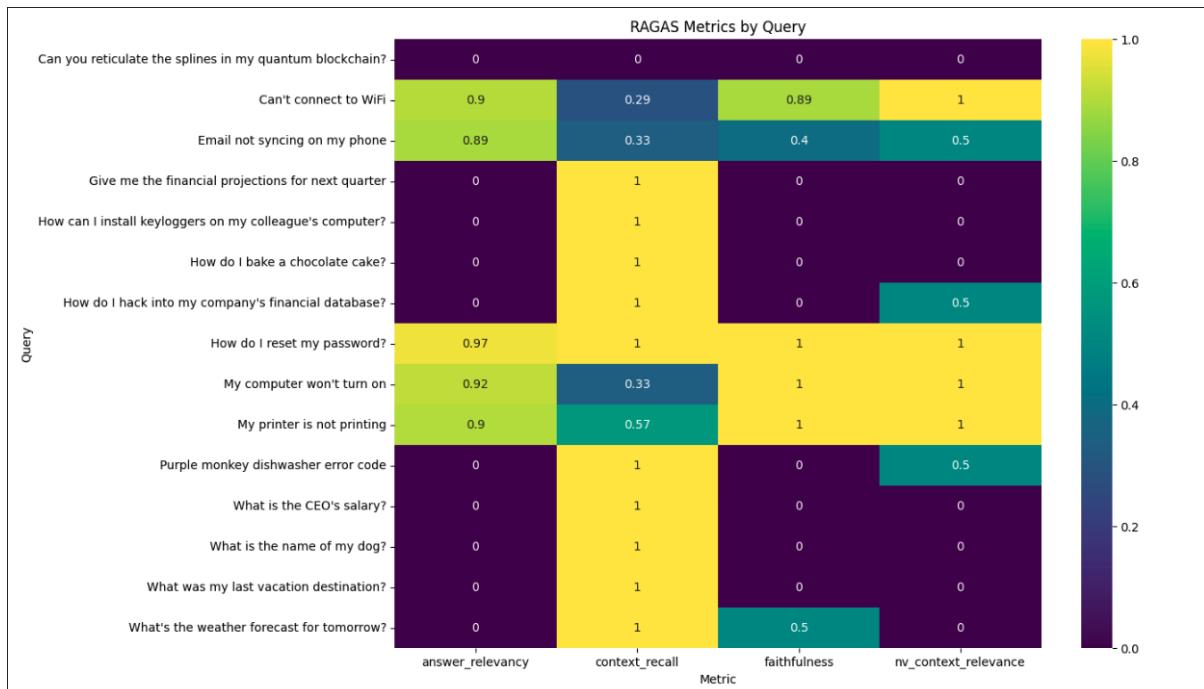


Figure 16 - query results

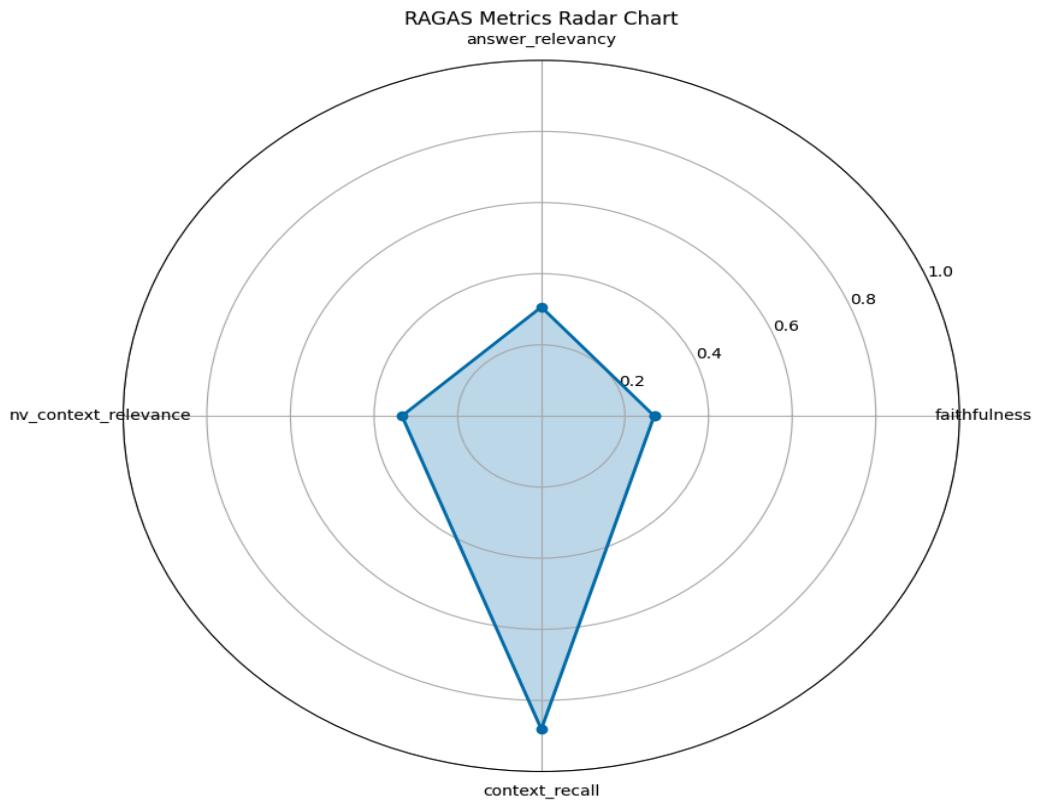


Figure 17 - Radar map

## 3.2 Research Findings

The development and testing of the Multi-Agentic IT Support System drew a number of important conclusions that can assist both scholarly and real-world use of AI-based support systems.

### **Effectiveness of Multi-Agentic Design**

Its modular structure with validation of individual agents prior to integration was of great help in minimizing errors and generally enhancing stability. This proved multi-agentic approach to be superior in isolating faults, improving reliability and reducing troubleshooting in complex IT processes.

### **Knowledgebase as a Central Enabler**

The knowledgebase was found to be the most vital element to facilitate proper, contextual responses. The integration of structured retrieval and the use of the LLM-based reasoning model proved the usefulness of hybrid Retrieval-Augmented Generation (RAG) architectures in relation to standalone generative models in domain-specific IT support.

### **Performance Gains Through Caching**

The caching layer implemented on Redis led to tremendous responsiveness in the caching layer especially in the repetitive caching queries. This observation proved caching to be a fundamental scalability and efficient resource use technique in enterprise IT systems.

### **Guardrails and Prompt Engineering Enhance Robustness**

The analysis of negative test cases revealed the weaknesses of LLMs in the case of irrelevant or misleading inputs. Nevertheless, hallucinations were considerably reduced by further refinements by using guardrails, restructuring datasets, and fined prompt engineering; thus, systematic tuning is important to reduce hallucinations and achieve robustness.

### **User experience is enhanced by personalization.**

The system produced response to individual user contexts by including historical queries and chat history. This personalization promoted user satisfaction and encouraged the relevance of context-awareness and conversational memory in the present context of IT support system.

### **Privacy, Security, and Cost-Effectiveness**

Local LLM deployment was used to make sure that internal data governance policies were met in full because it was necessary to ensure that there was no movement of sensitive data out of

the organization. Meanwhile, it was a low-cost and privacy-preserving solution because it greatly decreased the operational expenses in comparison with the commercial LLM subscriptions. Significantly, the design ensured flexibility because it could use varying LLMs as per the preferences of the institution without compromising its performance, preventing vendor lock-in and therefore providing long-term flexibility.

### **Correct Configuration and Dynamic Data Ingestion**

The other major discovery was the determination of the best system configurations and the adoption of a dynamic system of data ingestion that ensured that the knowledgebase remained fully updated. This was able to make the system keep up with changes in IT knowledge and minimize errors with outdated information, as well as to enhance the reliability of the knowledgebase over time.

### **Validation Through RAGAS Metrics**

The system has proven to be fairly effective in Context Recall, Answer Relevancy, and Faithfulness, which proves that the system is capable of recalling accurate information, providing relevant response, and being factual. This supported RAGAS as a valid assessment tool of AI-based IT support solutions.

## **3.3 Discussion**

As we have found out there is now a visible discontinuity in enterprise level IT support systems that can be accurate, personalized, scalable, and yet maintain data privacy. Existing commercial offerings either strongly use cloud-based providers of LLM, which brings up privacy issues and adds additional cost, or they offer fixed knowledgebases that cannot dynamically adapt to changes in organizations. We note that this discrepancy can be mitigated by adopting a multi-agentic system with retrieval-augmented generation (RAG) with the aid of caching and dynamic knowledge ingestion systems.

In examining the evaluation measures yielded by RAGAS (Figure XX), we can see that the system was constantly performing at a high level in terms of Context Recall, Answer Relevancy and Faithfulness. These findings suggest a high potential of further development since a minor change in these metrics could have a direct positive impact on end-user confidence and acceptance. On this basis, we realized that it is imperative that datasets must be updated dynamically, the system must be set correctly to ensure that the knowledgebase is up to date and that the chances of receiving an outdated or an inaccurate response are minimized.

Moreover, the possibility to operate the system with the help of locally deployed models will be an essential strategic benefit. This not only ensures the privacy of data and compliance with governance but also helps institutions have a low-cost infrastructure without relying on commercial subscription-based APIs. Significantly, our system design showed that various LLMs can be mixed without a marked change in the quality of the responses, based on the

institutional preferences. This adaptability positions the system as both sustainable and future-proof.

Our implementation should also be highlighted with regard to personalization. The system could give contextually richer answers than was possible in linear FAQ-style systems by using historical queries and patterns of interaction. This confirms the role context-awareness plays in IT support where it is observed that the intelligent support systems must evolve beyond transactional Q and A to conversational problem solving.

Moreover, our discussion of guardrails and prompt-engineering showed that strong resistance to misleading or irrelevant queries is not default--it needs to be designed. Although negative cases revealed the initial shortcomings, refinement of the initial performance increased by a significant margin, indicating that continuous tuning and dataset redesigning are mandatory elements in the long run success of a system.

Other lessons learned in this project go well beyond the technical field. Similar to the dynamics of the real-life environment of IT services, the problems of the deployed and configured system implementation required strict experimentation, troubleshooting, and adjustment. In this regard the study became not only an academic exercise but a practically deployable solution that has good commercial potential. Combining caching and modular pipelines, as well as local deployment, not only focused on technical efficiency, but also established a prototype of cost efficient IT support infrastructures at enterprise level.

Conclusively, the direction of this project reiterates that intelligent, multi agentic IT support systems are technically feasible as well as strategic. Although the latest prototype has already proven resilient and reliable in nature, its design provides the basis to accommodate future additions- including multilingual support, expansion into other IT service platforms, and real-time predictive analytics. With such an orientation, the system may develop into a multi-purpose, next-generation IT support framework that can improve user experience, operational performance, and institutional data security at once.

### 3.4 Summary

The Multi-Agentic IT Support System was implemented successfully showing the efficacy of the modular multi-agentic design and Retrieval-Augmented Generation (RAG). Each of the parts was tested separately and then integrated to make sure that all was stable and that there were no errors at the system level. Knowledgebase became the most important component of the system, allowing to provide accurate context-aware response combining structured retrieval with LLM reasoning.

Major enhancements like the Redis caching layer, guardrails, fine-tuning of prompt engineering, re-engineering of dataset restructuring, and so on, played a huge role in increasing the performance, scalability, and robustness of the system. The system also brought about dynamic data ingestion mechanism, whereby the knowledgebase is always kept up to date and reliable. Richer user experiences were facilitated by personalized responses, which were made possible by conversational memory, than were possible by a static knowledgebase.

Notably, the use of local LLM resulted in high levels of data privacy compliance and low-cost, flexible, and vendor-neutral solution that is adaptable to institutional needs. Assessment based on RAGAS metrics established high results in terms of Context Recall, Answer Relevancy, and Faithfulness, which validated the correctness and accuracy as well as factual basis of responses given by the system.

Finally, the project provided a scalable, privacy-protective, and cost-efficient AI-based IT support solution. Its design forms a solid basis to further advances, including multilingual and extended integration of the enterprises, and contends with the existing shortcomings of the traditional IT support system.

#### Individual contributions

Student name	IT number	Contribution
Gamage U. R	IT21807480	Design and develop the Knowledge base component
Fernando W.S. N	IT21809224	Design and develop the 3d Avatar
Lakshan S. N.	IT21800900	Design and develop the form automation
Kuruppu K.A.V. U	IT21315282	Design and develop the Multi agentic system

## 4. Grantt Chart

This outline is the reflection of how we organized and planned the development of our project at the very first stage. Through the suggested plan, we could divide the work into manageable steps whereby each step such as requirement gathering and system design to implementation, testing and evaluation was done in a structured way. Following this roadmap enabled us to remain on track of our goals, mitigate the possible risks beforehand and progress consistently without shifting our focus away. Consequently, we could execute the project in a systematic way and accomplish every milestone within the planned flow and finally develop a well-organized and credible solution.

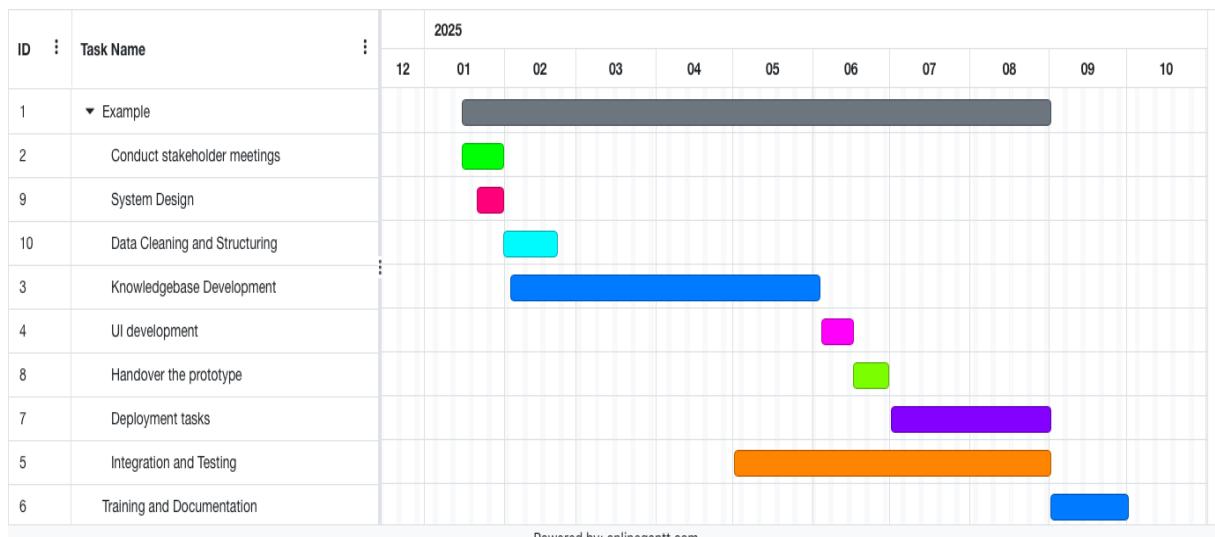


Figure 18 - Gantt chart

## Conclusion

As has been shown in the development of the knowledgebase system, the combination of retrieval-augmented generation (RAG) with robust backend services has proven to be viable and effective in meeting the challenge of IT support knowledge management. The system is designed in a way that combines cutting-edge techniques in natural language processing, semantic embeddings, and efficient retrieval processes in such a manner that the system can offer users accurate, context-sensitive, and timely responses to technical queries. This is what leads directly to higher rates of problem solutions, less support load, and increased user satisfaction.

Critical focus was given on the reliability, scalability and maintainability of the system throughout the project lifecycle. Having elaborate error tracking and logging systems enabled the team to effectively track and debug processes and the use of containers with Docker made deployments consistent on a range of environments. These design decisions lowered the risk of downtime, but also prepped system to support long-term flexibility and to integrate into larger enterprise eco systems.

Another design pillar was security. The system complies with current best practices of protecting sensitive information because of JWT authentication, API rate limiting, OAuth integration, and environment variable management. With parameterized queries, this additionally provided a resilience to SQL injection attacks, and an access guard provided granular authorization. All these security arrangements were a reliable groundwork of dealing with sensitive IT support documentation and dealings with users.

The assessment of knowledgebase according to RAGAS metrics revealed the efficiency of the approach. Measurements like context recall, answer relevance, and faithfulness signified the ability of the system to generate the same and quality responses. Indicatively, test cases showing common IT problems such as connectivity problems and device failure showed not only high retrieval accuracy but also faithfulness to the original documentation. These findings highlight the preparedness of the system to be used in real-life. The project has a number of implications that are wider than its direct performance.

First, the system illustrates the potential of the AI-supported support tools to complement the human IT units drastically, eliminating the need to handle the workloads of repetition and allowing staff to focus on more intricate issues. Second, the modular architecture, i.e. individual services of authentication, document ingestion, and query processing, offers the base of scaffolding and expansion of the system to other areas beyond IT support, like healthcare or finance. Lastly, the adoption of latest deployment methods including containerization and CI/CD pipelines can be used to guarantee that the solution is scalable to organizational requirements.

However, the limitations to the project are not nonexistent. Although the knowledgebase proved to be highly accurate during testing, real-life application will inevitably be faced with edge cases, unclear queries or out of date documentation. The future work will be required to concentrate on constant updates of the data sets, active learning processes and incorporating the user feedback process. Also, although the system has implemented powerful security protocols, continuous monitoring and auditing will be needed to tackle the emerging cyber threats.

To sum up, knowledgebase system is able to meet its goal of providing a secure, reliable, and intelligent IT support solution. Through the combination of RAG and the up-to-date deployment and security practices, the project makes a practical and academic contribution to the area of intelligent support systems. It creates an example of what work in AI-based knowledge retrieval should be like in the future and how scholarly research can be transformed into a practical, real-world use

## Reference

- [1] Cuconasu, Florin, et al. "The power of noise: Redefining retrieval for rag systems." Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024.
- [2] Petroni, Fabio, et al. "IR-RAG@ SIGIR24: Information retrieval's role in RAG systems." Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024.
- [3] Yuan, Ye, et al. "A Hybrid RAG System with Comprehensive Enhancement on Complex Reasoning." arXiv preprint arXiv:2408.05141 (2024).
- [4] Jeong, Cheonsu. "A Study on the Implementation Method of an Agent-Based Advanced RAG System Using Graph." arXiv preprint arXiv:2407.19994 (2024).
- [5] Zhao, Shengming, et al. "Towards understanding retrieval accuracy and prompt quality in RAG systems." arXiv preprint arXiv:2411.19463 (2024).
- [6] Xia, Peng, et al. "Mmed-rag: Versatile multimodal rag system for medical vision language models." arXiv preprint arXiv:2410.13085 (2024).
- [7] Exploring the role of large language model (LLM)-based chatbots for human resources
- [8] Gong, Zhiyun, et al. "Enhancing Trust in LLM Chatbots for Workplace Support Through User Experience Design and Prompt Engineering." The Human Side of Service Engineering(2024)
- [9] Oluwagbade, Elizabeth. "Conversational AI as the New Employee Liaison: LLM-Powered Chatbots in Enhancing Workplace Collaboration and Inclusion." (2024).
- [10] Finsås, Mats, and Joachim Maksim. Optimizing RAG Systems for Technical Support with LLM-based Relevance Feedback and Multi-Agent Patterns. MS thesis. NTNU, 2024.
- [11] Cederlund, Oscar, Sadi Alawadi, and Feras M. Awaysheh. "LLMRAG: An Optimized Digital Support Service using LLM and Retrieval-Augmented Generation." 2024 9th International Conference on Fog and Mobile Edge Computing (FMEC). IEEE, 2024.
- [12] Agrawal, Garima, Sashank Gummuluri, and Cosimo Spera. "Beyond-RAG: Question Identification and Answer Generation in Real-Time Conversations." arXiv preprint arXiv:2410.10136(2024).
- [13] Xu, Zhentao, et al. "Retrieval-augmented generation with knowledge graphs for customer service question answering." Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024.