

Intelligent Reasoning System Practice Module

Final Project Report

2024

*GuruDocs: ChatBot for Querying Documents using
RAG and LLMs*

Submitted By:

Alvin Wong Ann Ying (A0266486M)

Brandon Chua Hong Huei (A0168608U)

Ong Si Ci (A0266450E)

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY	3
2 PROBLEM STATEMENT AND MARKET RESEARCH.....	3
2.1 Problem Statement.....	3
2.2 Market Research.....	3
2.3 Business Value	5
3 SYSTEM ARCHITECTURE DESIGN.....	6
4 PRODUCT IMPLEMENTATION	7
4.1 Framework	7
4.1.1 Chunking Strategy.....	7
4.1.2 Embedding Model.....	8
4.1.3 Summary Technique	10
4.1.4 Retrievers.....	11
4.1.5 Prompt Template	12
4.2 Evaluation of GuruDocs Performance	12
4.3 Backend.....	15
4.3.1 Embed Endpoint	16
4.3.2 Summary Endpoint	17
4.3.3 Query Endpoint.....	18
4.4 Frontend	19
4.4.1 Streamlit.....	19
4.4.2 ReactJS	22
5 CONCLUSION.....	26
5.1 Summary	26
5.2 Challenges and Limitations	26
5.3 Future Recommendations	26
6 REFERENCES.....	28
7 APPENDICES	29
Appendix 7.1: Summary Results	29
Appendix 7.2: Prompt Templates	30
Appendix 7.3: GuruDocs Evaluation Ground Truth and Results.....	31
Appendix 7.4: Installation and User Guide	31
A7.4.1 Running GuruDocs using Docker Container	32
A7.4.2 Running GuruDocs using Local Installation	32

A7.4.3 How to use GuruDocs	33
Appendix 7.5: Mapped System Functionalities to Intelligent Reasoning System Courses	35
Appendix 7.6: Project Proposal	36
Appendix 7.7: Individual Project Report.....	37

1 EXECUTIVE SUMMARY

GuruDocs is a free chatbot designed to enhance the efficiency of information retrieval from lengthy policy and/or regulatory documents. Leveraging Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs), GuruDocs offers context-aware search results, thus improving user experience and productivity. This project addresses the limitations of keyword-based search systems and the challenges in fine-tuning LLMs for specific document queries.

2 PROBLEM STATEMENT AND MARKET RESEARCH

2.1 Problem Statement

In today's fast-paced work environment, employees frequently encounter challenges when trying to extract pertinent information from lengthy policy and regulatory documents. This difficulty often leads to frustration and a significant loss of productivity. The current reliance on keyword-based search systems falls short when dealing with complex documents that require nuanced understanding and context. Additionally, the process of fine-tuning LLMs for specific documents is highly resource-intensive, making it impractical for many organizations.

This gap in efficient document retrieval not only hampers productivity but also contributes to delays in decision-making processes. Employees may spend excessive time sifting through dense documents, leading to a drain on valuable resources and impacting overall organizational efficiency. Moreover, the lack of a user-friendly and intuitive document query solution further exacerbates these challenges, creating a pressing need for a more effective and accessible tool.

GuruDocs seeks to address these critical issues by offering a free, innovative solution that leverages RAGand LLM technologies. By enabling users to engage in natural language interactions with their documents, GuruDocs empowers individuals to quickly and accurately access the information they need, thereby enhancing productivity, reducing frustration, and facilitating informed decision-making.

2.2 Market Research

Currently in the market, several document question and answering products in the market (both commercial and open source) leverage on OpenAI's GPT4 for knowledge reasoning due to ease of

integration. The team conducted an in-depth market research review on four leading AI-powered document processing platforms –ChatDoc (ChatDOC, 2024), Hypotenuse AI (Hypotenuse AI, 2024), DocBots AI (DocsBot AI, 2024), and ChatGPT (OpenAI, 2024). Table 1 shows the comparison of GuruDocs against these products.

Table 1 Comparison of GuruDocs across different products in the market

	GuruDocs	ChatGPT's AskAI	CHATDOC	Hypotenuse AI	DocBots AI
Multiple Document Query	✗	✗	✓	✓	✓
Unlimited number of documents upload	✓	✗	✗	✗	✗
Unlimited number of queries	✓	✗	✗	✓	✗
Unlimited number of users	✓	✗	✓	✗	✗
Free	✓	✗	✗	✗	✗
Able to work with confidential data	✓	✗	✗	✗	✗

Based on the market research, it is evident that each platform offers a unique set of features and functionalities tailored to address the diverse needs of users. These platforms represent cutting-edge solutions aimed at enhancing document interaction, analysis, and collaboration. From multiple document queries to document uploads, user scalability, pricing structures, and compatibility with confidential data handling, each platform presents distinct advantages and limitations.

Multiple document query. ChatDoc allows users to seamlessly interact with PDF documents through AI chat, facilitating queries across multiple documents. Hypotenuse AI provides users with the ability to conduct multiple document queries through their HypoDoc uploads, available in paid plans starting at \$29/month. DocBots AI offers support for multiple Document Bots, enabling users to query across multiple documents with their subscription plan starting at USD \$41/month. For ChatGPT, despite its advanced language and AI capabilities, does not support multiple document queries, even with paid options, limiting its utility in scenarios requiring analysis across multiple documents.

Unlimited number of document uploads. For ChatDoc, users on the free plan can upload 2 documents per day, with a total of 10 uploads allowed, while Pro users have a generous limit of 300 uploads per 30 days, ensuring flexibility and accessibility. For ChatGPT, while the free version does not support document uploads, the paid version allows up to 40 document uploads per hour, providing users with increased flexibility in utilizing document-based interactions. Hypotenuse AI offers 5 HypoDoc uploads per month in their paid plan, which may be limiting for users with higher document upload requirements. DocBots AI provides 3 DocBots uploads per month in their subscription plan, offering users with moderate document upload needs a suitable option.

Unlimited number of users. ChatDoc supports an unlimited number of users, enabling widespread access and collaboration within organizations or communities. ChatGPT provides support for multiple users within the app interface, but access via API calls is available only in paid plans, potentially limiting scalability for certain use cases. Hypotenuse AI restricts usage to one user per paid plan, limiting collaboration and team-based utilization. Similarly, DocBots AI allows only one user per paid plan, which may restrict usage within larger teams or organizations.

Pricing. ChatDoc offers a free trial version with limited functionalities. Users can then subscribe for \$89.90 for a 360-day subscription to access the full features. ChatGPT provides a free version with reduced model capabilities and restrictions on uploading datasets or API usage. The pro version is available for \$20 monthly, and for specific tasks via API calling, it offers a pay-per-use model with an OpenAI API key. Hypotenuse AI does not offer a free version but provides pricing plans ranging from \$29 to \$59 monthly, catering to various functional needs and budget constraints. DocBots AI offers pricing plans ranging from \$16 to \$416 monthly, depending on the level of usage and functionalities required by the user.

Ability to work with confidential data. ChatDoc is limited to use on open-source cloud platforms and not suitable for government-safe infrastructure, potentially restricting its use in highly sensitive or classified environments. ChatGPT operates within open-source cloud environments and is not compatible with government-safe infrastructure, imposing limitations on its use for handling confidential data. Hypotenuse AI is also restricted to open-source cloud platforms and lacks compatibility with government-safe infrastructure, which may limit its applicability in environments requiring high levels of data security. DocBots AI shares the same constraints as other platforms, limited to open-source cloud environments and not suitable for government-safe infrastructure, potentially posing challenges for users with strict data security requirements.

As compared to existing products in the market, GuruDocs offers a unique business value of accessibility through free services, and its ability to work with confidential data. Users can access all features without facing restrictions on API keys, queries, or document uploads. Moreover, GuruDocs' open-source code on GitHub promotes transparency and allows users to contribute to its development.

2.3 Business Value

Traditional FAQ systems or chatbots have several limitations that GuruDocs is able to address:

1. **Limited Scope:** Traditional FAQ systems are typically limited to predefined questions and answers, which may not cover the full range of user queries or complex document-related inquiries.
2. **Lack of Context:** Chatbots, especially rule-based ones, may struggle with understanding context and providing relevant responses, leading to frustration for users seeking specific information from documents.
3. **Scalability Challenges:** As the volume and complexity of documents increase, traditional systems and chatbots may struggle to scale effectively, resulting in diminishing returns on investment and reduced user satisfaction.
4. **Maintenance Overhead:** Maintaining and updating traditional FAQ systems and chatbots can be resource-intensive, requiring ongoing efforts to keep information up to date and ensure accurate responses, which can add to operational costs over time.

GuruDocs brings several key business values:

1. **Improved Productivity:** By streamlining the process of extracting information from lengthy documents, GuruDocs helps employees find the information they need quickly and efficiently, saving time and increasing productivity.

2. **Enhanced User Experience:** With its context-aware search results and user-friendly interface, GuruDocs provides a more pleasant and effective way for employees to interact with policy/regulatory documents, improving overall user experience.
3. **Cost Savings:** By reducing the time employees spend searching for information, GuruDocs can lead to cost savings for businesses, as employees can focus on more important tasks.
4. **Data Security:** GuruDocs offers an on-premise solution for document query, ensuring that confidential information remains secure within the company's network, addressing data confidentiality concerns.
5. **Competitive Advantage:** By adopting GuruDocs, businesses can gain a competitive advantage by improving their information retrieval processes, making them more efficient and effective compared to competitors using traditional keyword-based search systems.

3 SYSTEM ARCHITECTURE DESIGN

GuruDocs adopts a modular and scalable architecture, leveraging containerization technologies for both the frontend and backend components. The frontend is encapsulated within a Docker container, hosting a React-based user interface along with HTML and CSS for rendering interactive and responsive user experiences. Containerization of the frontend ensures portability, ease of deployment, and efficient resource utilization.

On the backend side, GuruDocs utilizes another Docker container to house the application logic, primarily developed in Python. This backend container orchestrates the RAG process through LangChain. Chroma serves as the vector database, facilitating efficient retrieval and matching of relevant information from the documents. We leverage Ollama run the LLMs locally. Figure 1 shows the architecture design for GuruDocs.

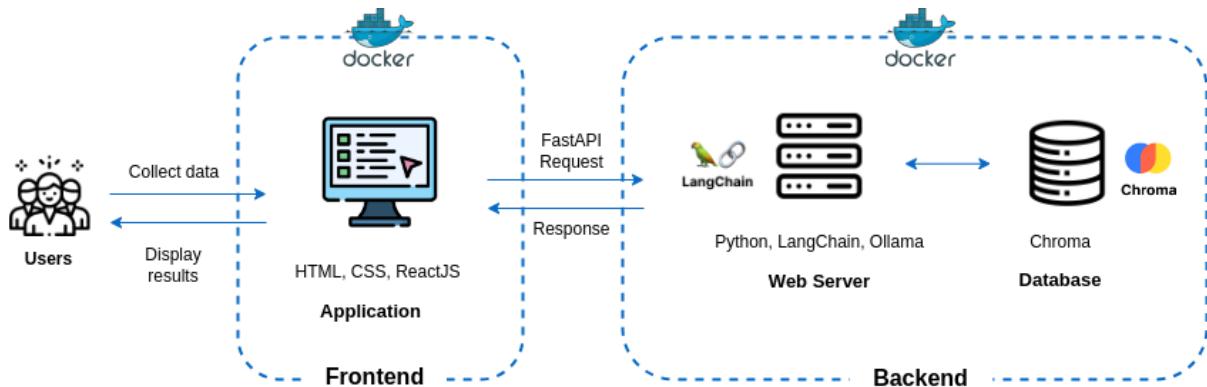


Figure 1 Architecture Design of GuruDocs

Within the backend, the process flow for GuruDocs' RAG is detailed in Figure 2. The document is first split using sentence chunking and HuggingFace Embedding Model is used to store the chunks as vectors in the vector database. When the user submits a query on the document, the same embedding model is used to extract the vector embeddings of the user query. The vector embeddings are then matched with the vectors in the vector database. Chunks with the highest similarity are returned as context that is parsed into the LLM together with the user query to generate a response.

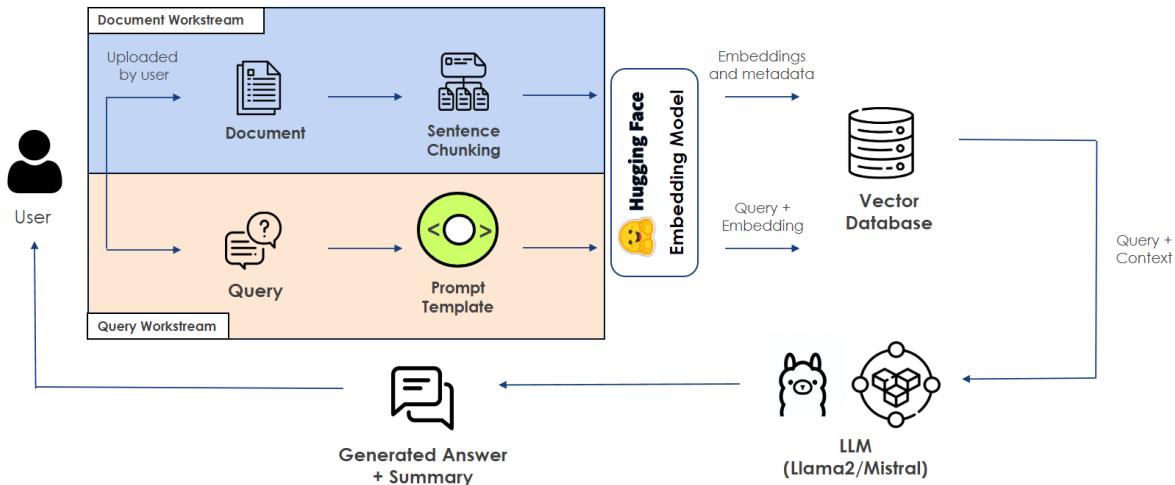


Figure 2 Process flow of RAG for GuruDocs

4 PRODUCT IMPLEMENTATION

4.1 Framework

After a thorough evaluation, the team concluded that LangChain is the optimal framework for GuruDocs due to several key factors. Firstly, LangChain's modular components allow the team to customize and switch out components such as LLM models, chunking strategy, types of retrievers and embedding models to fit GuruDocs' specific requirements (LangChain, 2024). This flexibility is crucial for GuruDocs, which focuses on providing free on-premises solution for handling confidential workloads. The ability to tailor LangChain to meet these requirements ensures that GuruDocs can maintain data security and confidentiality while providing efficient document query capabilities.

Secondly, LangChain's extensibility was a significant consideration for the team. The framework's ability to easily incorporate new features, such as optical character recognition (OCR) and speech-to-text translation from videos, makes it a future-proof choice for GuruDocs. These features are essential for enhancing GuruDocs' functionality and usability, especially as the project looks to expand its capabilities to meet the evolving needs of users.

In conclusion, LangChain's modular design, which enables customization and easy integration of new features, makes it the ideal framework for GuruDocs. By leveraging LangChain's capabilities, GuruDocs can deliver a secure, efficient, and user-friendly document query solution for corporate staff dealing with confidential documents.

4.1.1 Chunking Strategy

Chunking is a natural language processing technique that involves dividing a text into meaningful segments or "chunks" to simplify the complexity of text and extract important information for further

analysis or processing. There are two main types of chunking methods: page chunking and sentence chunking.

Sentence Chunking

`{content: 'The \nHandbook can be found at: http://hr.adobe.com/hr/main/health.aspx.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'It has been developed to \nprovide you with information regarding the terms and conditions of employment, as well as general company policies and benefits.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'Please note that the specific terms of your individual employment are set out in \nyour Offer Letter.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'In the event that there is any conflict between the terms of your Offer Letter and this Handbook, the terms of your Offer letter will prevail.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'If a policy is not addressed in this Handbook or your Offer Letter, please refer to the Employment Policies page at Inside Adobe.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'If you have any further queries, please contact HR.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'The Company reserves the right to change, in whole or in part, the contents of \nthe Employee Handbook.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'Any major change, in whole or in part, to the contents \nof the Employee Handbook shall be valid and effective upon notice to you.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'The \nHandbook remains the property of the Company and must be returned to the \nCompany upon demand or upon termination of employment.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'You are \nexpected to read through this handbook and comply with its.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'We \nbreak the policies in this Handbook is a serious matter and may result in disciplinary \naction.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'We hope you will find this Handbook useful.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}
{content: 'If you have any questions regarding \ninformation found in this Handbook, please contact Human Resources or your \nmanager.', 'metadata': {'source': '/home/nraway/Desktop/src/QA_Summary/PDFs/Adobe.pdf', 'page': 3}}`

Page Chunking

Figure 3 Output example for Sentence and Page Chunking on Adobe Employee Handbook

Page chunking involves dividing a document or text into larger segments, such as paragraphs or sections. This type of chunking is well-suited for summarization tasks, as it allows the summarizer to focus on the main ideas and key points of the text without getting bogged down by details.

On the other hand, sentence chunking involves dividing a text into smaller segments, typically sentences. Sentence chunking is particularly useful in scenarios where the questions are focused on specific details or facts within the text, as it helps to identify the relevant information more efficiently. Additionally, sentence chunking involves cleaning the text and removing unnecessary elements, enhancing the efficiency of information extraction.

In GuruDocs, both page chunking and sentence chunking were used based on their respective strengths. Page chunking was utilized for summarization tasks to condense longer texts into key points and main ideas, while sentence chunking was employed for question answering tasks to extract relevant information from individual sentences to answer specific questions. This approach allowed GuruDocs to effectively leverage the strengths of both chunking techniques to provide better response to the user.

4.1.2 Embedding Model

To ensure that GuruDocs do not incur any cost i.e., no API key requirement, the team experimented with various options available on LangChain (LangChain, 2024) and narrowed down to 2 models namely Ollama and HuggingFace. To determine which embedding model would be used for GuruDocs, the team experimented with both embedding models with a series of simple human generated questions. For example, “What is the policy on annual leave?”, “What is the company name?”, etc.

After thorough experimentation, the team decided to use the Hugging Face embedding model for GuruDocs. Despite the potential need for an API key, which could incur costs, HuggingFace consistently produced better results compared to Ollama. Its wide range of pre-trained embeddings and models, along with its strong performance, made it the preferred choice for GuruDocs. An

example of the result from GuruDocs using HuggingFace and Ollama embeddings when asked “What is the policy on annual leave?” is shown in Table 2 below.

Table 2 Sample Results of Ollama and HuggingFace embedding models on Adobe Employee Handbook

Embedding Model	Generated Results
Ollama	<p>Based on the provided context, Adobe's policy on annual leave appears to be as follows:</p> <p>"Employees are expected to maintain the highest standard of conduct and behavior at such events and must conduct themselves in a professional manner at all times. Failure to comply with this policy may result in disciplinary action, including but not limited to summary termination."</p> <p>There is no direct mention of annual leave in the provided context. However, it can be inferred that Adobe has a policy regarding annual leave based on the expectations for professional conduct and behavior at events. Without further information, I cannot provide a definitive answer to your question.</p>
HuggingFace	<p>Adobe's policy on annual leave provides eligible employees with up to 14 days of paid leave per year, in addition to any sick leave or other leaves of absence that may be available. Employees are entitled to use their annual leave at any time during the year, subject to advance notice and approval from management.</p> <p>Annual leave accrues at a rate of 14 days per year for employees who have completed their probationary period and have been employed by Adobe for at least six months. Employees who are covered by the Employment Act may be entitled to additional sick leave or other leaves of absence, as well as statutorily mandated benefits such as paid sick leave and hospitalization pay.</p> <p>Adobe's policy on annual leave is designed to provide employees with a reasonable amount of time off each year to relax, recharge, and pursue personal interests. By allowing employees to use their annual leave at any time during the year, Adobe aims to promote work-life balance and support employee well-being.</p> <p>To apply for annual leave, employees must submit a leave request form to their manager and HR representative in advance. The form can be found on Adobe's intranet or by contacting HR directly. Approval of annual leave requests will be based on business needs and availability of staff, and employees will be notified of the approval or denial of their request.</p> <p>It is important to note that annual leave does not accrue and will be refreshed to 14 days at the beginning of each calendar year. If an employee leaves Adobe for any reason, they will not be paid out for their unused annual leave.</p>

By selecting the Hugging Face embedding model, the team is confident that GuruDocs will be able to deliver a robust and effective document query solution that meets the needs of corporate staff dealing with confidential documents.

4.1.3 Summary Technique

Summarization tasks in Natural Language Processing (NLP) involve condensing a piece of text into a shorter version while retaining the key information and main ideas. LangChain offers a range of tools and techniques for both extractive and abstractive summarization tasks (Sojasingaray, 2024). Some of the key features and offerings of LangChain in this area include:

- Stuff Technique: LangChain's Stuff technique focuses on condensing text by removing redundant or irrelevant information while retaining the essential content. This technique is particularly useful for extractive summarization tasks where the goal is to identify and extract key sentences or phrases.
- Map_Reduce Technique: LangChain's map_reduce technique involves breaking down the text into smaller units and then mapping and reducing these units to extract the most important information. This technique is suitable for both extractive and abstractive summarization tasks, especially when dealing with large volumes of text or complex documents.
- Refine Technique: LangChain's refine technique focuses on refining and improving the quality of the summary generated by other techniques. This technique is beneficial for ensuring that the summary is accurate, coherent, and well-structured, especially in abstractive summarization tasks.

In the selection of the appropriate summarization technique for GuruDocs, several key considerations were carefully weighed, including processing speed and the accuracy of the summarization. The table below shows the analysis of the 3 summarization techniques. Summarization results can be found in Appendix 7.1.

Table 3 Analysis of Summarization Techniques

Summarization Technique	Stuff	Map_Reduce	Refine
Processing Time	4.9s	1m 48.9s	2m 59.7s
Details provided in Summary	Did not provide an overview of the document uploaded; Provided key sentences of certain sections of the document, missing out on most information.	Provided an overview of the document uploaded and pointed out the important sections deemed.	Provided an overview of the document uploaded along with short summaries of the certain sections of the document.

While the Stuff Technique offers the advantage of providing a summary in a relatively shorter timeframe, it was found to be inadequate in producing accurate summaries of the uploaded documents. This limitation arises from the technique's focus on condensing text by eliminating

redundant or irrelevant information, which may sometimes result in the exclusion of essential details from the summary.

In contrast, the Map_Reduce Technique, while potentially less accurate than the refine technique, offers a significant advantage in terms of processing speed. In addition, as the summary feature is meant to provide users a broad idea of the document uploaded, the requirement on the level of detail in the summary is lower. In conclusion, the team prioritized the speed and efficiency of processing summaries while still maintaining an acceptable level of accuracy and opted to implement the Map_Reduce technique for its summarization tasks.

4.1.4 Retrievers

In LangChain, Retrievers are components responsible for retrieving relevant information from a knowledge source, such as a document or database, based on a given query. They play a crucial role in information retrieval tasks by narrowing down the search space and improving the accuracy of the final results.

RetrievalQA and ConversationalQA are two approaches used in language processing, each with its own focus and objectives (Yang, 2024). RetrievalQA is primarily concerned with retrieving specific information from a knowledge source in response to a query. It is commonly used in applications where users need precise and factual answers to their queries, such as question-answering systems. For example, a RetrievalQA system may be used to answer questions like "What is the name of the company?" or "What are the types of leave available?" by retrieving the information from a reliable source.

ConversationalQA, on the other hand, focuses on maintaining context and continuity in a conversation by providing relevant responses based on previous interactions. It is commonly used in chatbots and virtual assistants, where the system needs to understand the context of the conversation and provide relevant responses. For example, in a ConversationalQA scenario, a user might ask a chatbot "What is the name of the company?" and then follow up with "Who is the founder?" The chatbot needs to maintain the context of the conversation and provide a relevant response based on the previous interaction.

The key differences between RetrievalQA and ConversationalQA lie in their approach to context handling, response generation, user interaction, and application. ConversationalQA systems focus on maintaining context and continuity in a conversation, generating responses dynamically based on the conversation's context. In contrast, RetrievalQA systems retrieve pre-existing information from a knowledge source and are more straightforward in providing direct answers to queries.

GuruDocs has opted to use ConversationalQA as the preferred option as it outperformed RetrievalQA in both speed and accuracy of response. Furthermore, its ability to maintain context, provide dynamic responses, and handle follow-up questions enabled GuruDocs to be more natural, engaging, and contextually aware. This enhancement in user experience of information retrieval made it an easy decision.

4.1.5 Prompt Template

Prompt template in NLP is a predefined format or structure used to guide the generation of natural language text. It consists of placeholders or variables that are filled in with specific information to create a coherent and contextually relevant response. Prompt templates play a crucial role in NLP tasks such as text generation, question answering, and summarization, as they provide a structured approach to generating text and ensure that the output is tailored to the intended purpose.

Firstly, they provide a standardized format for generating text, which helps to ensure consistency and coherence in the output. This is particularly important in applications where the generated text needs to be clear, concise, and grammatically correct. Secondly, prompt templates enable the generation of contextually relevant responses by specifying the information that should be included in the output. This helps to ensure that the generated text is relevant to the user's query or input. Finally, prompt templates can also improve the efficiency of text generation by providing a structured framework that guides the generation process, reducing the need for manual intervention and editing.

For GuruDocs, we experimented between 2 frameworks: i) Basic question-answering prompt framework; and ii) CO-STAR prompt framework (Teo, 2024). The basic question-answering prompt template provided the LLM a persona with a task to do i.e., AI Assistant for question-answering tasks. For the CO-STAR prompt template framework, it considers key aspects such as Context, Objective, Style, Tone, Audience, and Response to influence the effectiveness and relevance of the LLM's response, leading to more optimal responses. An example of prompts for both frameworks can be found in Appendix 7.2.

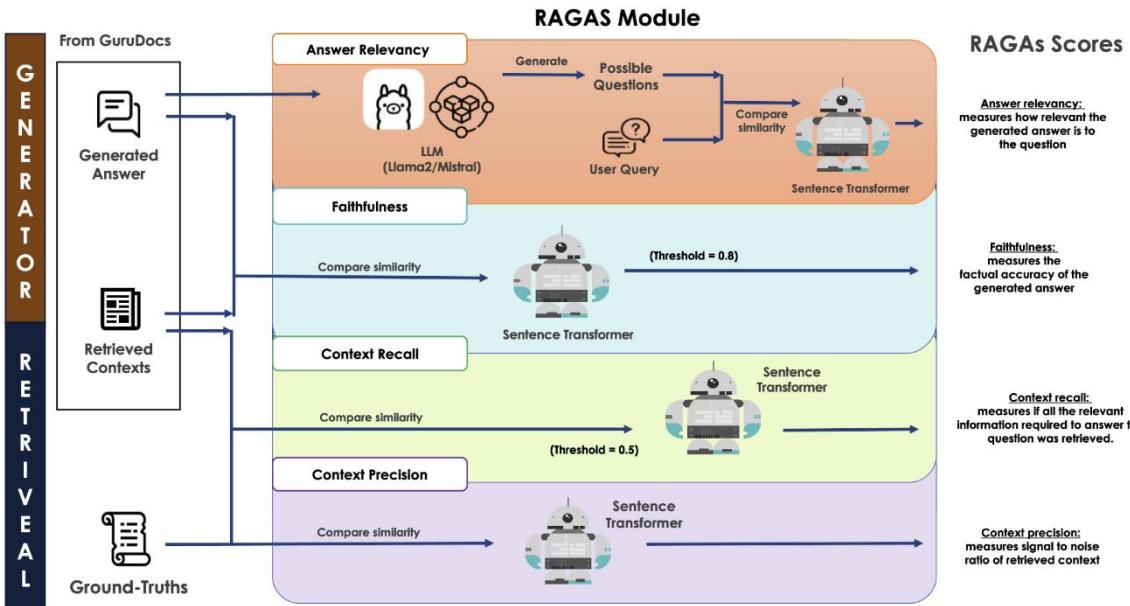
Through our experimentation, we found that the performance varied between the 2 frameworks with the basic question-answering outperforming the CO-STAR framework for 1 document and the CO-STAR framework outperforming basic question-answering framework for another. However, we noted a significant difference in the speed of response between the 2 frameworks. As such, we decided to adopt the basic question-answering framework for GuruDocs.

4.2 Evaluation of GuruDocs Performance

For evaluating GuruDocs, our team has chosen to adopt the Retrieval Augmented Generation Assessment (RAGAs) framework (Ragas, 2024). This choice was made due to the framework's comprehensive nature, which evaluates both the retrieval and generation aspects of an RAG application. Our goal in employing RAGAs is to ensure the accuracy of retrieved context and the coherence of generated responses based on the provided context and prompts.

In evaluating the retrieval component, our focus is on validating the accuracy and alignment with the intended context based on user queries. To achieve this, we utilized metrics such as context precision and context recall. Conversely, our evaluation of the generator component aims to ensure the coherence and consistency of the generated answers with the retrieved context and prompts. For this, we used metrics such as faithfulness and answer relevancy to measure the similarity and fluency of generated answers compared to ground truth. Figure 4 below shows the evaluation workflow for GuruDocs.

Figure 4 GuruDocs Evaluation Workflow



To facilitate the evaluation, ground truths were needed. Initially, we explored the use of ragas, an online application. However, we found that it required OpenAI's API key and was not aligned with GuruDoc's requirement of being cost-free. As a result, we developed a modified version of ragas by leveraging open-source Text-to-Text Transfer Transformer (T5) Vanilla model to generate the QA pairs. Additionally, we conducted a comparative analysis using the Transformer-QG-on-SQuAD model to ensure the quality of the QA pairs generated. A total of 230 QA pairs, 80 for Adobe Employee Handbook and 150 for NTUC Employee Handbook, were generated and manually verified for the evaluation of GuruDocs.

Our experimentation focused on three key components: the Large Language Model (LLM), chunking technique, and retrieval technique. Regarding the evaluation of the LLM, we tested Llama2 and Mistral, finding that Llama2 generally performed better, although the difference was not always significant. Consequently, we concluded that GuruDocs is LLM-agnostic, allowing users to select their preferred LLM, with Llama2 set as the default option.

In evaluating the chunking technique, we determined that sentence-level chunking was superior. Across both the Adobe and NTUC Employee Handbook, sentence-level chunking consistently improved all four metrics—Faithfulness, Answer Relevancy, Context Precision, and Context Recall—with the most substantial improvement seen in Faithfulness for the Adobe Employee Handbook, increasing from 6.82% to 45.46% as shown in Table 5 below.

Table 4 Evaluation Results on Adobe Employee Handbook

Mistral		Faithfulness	Answer Relevancy	Context Precision	Context Recall
Chunking					
Page Level		6.82%	55.62%	14.16%	3.26%
Sentence Level		45.46%	63.49%	14.77%	3.81%
Retrieval	Prompt Engineering				
Conversational QA	Minimal (provide basic instructions)	49.13%	64.12%	15.18%	4.14%
		49.33%	63.65%	15.02%	4.17%
Retrieval QA	COSTAR	30.97%	53.20%	16.68%	3.58%
Llama2		Faithfulness	Answer Relevancy	Context Precision	Context Recall
Chunking					
Page Level		15.61%	52.09%	14.37%	3.40%
Sentence Level		59.14%	57.67%	14.60%	3.47%
Retrieval	Prompt Engineering				
Conversational QA	Minimal (provide basic instructions)	67.06%	60.53%	14.32%	3.78%
		65.40%	58.53%	14.17%	3.77%
Retrieval QA	COSTAR	21.91%	41.42%	15.97%	2.10%

In exploring the retrieval technique, we explored three options: RetrievalQA, ConversationalQA with a basic question-answering prompt framework, and ConversationalQA with the CO-STAN prompt framework. Surprisingly, ConversationalQA outperformed RetrievalQA in all scenarios, despite RetrievalQA being designed for RAG applications. Additionally, the adoption of the CO-STAN framework did not significantly impact the results but did notably increase GuruDocs' processing time. As such, we decided to adopt ConversationalQA with the basic question-answering prompt framework.

Throughout our experiments, we observed consistent poor performance in Context Precision and Context Recall metrics, as shown in both Tables 5 and 6. Further investigation revealed that the issue stemmed from our generated ground truth answers lacking detail. These answers were often brief phrases, which did not align well with the training of the models, resulting in the models' inability to match the answers and consequently poor performance in Context Precision and Context Recall, which rely on ground truth. More details on the ground truth and prompt completions used for evaluation can be found in Appendix 7.3.

In conclusion, our experiments demonstrated that employing sentence-level chunking and ConversationalQA with a basic question-answering prompt framework yielded notably better performance across both documents. As such, these approaches will be incorporated into GuruDoc's backend solution.

Table 5 Evaluation Results on NTUC Employee Handbook

Mistral		Faithfulness	Answer Relevancy	Context Precision	Context Recall	
Chunking						
Page Level		30.18%	55.54%	13.28%	1.69%	
Sentence Level		45.93%	57.72%	12.58%	1.39%	
Retrieval Prompt Engineering						
Conversational QA	Minimal (provide basic instructions)	45.32%	55.09%	13.66%	1.90%	
Retrieval QA	COSTAR	45.45%	55.36%	13.63%	1.97%	
		34.23%	50.31%	14.34%	1.60%	
Llama2		Faithfulness	Answer Relevancy	Context Precision	Context Recall	
Chunking						
Page Level		42.07%	52.97%	13.18%	1.98%	
Sentence Level		51.71%	59.04%	12.75%	1.43%	
Retrieval Prompt Engineering						
Conversational QA	Minimal (provide basic instructions)	57.20%	59.05%	12.88%	1.59%	
Retrieval QA	COSTAR	59.49%	57.69%	12.82%	1.59%	
		23.60%	42.31%	13.20%	0.87%	

4.3 Backend

In crafting the backend architecture for this project, several considerations were pivotal in the selection of FastAPI as the framework of choice. FastAPI's asynchronous capabilities were instrumental in handling concurrent requests efficiently, especially during intensive tasks such as document processing, embedding, and summarization. Its robust design and adherence to modern Python standards facilitated the development of a scalable and responsive API. Additionally, FastAPI's support for OpenAPI documentation streamlined endpoint management and ensured clear documentation of API functionalities, which will be crucial for future scaling and deployment of GuruDocs.

The backend system comprises three key endpoints designed to handle distinct functionalities. These endpoints—embed, query, and summary—form the backbone of document embedding, user query handling, and document summarization within the application. While currently we did not implement the APIs asynchronously, this feature will be needed in the future for full scale production with higher workloads. Besides the three key functional endpoints, we have also implemented a ping endpoint to allow us to check whether the FastAPI service is healthy. Figure 5 shows the OpenAPI documentation of our ping endpoint.

The screenshot shows the OpenAPI documentation for a 'Ping' endpoint. At the top, it specifies a 'GET' method and the URL '/ping'. Below this, there's a 'Parameters' section which states 'No parameters'. In the top right corner, there's a 'Try it out' button. The main content area is titled 'Responses' and contains a table for the 200 status code. The table has columns for 'Code', 'Description', and 'Links'. The 'Description' column for 200 says 'Successful Response'. The 'Links' column for 200 says 'No links'. Under the 'Description' column, there's a 'Media type' dropdown set to 'application/json', with a note below it saying 'Controls Accept header.' There are also 'Example Value' and 'Schema' links. A large black redacted box covers the actual response example.

Figure 5 OpenAPI Documentation of GuruDoc Ping Endpoint

4.3.1 Embed Endpoint

This endpoint facilitates the extraction and embedding of PDF documents received from the frontend. Utilizing FastAPI, it processes base64 strings representing PDFs that are passed from the frontend, storing them temporarily for efficient handling. Upon receiving the document, the backend employs llm_utils functions to dissect the content into sentences, chunk them appropriately, generate vector embeddings, and store these embeddings in Chroma vectorstore. The vectorstore object and pages are stored in different dictionaries, each with unique UUIDs as keys to access the objects. To manage data transfer efficiently, the API response returns the unique UUIDs of the vectorstore and pages, enabling subsequent operations without overwhelming the response payload. Upon completion of document embedding processes, the temporary PDF file is removed from local file storage and the "summary" endpoint will be automatically called. Figure 6 shows the OpenAPI documentation for embed endpoint.

POST /embed

Parameters

No parameters

Request body required

Example Value | Schema

```
{
  "model": "string",
  "files": [
    "string"
  ]
}
```

Responses

Code	Description	Links
200	Successful Response	No links
422	Validation Error	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Figure 6 OpenAPI documentation of embed endpoint

4.3.2 Summary Endpoint

This endpoint takes in the UUID of the pages chunks from the “embed” endpoint and the name of the LLM in the API body. The pages are then retrieved from the dictionary stored in memory and the relevant functions from llm_utils will then be called to generate a summary of the document as described in Section 4.1.3 above. This endpoint is automatically called when the document has been successfully chunked and embedded in the vectorstore. Figure 7 shows the OpenAPI documentation of the “summary” endpoint.

The screenshot shows the OpenAPI documentation for a 'summary' endpoint. At the top, it indicates a **POST** method and the URL **/summary**. The **Summary** field contains the text 'Summary'. Below this, there's a **Parameters** section which states 'No parameters'. In the **Request body** section, the **required** status is shown, along with a dropdown menu set to 'application/json'. A code block shows an example JSON payload:

```
{
  "model_option": "string",
  "pages_id": [
    "string"
  ]
}
```

Under the **Responses** section, there are two entries:

- Code 200**: Description 'Successful Response'. It shows a media type dropdown set to 'application/json' with the note 'Controls Accept header.' Below it is an example value: 'string'.
- Code 422**: Description 'Validation Error'. It shows a media type dropdown set to 'application/json'. Below it is an example value: a JSON object with 'detail' key containing an array of objects with 'loc', 'msg', and 'type' keys.

Figure 7 OpenAPI documentation of the “summary” endpoint

4.3.3 Query Endpoint

This endpoint takes in the unique UUID of the vectorstore where the document chunks have been embedded, the model of choice and user’s query in the API body. Similar to how the pages UUID was used to retrieve the document pages, the vectorstore UUID is used to retrieve the vectorstore object from the dictionary stored in memory. The relevant functions will then be called in `llm_utils` to generate vector embeddings of the user’s query, match it with the document vector embeddings in the vectorstore to find those that are similar and retrieve it. The text of the retrieved vector embeddings are then passed into the LLM as context to generate a completion which is returned as the API response. Figure 8 shows the OpenAPI documentation of the “query” endpoint.

Figure 8 OpenAPI documentation of query endpoint

4.4 Frontend

GuruDocs' initial frontend development phase leveraged Streamlit for its rapid prototyping capabilities and ease of use in crafting interactive data-driven applications. However, as the project progressed, we encountered limitations within Streamlit. To address these challenges and ensure a more robust and customizable frontend experience, we transitioned to the React framework. React's component-based architecture, efficient virtual DOM rendering, and extensive ecosystem of libraries and tools provided us with the flexibility of our application.

4.4.1 Streamlit

In the initial phase of development using Streamlit, our primary focus was on building foundational features that would lay the groundwork for the application's functionality. The first feature we developed was the File Upload feature, allowing users to upload documents or files directly into the application. This feature was essential as it enabled users to interact with the application by providing input data for processing.

We then implemented a Chatbot feature that allowed users to engage with the application through a conversational interface. This Chatbot served as a user-friendly way for users to interact with the uploaded documents. These initial features were designed to showcase basic functionality and were not integrated with the backend. Figure 9 shows the first iteration of the Streamlit frontend.

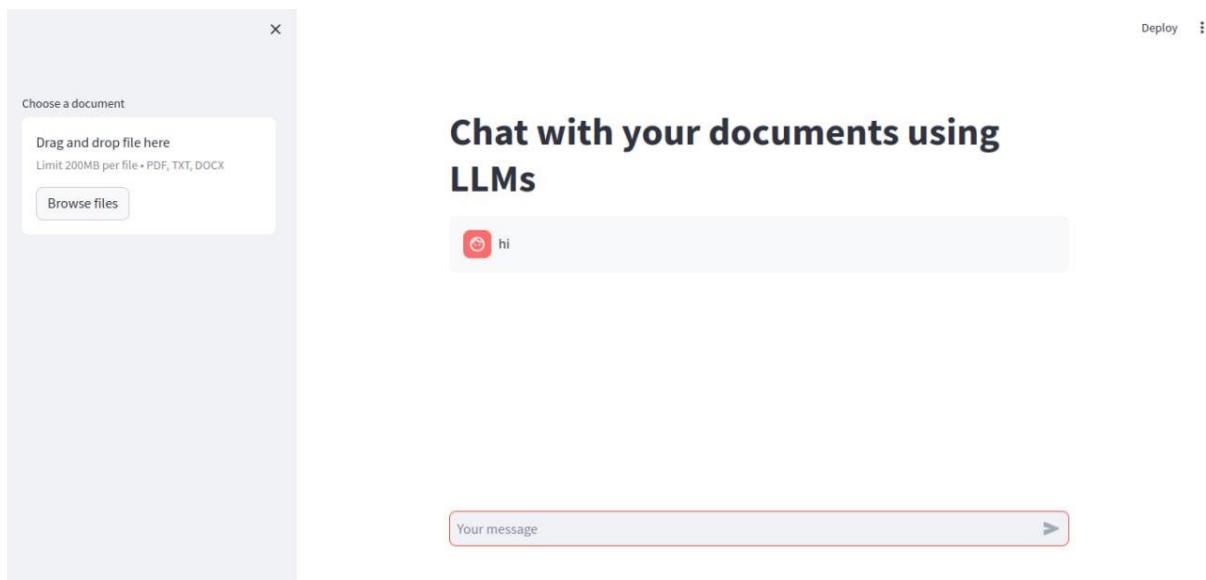


Figure 9 First iteration of GuruDocs frontend without integration with backend logic

In the second iteration, we expanded upon the initial features by integrating them with the backend RAG process logic. The File Upload feature was integrated with the backend processes to call the “embed” API to chunk and embed the document chunks in the vector database.

Simultaneously, the Chatbot feature was integrated with the backend where the vector embeddings of user’s query was extracted using the same embedding model, and matched with the embeddings in the vector database. The generated output from the LLM based on extracted contexts were then presented to the user as a formatted response.

This iteration demonstrated the key features and functionalities of GuruDocs. Figure 10 shows the second iteration outcome of the Streamlit frontend.

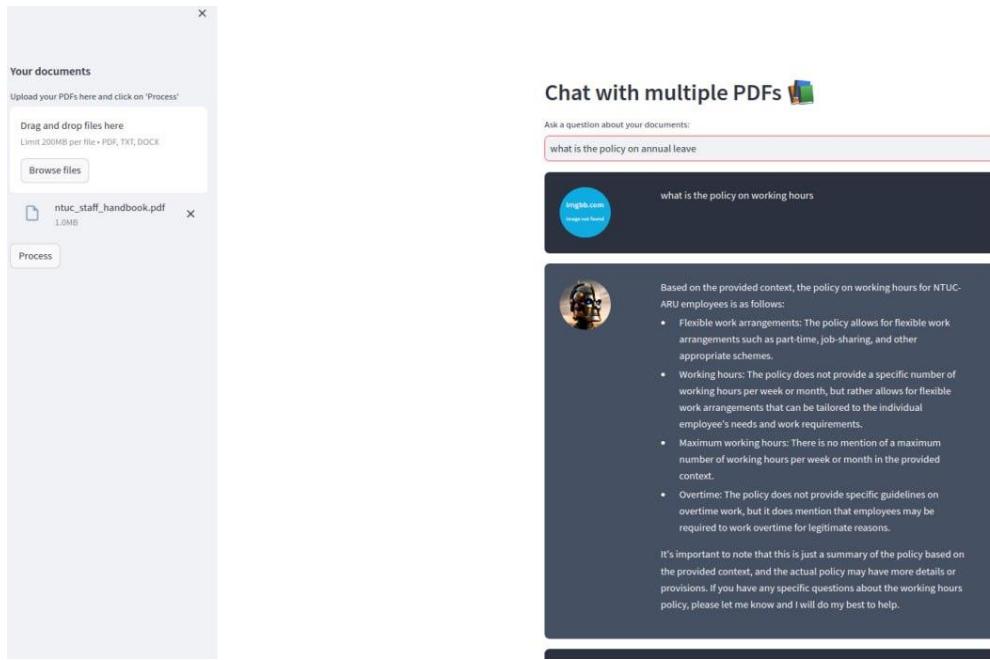


Figure 10 Second iteration of GuruDocs' frontend with integration to backend logic

The final iteration on Streamlit focused on implementing the remaining features of GuruDocs such as the Model Option feature and a placeholder for extracting document summaries. The Model Option feature provided users with the ability to choose from different LLMs to generate the response, offering flexibility and customization. This feature was also integrated with the backend to call the model that was selected by the user in the response generation step.

Furthermore, we implemented a placeholder for the document summary feature while waiting for backend developments. Figure 11 shows the final iteration of GuruDocs' frontend on Streamlit.

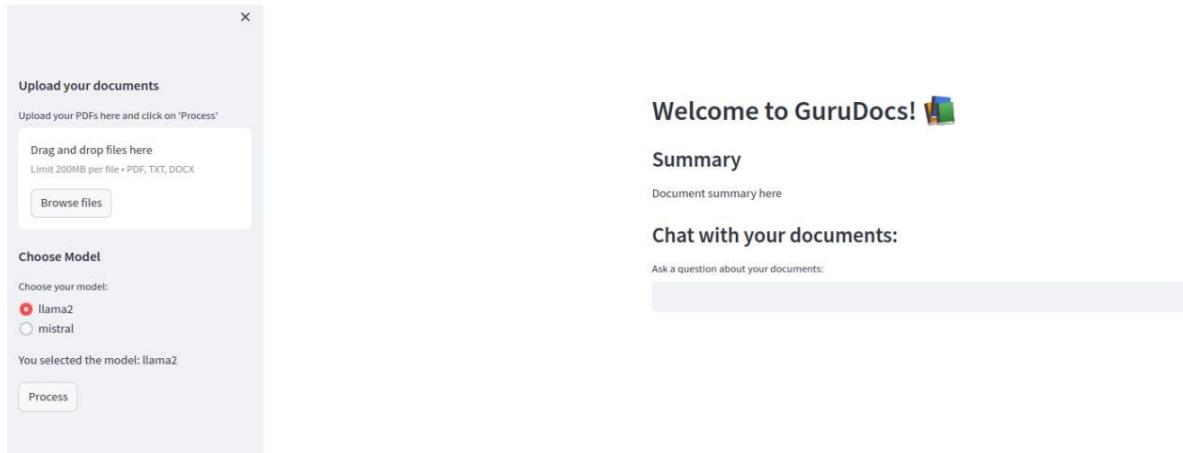


Figure 11 Final iteration of GuruDocs frontend on Streamlit

While Streamlit was chosen initially for its benefits of ease of use and pythonic nature, we also realised the limitations of Streamlit.

One notable limitation we encountered was the need for Streamlit to be run as a main application. This meant that the whole application interface would be refreshed whenever there was a small

update or when the response was returned from the backend. This led to performance inefficiencies, especially when the user poses a large number of questions.

Moreover, Streamlit's limited customizability posed challenges for displaying informative loading interfaces. For instance, our application required notifying users about ongoing processes, such as document embedding or backend retrieval and generation tasks, which often required significant processing time. However, Streamlit's constraints made it difficult to implement dynamic loading indicators or progress notifications, hindering user feedback and engagement.

We also considered a future full-fledged web deployment of GuruDocs, and Streamlit's scalability considerations emerged as a concern. Streamlit's architecture as the main application posed challenges for incorporating load balancers, gateway services, or server-side technologies like Nginx, which are crucial for managing high traffic and ensuring optimal performance in scalable web applications.

These limitations highlighted the need for a more flexible and scalable frontend framework, leading to our decision to transition to React. React's component-based architecture, robust state management capabilities, and support for server-side rendering and backend integrations addressed these challenges, allowing us to deliver a more seamless, customizable, and scalable user experience in our application's web deployment phase.

4.4.2 ReactJS

Due to the abovementioned limitations, the team decided to transition to ReactJS for the frontend framework despite not having any experience in React or frontend development. We focused on building the key features such as document upload, summary display and chatbot. Similar to the development approach taken for Streamlit, our first iteration of development on React did not include backend logic integration. To maximise real estate on the home page for users to interact with the chatbot and view the document summary, we decided to implement the document upload and model selection features in a Dialog component. Figure 12 shows the first iteration of GuruDocs on React.

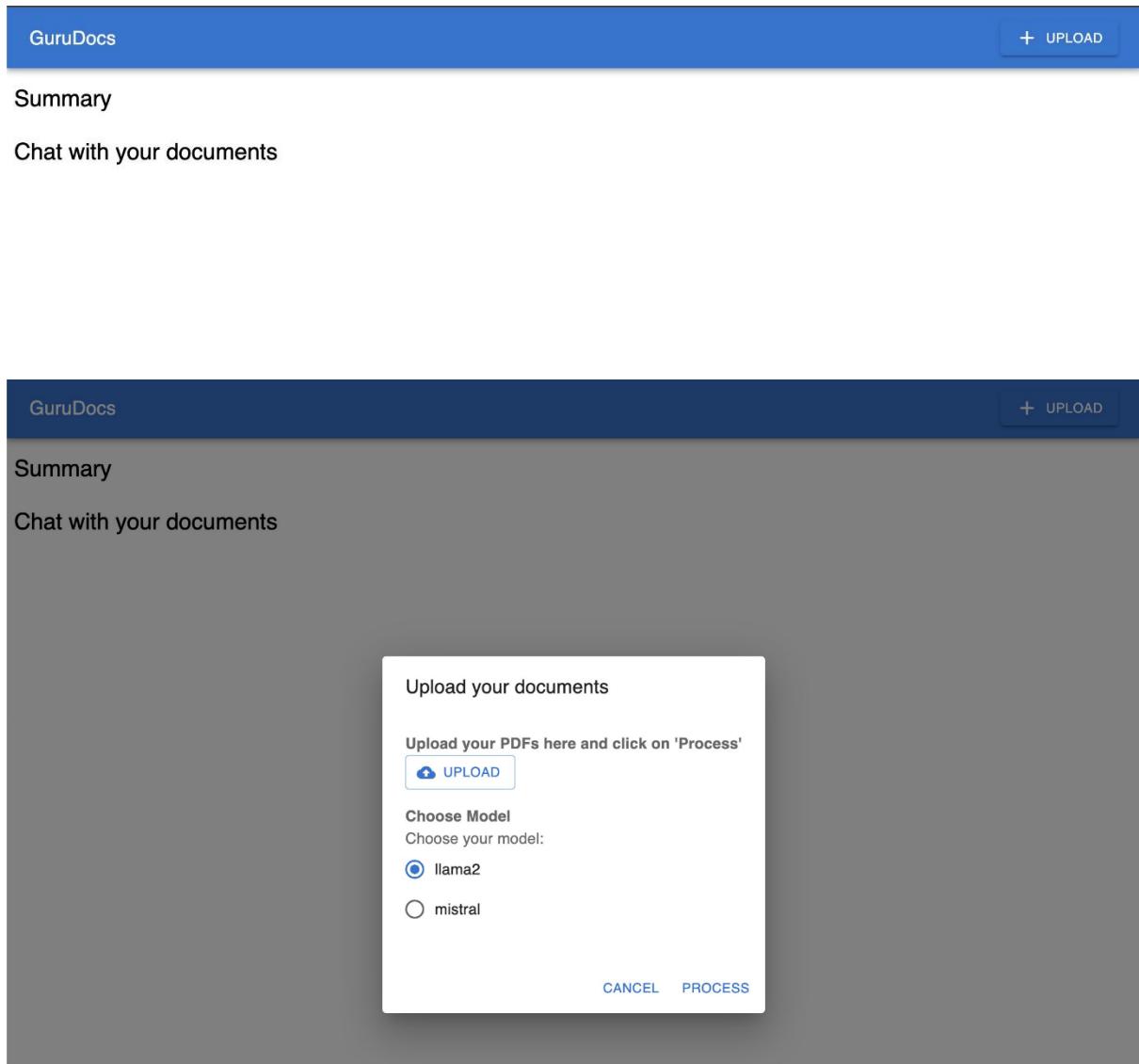


Figure 12 First iteration of GuruDocs homepage (above) and the upload Dialog component (below) on React.

In the second iteration, we integrated the features in the Dialog component with the backend logic. Once the document was uploaded and the user has selected their model of choice, the document will be sent as base64 string to the backend “embed” API to perform chunking and extract the vector embeddings to store in the vector database. As mentioned in Section 4.3.1, we also implemented the logic to automatically trigger the “summary” endpoint to be called once the vectors of the document chunks have been stored in the vector database. As such the document summaries will be displayed on the frontend once the “summary” endpoint returns the summary response. Figure 13 shows the second iteration of GuruDocs frontend.

GuruDocs

+ UPLOAD

Summary

Adobe_Handbook-1-3.pdf

The text is an employee handbook for Adobe Singapore. It covers various aspects of employment, including job roles and responsibilities, work hours and leave policies, performance management, and communication channels. Some important details include: * Job roles and responsibilities are defined for different positions within the company. * Work hours and leave policies are outlined, including paid time off, sick leave, and parental leave. * Performance management is discussed, including regular evaluations and feedback. * Communication channels are established, including a company intranet and various email addresses. Overall, the handbook provides guidelines for employees to understand their roles and responsibilities within the company, as well as the policies and procedures in place to support their work experience.

NTUC_Handbook-1-3.pdf

The text is an updated staff handbook for an organization, dated September 3, 2012. The handbook outlines the main points and details related to the organization's policies and procedures. Some of the key points include: * The handbook is updated as of September 3, 2012. * The document provides information on various aspects of the organization, including its staff, policies, and procedures. * The handbook covers topics such as employee conduct, work hours, leave, and performance evaluations. * It also includes information on the organization's mission, vision, and values. * The handbook is intended to provide guidance and support for employees in their daily work.

Chat with your documents

Figure 13 Second iteration of GuruDocs frontend where the document summaries will be automatically displayed once the documents have been embedded and summary has been generated.

In the next iteration, we implemented the Chatbot feature using React's *react-chatboot-toolkit* library and integrated it with the backend "query" API endpoint once the user submits a query. As the API response usually takes a couple of seconds to process and return a response, we return a default message on the chatbot, "Sending query, please wait..." to inform the user that the message has been sent to the backend. Once the "query" API endpoint returns a response, it will be rendered on the chatbot as a reply. We also added in icons to each of the section headers. Figure 14 shows the GuruDocs frontend implementation with Chatbot feature.

Welcome to GuruDocs!

+ UPLOAD

Summary

adobe_employee_handbook.pdf

The text is an employee handbook for Adobe Singapore. It covers various aspects of employment, including job roles and responsibilities, work hours and attendance, performance management, and employee conduct. The handbook also outlines the company's policies on confidentiality, non-disclosure, and intellectual property. Additionally, it provides information on benefits, compensation, and training programs available to employees. Some important details included in the handbook are: * Job roles and responsibilities for various positions within the company * Expectations for employee conduct and behavior both in and out of the workplace * Policies on confidentiality and non-disclosure of company information * Procedures for reporting and addressing any misconduct or violations of company policies * Information on benefits and compensation packages available to employees * Details on training programs and opportunities for professional development.

Chat with your documents

Conversation with GuruDocsBot

B Hello! What questions do you have about your document?

what is the company name?

Sending to LLM...

B Based on the provided text, the company name is Adobe Systems Incorporated.

what is the leave policy?

Sending to LLM...

B Based on the provided document, Adobe's leave policy includes the following options: 1. Medical Leave: Up to 9 days for medical emergencies, with a maximum payout of US\$250,000. 2. Maternity Leave: Up to 10 days for new mothers, with a maximum payout of US\$250,000. 3. Parental Leave: Up to 10 days for new fathers, with a maximum payout of US\$250,000. 4. Childcare Leave: Up to 10 days for employees to care for their children, with a maximum payout of US\$250,000.

Write your message here

Figure 14 Third iteration of GuruDocs frontend with Chatbot implemented.

In the final round of iteration of GuruDocs frontend, we focused on the smaller features that were important for a good user experience. First, we added a display of the filename for the file that was selected by the user as shown in Figure 15. Second, we added the Spinner components with description when the documents are being chunked and embedded, and when the summary is being generated as shown in Figure 16. This allows the user to know which processes are running. We also

added a toast message to inform the user when the document has been successfully embedded as seen in Figure 17. Finally, we implemented HTML wrapping of the texts as some of the summary or chatbot responses returned by the API consists of several lines with new lines or numbered lists.

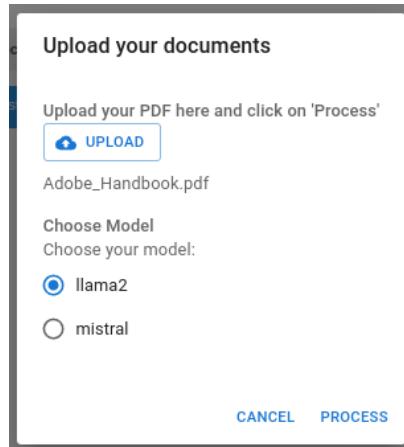


Figure 15 Display of filename of the file selected by the user in the upload Dialog component



Figure 16 (Left) Spinner component in Dialog component when documents are being chunked and embedded, and (Right) Spinner component in Summary section when the document summary is being generated

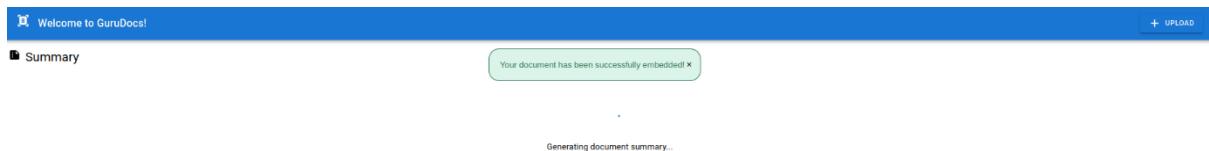


Figure 17 Toast message to inform the user that the document has been successfully embedded

While the current frontend UI may not be the most aesthetic or stylistic, it is functional and fulfills the key requirements of GuruDocs, to allow the user to upload a document, select their model of choice and pose queries that they have on their document.

5 CONCLUSION

5.1 Summary

In conclusion, GuruDocs presents a solution to a common workplace challenge faced by employees dealing with policy and/or regulatory documents. By leveraging RAG and LLMs, GuruDocs, provides a user-friendly and efficient platform for querying documents, ultimately enhancing the overall user experience and productivity.

5.2 Challenges and Limitations

One of the biggest challenges in GuruDocs is the compute resources required to perform the document question and answering. As we did not leverage on LLMs that run on the cloud (e.g. OpenAI's GPT4), we are dependent on the local compute resources to run the workflow. We have noticed that the embedding and query durations are significantly higher on laptops that have lower number of CPU cores and GPU memory, as compared to more powerful workstations that have a larger number of CPU cores and GPU memory.

Another limitation of GuruDocs currently is that it only supports Linux/Ubuntu Operating System (OS). This is because Ubuntu was the OS that was used to develop the product and there are certain Linux specific deployment methods used such as shell scripts for running the docker compose, which would not work on Windows or macOS.

The final limitation of GuruDocs is that the team has only tested it for English language and will not work for multi-lingual document question and answering. This could be a possible feature for future recommendations.

5.3 Future Recommendations

Multi-Modal Capabilities. For future implementations, we can explore expanding the capabilities of GuruDocs beyond text-based documents to incorporate multi-modal understanding. Our current implementation only ingests texts from documents, which creates limitations when documents contain images or tabular data as GuruDocs would lack data extraction accuracy and coverage of diverse content formats. By integrating technologies such as computer vision for image analysis, optical character recognition (OCR) for text extraction from images, and support for other document or image formats, GuruDocs can achieve a more comprehensive analysis of documents. This expansion not only enriches content analysis by recognizing visual elements like graphs, tabular data and identifying key information in images but also allows for a holistic view of the document's structure and insights.

User Authentication and Management. Second, we can introduce user authentication and management for future implementations. Our current implementation did not require user authentication as it was designed for local, on-premise installation on the user's laptop with no shared

account access. However to cater for robustness in future implementations on Cloud, user authentication and management will be needed. This would be crucial for ensuring the security and privacy of user data. By incorporating features such as secure login, role-based access control, and encryption of sensitive data, the GuruDocs can prevent unauthorized access to confidential information and mitigate the risk of data leakage. This recommendation aims to establish trust and confidence among users by prioritizing data security and privacy.

Trust and Safety Guardrails. Third, future implementations of GuruDocs can include trust and safety guardrails which are important in protecting users from potential risks such as malicious content or misleading information. While RAG with LLM applications can have fewer instances of model hallucinations as the context is provided for the LLM to generate completions, GuruDocs can still proactively pre-empt against possible hallucinations or malicious users and intents.

Knowledge Graph Visualisation. Fourth, we can integrate GuruDocs with knowledge graph representation libraries such as RAGxplorer (Chua, 2024). This would allow the user to better visualise the vector queries and understand which vector chunks were identified to be more similar to their user query. Users are also able to hover over the vector chunks to view the corresponding text of the chunks as well. Figure 18 below shows an example of RAGxplorer's visualization, extracted from their GitHub.

Visualise which chunks are most relevant to your query.



Figure 18 Example of RAGxplorer's visualization. Users are able to view the vector representation of their query together with that of the vector chunks from the document, and easily identify which chunks are similar to their query.

Customisable Configurations. Finally, further customisation configurations can be provided for the user to allow them to have more options such as choosing the embedding model to use, chunk size, etc. Our current implementation of GuruDocs only has one configurable parameter (model option) as we designed it with the intention of allowing users without generative AI or RAG knowledge to be able to use our product. As users become increasingly familiar with generative AI and concepts of RAG, customisation of configurations would allow more savvy users to explore different parameters on their document question and answering results.

6 REFERENCES

- ChatDOC. (2024, February 01). *Chat with documents. Get instant answers with cited sources.* Retrieved from ChatDOC: https://chatdoc.com/?src=google&gad_source=1&gclid=CjwKCAjw88yxBhBWEiwA7cm6pXQzoLETbyT-mLIPEY5hZnX_n2awLkADsKAHBItxH3QU4Fy0D_d6ExoC9D8QAvD_BwE
- ChatDOC. (2024, February 01). *OpenAI introduces GPTs, But ChatDOC Still Excels in PDF Chat.* Retrieved from ChatDOC: <https://chatdoc.com/blog/openai-introduces-gpts-but-chatdoc-still-excels-in-pdf-chat/>
- Chua, G. (2024, March 01). *RAGxplorer: Open-source tool to visualise your RAG* 🌐. Retrieved from GitHub: <https://github.com/gabrielchua/RAGxplorer>
- DeepLearning.AI. (2024, February 01). *LangChain for LLM Application Development.* Retrieved from DeepLearning.AI: <https://www.deeplearning.ai/short-courses/langchain-for-lm-application-development/>
- DeepLearning.AI. (2024, Feburary 01). *LangChain: Chat with Your Data.* Retrieved from DeepLearning.AI: <https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/>
- DocsBot AI. (2024, February 01). *Custom chatbots from your documentation.* Retrieved from DocsBot AI: <https://docsbot.ai/#features>
- Hypotenuse AI. (2024, February 01). *Free AI Chat - Hypotenuse AI.* Retrieved from Hypotenuse AI: https://www.hypotenuse.ai/ai-chat?utm_source=google&utm_medium=cpc&utm_campaign=hypo-brand_search_eng_nov23&gclid=CjwKCAjw88yxBhBWEiwA7cm6pQeQyvnMyN0tWBl8Ze68gitUVVsNjbpSv65eeDRmHivmukoMZ060ZBoCNU4QAvD_BwE
- LangChain. (2024, Mar 01). *Embedding models.* Retrieved from LangChain: https://python.langchain.com/docs/integrations/text_embedding/
- LangChain. (2024, Feb 01). *Introduction.* Retrieved from LangChain: https://python.langchain.com/docs/get_started/introduction
- OpenAI. (2024, February 01). *Pricing.* Retrieved from OpenAI: <https://openai.com/api/pricing>
- Ragas. (2024, March 01). *Introduction.* Retrieved from Ragas: <https://docs.ragas.io/en/stable/>
- Sojasingarayar, A. (2024, Mar 01). *Summarization with LangChain.* Retrieved from Medium: <https://medium.com/@abonia/summarization-with-langchain-b3d83c030889>
- Teo, S. (2024, Apr 1). *How I Won Singapore's GPT-4 Prompt Engineering Competition.* Retrieved from Medium: <https://medium.com/@mrumair384/how-i-won-singapores-gpt-4-prompt-engineering-competition-361253d3b14a>
- Yang, S. (2024, Mar 01). *4 Ways to Do Question Answering in LangChain.* Retrieved from Medium: <https://towardsdatascience.com/4-ways-of-question-answering-in-langchain-188c6707cc5a>

7 APPENDICES

Appendix 7.1: Summary Results

Summarization Technique	Summary Results
Stuff	<p>Personal information about employees, including address and phone number, is confidential and will not be shared outside the company without authorization.</p> <p>Employees are expected to dress appropriately for work, in smart business attire.</p> <p>The company does not accept responsibility for damage or loss of personal property on its premises.</p> <p>Loans or loan guarantees are not provided to employees.</p> <p>All employees must present for work sober and without any prohibited substances, including alcohol and non-prescription drugs.</p> <p>Outside committees/boards requests by employees must be approved by Adobe's Legal Department.</p> <p>The company is committed to providing opportunities for employee development and career progression.</p>
Map_Reduce	<p>The text is an employee handbook for Adobe Singapore. It covers various aspects of employment, including job roles and responsibilities, work hours and breaks, performance management, communication channels, and policies on confidentiality, intellectual property, and non-discrimination.</p> <p>The handbook also outlines the company's expectations for employees in terms of conduct and behavior, and provides information on benefits, compensation, and training opportunities. Some important details included in the handbook are:</p> <ul style="list-style-type: none">* Job roles and responsibilities for various positions within the company* Expectations for employee conduct and behavior both in and out of the workplace* Policies on confidentiality and intellectual property* Procedures for reporting and addressing any issues or concerns* Information on benefits, compensation, and training opportunities available to employees.
Refine	<p>Based on the new context provided, here is a refined summary of Adobe's policies and benefits for its employees:</p> <p>Adobe prioritizes creating a healthy and productive work environment for all employees, with a focus on innovation, involvement, and people as the greatest asset. The company has established clear policies and procedures that are transparent and fair, including licenses, maternity/paternity leave, adoption leave, family leave, medical leave, and other benefits. Adobe also has a set of core values that are at the heart of its culture, such as authenticity, innovation, involvement, and people as the greatest asset.</p> <p>In addition to the existing summary, here are some additional points that can be added:</p> <ul style="list-style-type: none">* Adobe provides a medical check-up benefit for employees only, which includes a preventive health check-up (body screening) not for the medical check-up for the treatment of specific illness. Family members are not eligible for this medical check-up benefit.* Adobe also provides vision reimbursement for employees only, which covers preventive eye exam, prescribed glasses/lens and contact lens. Exclusions include cosmetic surgery, laser surgery, eye surgery that requires specialist care, contact lens solution, sunglasses, etc. Family members are not eligible for this vision reimbursement benefit.

	<p>* Adobe has a business travel insurance policy which provides employees with three times their salary, up to a maximum of US\$500,000. The program pays benefits if an employee dies or becomes disabled as a result of an accident while traveling on business.</p> <p>In terms of responsibilities, employees are responsible for immediately notifying their manager or HR if they experience harassing or discriminatory behavior and/or are aware of any such behavior by any individual. Managers are responsible for making employees aware of the policy and preventing harassment from occurring, while HR is responsible for undertaking effective, thorough, and objective investigations of any harassment allegations and handling any appropriate disciplinary or remedial action in respect of a complaint.</p> <p>Adobe takes harassment and discrimination seriously and has procedures in place to handle any incidents of such behavior. Any employee or contractor who engages in conduct contrary to this policy will be disciplined, and serious cases of discrimination, harassment or victimisation will result in termination of employment. There are no exceptions to this policy, and employees are encouraged to voice grievances connected with their work to obtain a response from Adobe. However, the abusive use of this policy, such as raising unfounded allegations with malicious intent, will be treated as a serious disciplinary matter.</p> <p>Overall, Adobe prioritizes creating a healthy and productive work environment for its employees while also providing a range of benefits and policies to support their well-being and professional growth.</p> <p>Please let me know if you would like me to refine the summary further or if there is anything else I can help with.</p>
--	---

Appendix 7.2: Prompt Templates

Type of Framework	Basic Question-answering Framework	CO-STAR Prompt Framework
Prompt Template	You are an AI assistant for question-answering tasks. Use the following pieces of retrieved context to answer the question. If you don't know the answer, just say that you don't know. Use three sentences maximum and keep the answer concise.	<p># Context #</p> <p>You are an AI assistant tasked to efficiently query and retrieve information from staff documents, including employee handbooks, policies, and regulations</p> <p># Objective #</p> <p>Retrieve relevant information and present it in a readable format, improving overall information retrieval efficiency for employees.</p> <p># Style #</p> <p>The writing style should be clear, concise, and professional, aligning with the tone and style of our company's official documents. The language should be easily understandable by employees across all levels of the organization.</p>

		<p># Tone # Informative and authoritative, reflecting the importance and seriousness of the information contained in the staff documents. It should also be user-friendly and helpful, ensuring employees feel supported in their information retrieval efforts.</p> <p># Audience # The response is intended for internal employees of our company who need to access and understand staff documents, including new hires, current employees, and managers. The information should be tailored to the understanding and expertise of the audience.</p> <p># Response # The output format should be user-friendly and accessible, providing information in a structured manner that is easy to read and navigate. Consider using bullet points, summaries, and links to relevant sections for quick reference.</p>
--	--	---

Appendix 7.3: GuruDocs Evaluation Ground Truth and Results

Refer to separate Excel attachment.

Appendix 7.4: Installation and User Guide

Note: GuruDocs was developed on a workstation with the following specifications: Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz, 72-cores and NVIDIA Quadro RTX 8000 GPU. Do note that there would be significant latency for both embedding and querying documents for machines with lower specifications.

Installation Procedure:

1. Install Ollama, see download instructions at <https://ollama.com/download>
2. Clone GuruDocs repository

```
git clone
cd GuruDocs/
```

3. Download LLM models from Ollama

```
ollama pull mistral  
ollama pull llama2
```

Once you have completed the above steps, you can choose to run the application using either Docker containers or run it locally.

A7.4.1 Running GuruDocs using Docker Container

Prerequisites:

1. Ensure that you have **Docker engine** installed. If not, please follow instructions from <https://docs.docker.com/engine/install/>
2. **Nvidia container toolkit** to run Docker images with GPU. Please follow instructions from <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html> if you do not have it installed.

Once you have the above installed, you can proceed to build the Docker image

Building Docker image

1. You can build the Docker images using the shell script that we have prepared.

```
chmod +x ./docker_build.sh  
./docker_build.sh
```

2. Once the docker images have been built, spin up the Docker containers using the shell script that we have provided as well. You can spin down the containers using the stop command as well.

```
chmod +x ./gurudocs.sh  
./gurudocs.sh start  
./gurudocs.sh stop
```

3. After you have started the containers with the shell script, open your browser and go to <http://localhost:3001>. The page will load once the containers are running.

A7.4.2 Running GuruDocs using Local Installation

Requirements:

- Python 3.10
- NodeJS v18.20.1 (recommend to use nvm install)

```
#                                     Install          NVM  
sudo          apt           install        curl  
curl  
https://raw.githubusercontent.com/creationix/nvm/master/install.sh
```

```
h                                |                                bash
#                               Refresh                                terminal
source                           ~/.bashrc
#
#                               Verify                                Installation
nvm                                --version
#
#           Install                                NodeJS      v18.20.1
nvm install 18.20.1
```

Installation and run:

1. Create your conda environment (or venv)

```
conda create -n gurudocs python=3.10 -y
conda activate gurudocs
pip install -r requirements.txt
```

2. Start the application, both frontend and backend

```
python main.py
cd frontend/
npm start
```

3. Once the frontend has started, your web browser should automatically open with the address <http://localhost:3001>

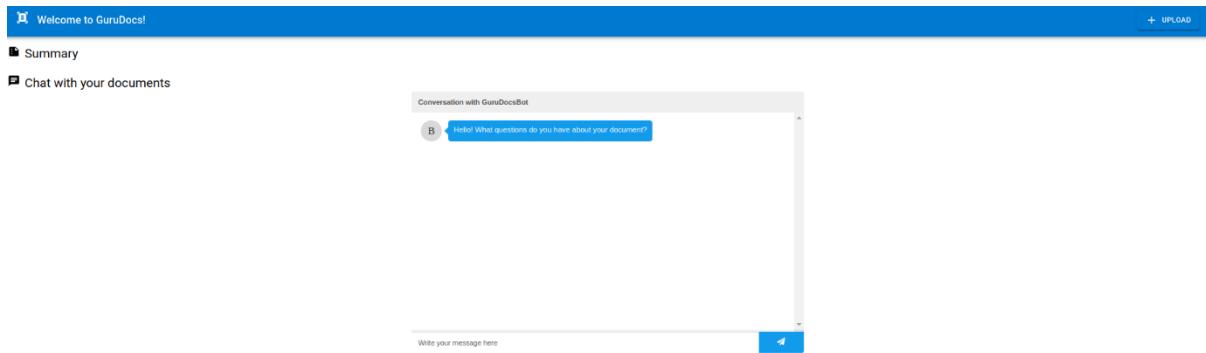
Troubleshoot

If you face issues with starting the frontend, we recommend removing all node modules and installing them again. You can do so using the following code. Once you have completed this, run the above steps again.

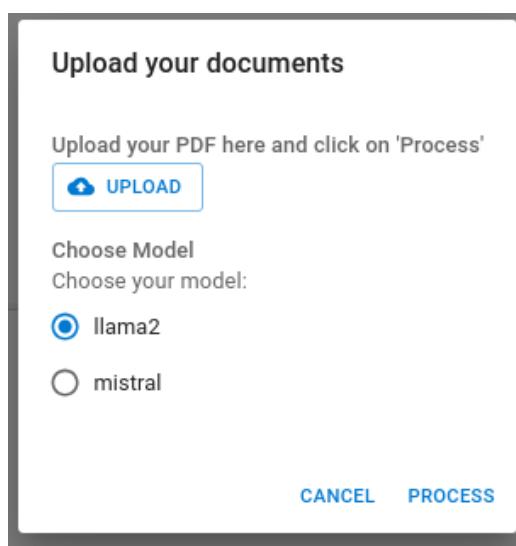
```
cd frontend/
rm -rf node_modules
npm install
```

A7.4.3 How to use GuruDocs

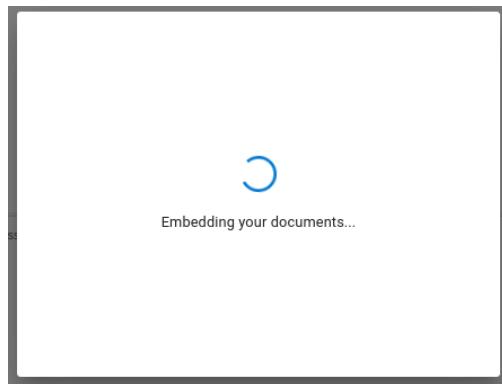
1. Once your application has started successfully, you should see the main page.



2. Click the “Upload” button on the top right hand corner and a Dialog will pop up
3. In the Dialog window, you can:
 - a. Upload the document (PDF) that you want to query by clicking the “Upload” button. Once you have selected your document, the document name should appear below the “Upload” button.
 - b. Select the LLM you would like to use in the RAG process. The default option is “llama2”
 - c. Once ready, click on “Process”



4. After you have clicked on “Process” the document chunking and embedding will begin. You will see a spinner window to let you know the process. You can either close this window by clicking anywhere outside of the Dialog box, or you can wait for the process to complete and it will automatically close when done.
 - a. Do wait for the green toast message in Step 5 below before you start chatting with GuruDocs. Otherwise GuruDocs will reply “Please upload a document or wait for the document processing to be completed.”



5. You will see a toast message in green once the document has been successfully embedded. This means that you can start chatting with GuruDocs on your queries! In the meantime, the document summary will be generated in the background and you will see the spinner in the "Summary" section.

6. To start chatting with GuruDocs, simply type your question into the chatbot and hit "Enter".

Appendix 7.5: Mapped System Functionalities to Intelligent Reasoning System Courses

Course	Mapped Functionalities
Machine Reasoning	<ul style="list-style-type: none">• <u>Knowledge Search</u>: To find and retrieve relevant information sought by the user
Reasoning Systems	<ul style="list-style-type: none">• <u>Knowledge based reasoning</u>: Understanding, interpreting and synthesizing information to provide meaningful responses to user queries

Cognitive Systems	<ul style="list-style-type: none">• <u>Natural Language Understanding</u>: embedding of document text chunks• <u>Natural Language Generation</u>: generating coherent answers from extracted content based on user's semantic query

Appendix 7.6: Project Proposal

Refer to separate PDF attachment in ZIP file.

Appendix 7.7: Individual Project Report

Project Team Member 1: Alvin Wong Ann Ying

1) Personal Contribution

- Developed project idea in collaboration with the team.
- Conducted literature review to inform market research on similar products.
- Annotated data to create ground truth datasets.
- Implemented sentence-level chunking as a preprocessing step, including text cleaning.
- Established an end-to-end evaluation pipeline.
 - Utilized the T5 model to generate 230 question-and-answer pairs.
 - Implemented a customized evaluation functionality based on modified RAGAs.
 - Evaluated performance across various experiment permutations through multiple runs, with detailed results documented in an Excel sheet.
 - Created an automated script for batch evaluation execution.

2) Most valuable knowledge/skills acquired

For me, acquiring expertise in evaluating Large Language Models (LLMs) using metrics like RAGAs equips me to assess their performance accurately. As the evaluator of Large Language Models (LLMs), I've gained valuable insights into deploying end-to-end solutions, from document uploading to obtaining RAGAs scores. This comprehensive understanding allows me to oversee the entire process, ensuring seamless flow and effective evaluation. As an AI practitioner in the LLM domain, these skills are invaluable, providing a solid foundation for effectively working with LLMs and tackling natural language processing tasks with confidence. Additionally, mastering the assessment techniques and steps involved in dataset creation post-project execution enhances my ability to prepare high-quality datasets crucial for accurate evaluation. Recognizing the impact of dataset quality on model performance further enables me to optimize results effectively.

3) How to apply knowledge/skills acquired in your workplaces

The expertise gained in preparing high-quality datasets for evaluation can be applied across various multimodal projects. Additionally, understanding the development of GuruDocs enables me to address the gaps in "interaction as an explanation." This approach focuses on providing an interactive Q&A version to users, targeting them in a sequential manner and tackling each part with interpretation. This methodology is valuable not only for vision AI tasks but also for other projects where detailed explanations may not be necessary upfront, but an interactive and interpretive approach is preferred.

Project Team Member 2: Brandon Chua Hong Huei

1) Personal Contribution

During the GuruDocs project, I assisted in the conceptualization of the project idea with Si Ci by developing a simple proof-of-concept to validate its feasibility. Upon the successful validation, I conducted in-depth research into various modules of the solution, including the database architecture, Large Language Models (LLMs), types of embeddings, document splitting methodologies (chunking techniques), retrieval techniques, summarization techniques, and prompt templates. Additionally, I took charge of developing the RAG model from scratch, ensuring its seamless integration with other components of the solution. Furthermore, I actively contributed to the evaluation of the solution, refining it based on feedback and performance metrics.

2) Most valuable knowledge/skills acquired

The most valuable takeaway being the knowledge of how to build a Retrieval-Augmented Generation (RAG) solution. This hands-on experience has not only deepened my understanding of NLP concepts but has also honed my skills in using the LangChain framework, which is instrumental in developing such solutions. Additionally, I have gained valuable insights into the complexities of document processing and information retrieval, which have broadened my skill set as an AI practitioner.

3) How to apply knowledge/skills acquired in your workplaces

The knowledge and skills acquired are highly applicable in my current workplace as an AI Engineer. The trend in AI is shifting towards large multi-modal models and intelligent agents, and my experience with LangChain and RAG solutions positions me well to contribute meaningfully to such projects. My understanding of these advanced concepts acts as a stepping stone for me to propose and explore new use cases in my work, thereby enhancing the capabilities of our AI systems and driving innovation in our organization.

Project Team Member 3: Ong Si Ci**1) Personal Contribution**

- Conceptualised project idea and implementation plans
- Coordinated and delegated project tasks to team members
- Design of end-to-end architecture of GuruDocs
- Developed and implemented frontend (initial Streamlit prototype and final React implementation)
- Developed and implemented backend FastAPI endpoints
- Dockerisation of frontend and backend. Prepared docker compose and shell scripts for easy installation and starting/stopping the application.
- Initial implementation of RAG model with Brandon

2) Most valuable knowledge/skills acquired

Through the development of GuruDocs, I gained valuable knowledge on RAG and LLM workflows. This included taking courses on DeepLearning.AI and building prototypes to validate the project idea. The project also allowed me to have hands-on experience working with LangChain and Ollama model deployment, providing me with a deeper understanding of the strengths and limitations of these frameworks. Additionally, I did not have any experience in frontend development prior to this, and managed to pick up ReactJS through this project. This allowed me to expand my repertoire of technical skills from the existing AI modelling and backend development, to include frontend skills as well.

3) How to apply knowledge/skills acquired in your workplaces

The skills and knowledge acquired during the development of GuruDocs are directly applicable to my role as an AI engineer. The expertise gained in RAG and LLM workflows not just equips me to design and implement RAG solutions, but also allows me to have a better appreciation of how agentic AI could work using LangChain framework. These would be useful in developing AI products to enhance operational use cases. Furthermore, basic frontend development skills would allow me to be able to build simple prototypes for proof of concepts instead of having to rely on a frontend engineer.