



**TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND  
TECHNOLOGY**

**FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY**  
**Session 202501**

**BMCS2123 NATURAL LANGUAGE PROCESSING**

		
<p><i>Saw Xue Yi</i> 24WMR08887 Chatbot Main Implementation &amp; Documentation Writing</p>	<p><i>Cha Ming En</i> 24WMR08850 Chatbot UI Design &amp; Database Integration &amp; Documentation Writing</p>	<p><i>Hea Hui Min</i> 24WMR08865 Text Extraction &amp; Text Preprocessing &amp; Documentation Writing</p>

Title	:	Interactive Chat Application for Conversations with PDF or Document Content Using a Language Model (LLM)
Tutorial Class	:	Group 4
Tutor Name	:	Dr Noor Aida Binti Husaini
Submission Date	:	5/5/2025

# Table of Contents

<b>1. Introduction</b>	<b>3</b>
1.1 Project Aim	3
1.2 Project Objectives	3
1.3 Existing Problems	4
1.4 Proposed Solution	4
1.5 Research Questions	5
1.6 Research Design	5
1.7 Expected Outcome	6
<b>2. Research Background</b>	<b>7</b>
2.1 Large Language Model	7
2.2 Trending Large Language Models (LLMs)	8
2.3 Techniques for Retrieval-Augmented Generation (RAG)	9
2.4 OpenRouter Large Language Model API Key	10
2.5 What is a File Chatbot?	10
2.6 Existing File Chatbots in the Market	11
2.7 General methodology for a text-based conversational agent	12
2.8 Theoretical and Research Literature	13
2.9 Peer-Reviewed Literature	14
<b>3. Methodology</b>	<b>18</b>
3.1 System Design	18
3.2 Key Components	20
3.2.1 Document Processing Module	20
3.2.2 Embedding Generation	21
3.2.3 Vector Database	22
3.2.4 LLM Integration	23
3.2.5 Export Chat Record	24
3.2.6 Preview Uploaded File Content	25
3.3 Implementation Details	25
3.3.1 Backend	25
3.3.2 Frontend	26
3.3.3 Chat History Management	27
<b>4. Results</b>	<b>28</b>
4.1 User Interface	28
4.2 Accepting Multiple File Types (PDF, DOCS, TXT)	30
4.3 Validation of Responses to File-Related Questions (Summarization and Translation)	32
4.4 Answering General or Unrelated Questions	38
4.5 Handling Spelling Mistakes or Typo	43
4.6 Handling Mixed Language	46

4.8 Error Handling Performance	48
4.9 Chat History Management	49
4.10 Export Chat Record	51
4.11 Uploaded File Preview	53
4.12 User Feedback and Satisfaction	55
4.14 Test Case	58
<b>5. Discussion and Conclusion</b>	<b>59</b>
5.1 Achievement of Objectives	59
5.2 The Significance of Result	60
5.3 Limitations	61
5.4 Future Improvements	62
<b>References</b>	<b>63</b>

# **1. Introduction**

## **1.1 Project Aim**

The purpose of this project is for an interactive chat application where users can communicate with the content of a PDF or document. By introducing a Language Model (LLM), we will model context-aware and intelligent interactions, changing presentation from a static document into a dynamic conversation partner. It will be able to upload files, ask questions, ask for summaries, extract specific information, and get accurate, NLP-powered responses based on the content of the document.

It provides an environment that fuses the traditional text access and modern AI-based interaction so that users can easily engage a document by manipulating a decomposition of it. The chatbot will work towards reliable and timely responsiveness through rigorous validation and integration with the LLM model rather than just reading the document passively. Combining Natural Language Processing (NLP) with usability is practical; this project proves that LLM can transform the way people interact and understand texts.

## **1.2 Project Objectives**

- Create an intuitive interface that facilitates seamless uploads and interactions with PDFs and other document formats.
- Enable users to upload PDF or a document file and accurately extract the content.
- Allow users to ask questions, request summaries, or retrieve specific information from uploaded PDF or document.
- Ensure users receive accurate and contextually relevant responses based on the document content.
- Chatbot able to generate summaries of uploaded PDF or document.
- Implement translation capabilities to enhance user accessibility.
- Implement file preview feature for reference.
- Enable users to track their chat history.
- Achieve low-latency responses for real-time interactions.

- Implement error handling for edge cases, such as unsupported formats, unreadable text, file size limitations.

## 1.3 Existing Problems

Nowadays, there are many challenges when working with documents, as non-native speakers struggle with untranslated content, restricting access to critical information across languages, while users waste time sifting through lengthy texts to find relevant details, resulting in inefficient reading and poor comprehension; busy professionals and students often lack the time to read entire documents thoroughly, and traditional document tools offer only basic keyword searches, missing contextual understanding and related concepts, all worsened by one-way interaction systems that prevent users from asking follow-up questions or getting clarifications.

## 1.4 Proposed Solution

Since sifting through dense documents can seem like finding a needle in a haystack, especially when language barriers, time limitations, and generic tools make the process feel even more daunting, our solution turns this awful experience around by transforming static documents into interactive conversations, where a user can ask questions, get summaries, have a conversation, and have the answers positioned in relevant parts of the document. Unlike existing systems like Paperpal, which is limited to PDF uploads and lacking translation features (*Paperpal, 2025*), our file chatbot redefines accessibility by supporting PDF, DOCX, and TXT, while integrating AI-powered translation and context-aware insights, catering to both students skimming for key ideas and professionals conducting deep analysis. While incorporating the core functionality of existing systems like Paperpal, we've significantly enhanced the document interaction experience with innovative features that go beyond basic text processing. Our solution not only matches but surpasses conventional capabilities by delivering clear, accurate, and intuitive document understanding through natural conversation. Each interaction is intelligently preserved for future reference, effectively bridging the gap between static document formats and the dynamic, conversational interfaces that define modern digital experiences.

## 1.5 Research Questions

The research questions that this project aims to address are:

1. What are the key UI design principles for creating a user-friendly file chatbot?
2. What are the essential features needed to make a file chatbot compatible with various document types, such as PDF, DOCX, and TXT?
3. How can a file chatbot be used to improve the accessibility of research information to researchers and the public?
4. How effectively can the chatbot handle errors, unexpected inputs, or ambiguous queries related to file interactions?
5. What methods can be used to enhance the chatbot's ability to summarize and extract key insights from lengthy documents?
6. How can the file chatbot maintain robustness and reliability when confronted with mixed-language inputs, ambiguous queries, or incomplete user inputs?

## 1.6 Research Design

The research design for this project is centered around two main components: gathering user feedback and calculating cosine similarity scores to evaluate the interactive chat application's performance. User feedback will be collected through google form surveys, aiming to capture qualitative insights into the user experience, including the application's ease of use, the relevance and accuracy of the chatbot's responses, and overall satisfaction. This qualitative data will provide valuable insights into user preferences and potential areas for improvement.

Quantitatively, the project will measure the system's performance by calculating cosine similarity scores between the chatbot's responses and reference responses, such as those provided by ChatGPT. This metric will objectively assess the relevance and quality of the chatbot's outputs in relation to the document content. The cosine similarity scores will be used to compare different scenarios, including variations in preprocessing and the use of different language models like Deepseek R1 and Deepseek V3 0324.

By integrating user feedback with quantitative performance metrics, the research design ensures a balanced assessment of the chat application's effectiveness. The data analysis will

involve thematic analysis of user comments to identify common themes and patterns, and statistical analysis of cosine similarity scores to draw meaningful conclusions about the system's performance. This structured approach will help refine the chat application, enhancing its functionality and transforming static documents into dynamic conversational partners that better meet user needs.

## 1.7 Expected Outcome

Our interactive chat application is expected to deliver the following outcomes:

### 1. Enhanced Document Interaction

It offers users a conversational experience for reading PDFs and other document types. Instead of the plain PDF text information, users can interact with the document via Q&A, summarization, information extraction, etc.

### 2. Accessibility & Efficiency

By generating instant summaries, accurate answers, and translation services, the chatbot aims to eliminate time spent on manually finding information and enhance content for non-native speakers and people always in a hurry.

### 3. Accurate & Context-Aware Responses

The chatbot is expected to produce more reliable, document-grounded responses, reducing hallucination effects and keeping the responses closer to the uploaded contents.

### 4. User-Friendly Interface

An intuitive UI will allow seamless file uploads, real-time interactions, and easy navigation through chat history, improving overall usability.

### 5. Robust Error Handling

The system will gracefully manage unsupported formats, unreadable text, and ambiguous queries, providing clear feedback to users when issues arise.

### 6. Real-Time Performance

Optimized processing will ensure low-latency responses, making the chatbot suitable for quick, on-demand document analysis.

## 2. Research Background

### 2.1 Large Language Model

Natural language processing (NLP) has emerged as one of the most challenging and promising research fields as artificial intelligence (AI) technology continues to advance. Language modelling is NLP's core technique, with the goal of understanding, generating and processing human language. Most early language models rely on statistical methods, such as n-gram models, which use the probability of word occurrence to forecast, but have clear difficulties in processing long-range dependencies and contextual knowledge (*Minaee et al., 2024*).

With the advent of deep learning, recurrent neural networks (RNNs) and their variations, long short-term memory (LSTM), have begun to be used in language modeling considerably, boosting the ability to process time series and context. However these models continue to encounter hurdles in terms of training efficiency and global semantic capture. In 2017, Vaswani et al. developed the Transformer architecture, which achieved efficient modeling of the interaction between all words in a sequence using the self-attention mechanism, radically altering the design of language models. This architecture not only enhances the model's ability to handle long-range dependencies, but it also enables large-scale parallel processing (*Vaswani et al., 2017*).

Many breakthrough models have been introduced as a result of the emergence of Transformers. Google proposed BERT (Bidirectional Encoder Representations from Transformers) in 2018, which enables higher contextual understanding using bidirectional training technology (*Devlin et al., 2018*). OpenAI's GPT (Generative Pre-trained Transformer) series is notable for its generative language capabilities, particularly GPT-2 and GPT-3, which excel at few-shot learning and contextual reasoning (*Brown et al., 2020*). These models have set new performance records for a variety of NLP applications, including text comprehension, machine translation, summary creation, and dialogue systems.

Researchers also discovered that when the model size grows, including the number of parameters, training data, and computer resources, the language model's performance

improves steadily. This phenomenon is known as “scaling laws” (*Minaee et al., 2024*). This result spurred the research community to create larger scale models, including GPT- 4, PaLM and LLaMA.

Large language models offer a wide range of applications including intelligent customer service, content production, medical consultation, and program support, but they also carry significant hazards and ethical concerns. For example, LLMs can reproduce biased and misleading information in training data and can even be used to generate fraudulent content. Furthermore, because these models learn a vast amount of uncensored online material during the pre-training stage, ensuring the accuracy and dependability of the content they outputs has emerged as a pressing issue (*Chatbot | Definition, History, & Facts | Britannica, 2024*). As a result, research on model security, fairness and controllability has gained traction throughout time.

To summarize, the creation of massive language models not only advances artificial intelligence technology, but also opens up new avenues for language comprehension and generation. However, as we continue to investigate the promise of technology, we must also consider and respond to the social and ethical concerns it presents.

## **2.2 Trending Large Language Models (LLMs)**

The field of artificial intelligence has been transformed by the emergence of powerful Large Language Models (LLMs), with OpenAI's GPT, Google's Gemini, and DeepSeek's models leading the charge. These cutting-edge AI systems continue to evolve, offering increasingly sophisticated capabilities in natural language understanding and generation (*365mechanix. 2025, February 18*).

OpenAI's GPT series has set industry benchmarks, progressing from GPT-3 to the more advanced GPT-4, with specialized variants like GPT-4-turbo optimizing for speed and efficiency (*Sharma A., 2025*). Meanwhile, Google's Gemini represents a major leap in multimodal AI, with versions such as Gemini 1.0, Gemini 1.5, and the ultra-capable Gemini Pro and Gemini Ultra, designed to handle text, images, audio, video and even code seamlessly (*Guinness, H., 2025*).

On the other hand, DeepSeek, a rising star in the AI landscape, has introduced models like DeepSeek Chat v3.0324 and DeepSeek R1, which provide high-quality, open-access alternatives for developers and researchers (*Guinness, H., 2025*). These models excel in retrieval-augmented generation (RAG) tasks, making them ideal for applications requiring deep contextual understanding.

As these LLMs continue to advance, they are reshaping industries—from customer service automation to AI-assisted research—ushering in a new era of intelligent computing. The competition between GPT, Gemini, and DeepSeek ensures rapid innovation, promising even more powerful and accessible AI tools in the near future.

## 2.3 Techniques for Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) enhances AI-generated responses by combining information retrieval with text generation, ensuring more accurate and contextually relevant outputs. A key technique in RAG involves generating embeddings using Sentence Transformers, such as the efficient **distiluse-base-multilingual-cased-v2** model, a multilingual model provided by Sentence Transformers that maps sentences and paragraphs into a 512-dimensional dense vector space suitable for tasks such as clustering or semantic search (*distiluse-base-multilingual-cased-V2*). The AI Tool Documentations suggests that the model, which supports 50 languages and prioritizes embedding speed and accuracy, is ideal for semantic search and cross-language tasks (*Sentence Transformers documentation. n.d.*).

The generated embedding vector is saved in the vector database, where the system performs similarity searches between the user query and stored text fragments. Using the FAISS vector database's similarity search functionality, the system identifies and returns the most semantically relevant text passages. This mechanism ensures that the context obtained through user input is highly relevant, resulting in more accurate reference information for subsequent language model generation.

In order to generate high-quality responses, the retrieved information will be fed into powerful language models via OpenRouter API, which provides access to models like

DeepSeek Chat v3.0324 (DeepSeek v3 0324) and DeepSeek R1 (R1) for free. These models refine the retrieved content into coherent and precise answers. By integrating these components, Sentence Transformers for embeddings, similarity search for relevance ranking, and LLM model for generation, developers can build a robust RAG system that improves response accuracy while remaining efficient and scalable. This approach leverages cutting-edge AI techniques to deliver reliable and context-aware results.

## 2.4 OpenRouter Large Language Model API Key

In recent years, the rapid development of large language models (LLMs) has led to the emergence of numerous powerful models from different providers, each offering unique strengths in language understanding and generation. However, integrating and managing access to these models individually can be complex and time-consuming. OpenRouter addresses this challenge by serving as a unified API gateway that allows developers to access a wide variety of LLMs, such as those from OpenAI, Gemini, Deepseek, Anthropic, Cohere, and Mistral, through a single platform using one API key. This centralized approach simplifies model integration, enhances flexibility in model selection, and supports efficient experimentation across multiple models. By enabling dynamic switching between models based on user needs or application requirements, OpenRouter is particularly suitable for building intelligent chatbot systems that require scalable, diverse, and easily managed language model access.

## 2.5 What is a File Chatbot?

Traditional information query methods have increasingly given way to more interactive and intelligent dialogue systems as Natural Language Processing (NLP) and Large Language Models (LLMs) have advanced rapidly. File Chatbot is one such upcoming tool that may engage in real-time chat and answer queries based on the content of files supplied by users (*OpenAI, 2023*). This type of system integrates a language model's semantic comprehension capacity with information retrieval technology, significantly boosting user efficiency and experience in data discovery (*Minaee et al., 2024*).

A document-based chatbot's core technical process usually includes three main steps. First, the file uploaded by the user such as PDF, Word or TXT is formatted and text extracted, then the text data is segmented and vectorized, and stored in a vector database or semantic indexing system such as FAISS or Chroma. Finally, relevant paragraphs are matched according to a language model such as GPT or BERT combined with semantic retrieval technology (*Minaee et al., 2024*).

Document-based chatbots are widely utilized in a variety of settings, including enterprise knowledge management, instructional technology, assisted medical document reading, and legal text question and answer. For example, companies can use this system to swiftly access internal regulations and product manuals; students can utilize textbooks to consolidate learning outcomes, while attorneys and doctors can instantly search for technical clauses or case summaries (*Chatbot | Definition, History, & Facts | Britannica, 2024*).

Despite this, such systems confront a number of issues, including how to efficiently handle unstructured data, verify the correctness of generated responses and prevent producing misleading or biased information. Furthermore, research on the privacy protection of private documents and the controllability of model responses has gained prominence in academia and industry (*Minaee et al., 2024*).

## **2.6 Existing File Chatbots in the Market**

With maturation of huge language models and vector database technologies, the market has seen an increase in the number of file-based chatbots, offering users a more efficient knowledge query and document interpretation experience. Currently, such systems primarily use semantic search, contextual retrieval (RAG) and dialogue generation technologies to enable natural and accurate dialogue interactions based on document content (*Minaee et al., 2024*).

The more representative document-based chatbots now on the market are ChatGPT, which was launched by OpenAI and has the “Custom GPTs” function that allows users to upload PDF, DOCX, or TXT files and interact with the content. In addition, ChatPDF is a chatbot designed exclusively for PDF files. Users just upload files to ask inquiries and receive

succinct summaries or accurate paragraph citations responses (*ChatPDF - Chat with Any PDF!*, 2023). Another application, AskYourPDF, combines ChatGPT with a file storage system, supports many formats, and integrates with APIs. It is widely used in educational and business settings (*AskYourPDF: The Best PDF AI Chat App*, 2024).

In addition, several businesses have begun to include document-based chatbots into their internal knowledge management systems to assist employees in swiftly obtaining policies, operating manuals, or product information, thereby improving work efficiency. These systems typically use open source vector databases such as FAISS, Chroma, or Weaviate, and language generation engines like OpenAI or Hugging Face models (*Chroma*, 2024). This system, built on the RAG architecture, effectively combines the benefits of “accurate information retrieval” and “natural language response.”

Overall, these commercially available document-based chatbots demonstrate the utility of language models in real-word scenarios. However, the quality and traceability of generated content remain significant issues, particularly in industries such as education, medicine and regulation, which require exceptionally high information accuracy. As a result, creating systems that can provide accurate reference sources and dynamically update knowledge bases has become a priority for the future (*Minaee et al.*, 2024).

## **2.7 General methodology for a text-based conversational agent**

With the advancement of natural language processing technology, text-based conversational agent systems have emerged as a key application in intelligent systems, with widespread use in situations such as customer service, education, medical care and document question and answer including systems development methodologies often include multiple important technical modules, including as input processing, stage, the system will perform basic processing on the user’s natural language text input, such as word segmentation, named entity recognition, and grammatical analysis in the semantic understanding (NLU) stage, the system will convert natural language into structured information using technologies such as intent recognition and parameter filling (Chen et al., 2017). The second step, Dialogue Management, calculates the next action based on the context and conversation history. In the

generation phase, huge language models (such as GPT or T5) are used to generate neutral and semantically consistent responses (Lewis et al., 2020). Finally, the system will execute post-response processing, such as grammatical correction, formatting, or sensitive word screening, before presenting the output to the user.

If applied to a document-based dialogue system, this methodology will be extended to the document processing module, such as converting PDF or Word into plain text segmentation, vectorization, semantic indexing, and so on as well as using retrieval-augmented generation (RAG) technology to improve the system's answer's accuracy and contextual adaptability (OpenAI, 2023). In general, modern text-based dialogue systems use a hybrid architecture that combines generative models and retrieval, allowing them to handle both open domain dialogues and specific task-based queries, making them an important component of intelligent application systems (Jurafsky & Martin, 2023).

## 2.8 Theoretical and Research Literature

The research history of dialogue dates back to the 1960s. The first dialogue systems, such as ELIZA, used a simple rule matching mechanism to facilitate human-computer interaction. However, when natural language processing (NLP) technology advanced, modern dialogue systems shifted to model architectures based on machine learning and deep learning (*Jurafsky & Martin, 2023*). On the theoretical level, conversation systems primarily rely on language model theory, pragmatics, and context modeling, and have produced a number of dialogue management algorithms, including finite state machines (FSM), information state updates (ISU), and reinforcement learning.

Recent research focuses on generative pre-trained language models (such as the GPT series), which can produce semantically rich and contextually coherent responses when trained on vast amounts of text data (*Brown et al., 2020*). Furthermore, the RAG (Retrieval-Augmented Generation) architecture has become a widely used technology that combines knowledge retrieval and generation, making it ideal for developing document-based chatbots. It can increase response accuracy and traceability while keeping the language model's capabilities intact (*Lewis et al., 2020*).

Many studies have shown that a hybrid architecture that combines retrieval and generation may effectively manage issues including open question-answering, knowledge-intensive discourse, and multi-round context maintenance (*Karpukhin et al., 2020*). For example, OpenAI's GPT model can not only generate natural content but also collaborate with external knowledge sources (custom documents, API's knowledge graphs and so on) to increase dialogue quality (*OpenAI, 2023*). Furthermore, for application scenarios in education and corporate information management platforms such as AskYourPDF, ChatPDF, and Custom GPT have proved the practicality and promise of document-based dialogue systems in real situations (*ChatPDF - Chat with Any PDF!, 2023; AskYourPDF: The Best PDF AI Chat App, 2024*).

## 2.9 Peer-Reviewed Literature

In today's information-saturated world, professionals across industries face a common challenge: quickly extracting relevant knowledge from growing mountains of documents. Enter file chatbots - sophisticated AI systems that are revolutionizing how we interact with PDFs, manuals, and reports. Unlike traditional chatbots limited to scripted responses, these advanced tools combine cutting-edge natural language processing with dynamic information retrieval to deliver context-aware answers drawn directly from source materials.

Recent advancements in artificial intelligence have enabled what was once science fiction - conversational interfaces that can comprehend complex documents as effectively as human experts. From aviation safety manuals to legal contracts, file chatbots are demonstrating remarkable capabilities while revealing important limitations that guide future development.

Early innovations in file chatbots emerged from information retrieval (IR) systems designed to parse unstructured documents. The DocChat framework (*DocChat n.d.*) pioneered this approach by combining IR with learning-to-rank models to select responses from document repositories. By extracting sentence-level candidates and ranking them using granular features, DocChat achieved a 15% improvement in response relevance compared to Q-A pair-based systems. Subsequent work by Jatmika et al. (*Sunu Jatmika n.d.*) introduced cognition-based document matching, evaluating Euclidean Distance, Cosine Similarity, and BERT embeddings for semantic alignment. Their experiments on 1,755 labeled queries revealed BERT's superiority in contextual understanding (89.2% accuracy) despite higher

computational costs, whereas Cosine Similarity offered optimal efficiency for real-time applications.

These methodologies address critical limitations in traditional chatbots, such as rigid response templates and insufficient contextual awareness. For instance, the Euclidean Distance method (*Sunu Jatmika n.d.*) excelled in speed but struggled with polysemy, while BERT's bidirectional attention mechanisms resolved ambiguities in aviation manuals (*XIN TAN. n.d.*). Such advancements underscore the importance of hybrid models that balance accuracy and computational overhead.

The integration of RAG frameworks has further enhanced file chatbots by coupling large language models (LLM) with dynamic data retrieval. Pokhrel et al. (*Sangita Pokhrel, 2021*) demonstrated a RAG-based chatbot that scrapes web content, generates vector embeddings, and performs semantic searches to answer user queries. Their system achieved an 84.5% F1 score in accuracy evaluations by combining BM25 and Dense Passage Retrieval (DPR) models, illustrating the benefits of multi-channel retrieval. Similarly, Tan et al. (*XIN TAN. n.d.*) implemented a "reader-retriever" pipeline for flight crew operating manuals (FCOM), where a BERT-based retriever reduced average response time to 2.3 seconds while maintaining 87% answer correctness.

RAG models mitigate the "static knowledge" problem of conventional LLM by incorporating up-to-date information from external sources. For example, the system described in (*Sangita Pokhrel, 2021*) dynamically updates its knowledge base through web scraping, ensuring responses reflect the latest website content. This capability is particularly valuable in domains like e-commerce, where product details and promotions change frequently. However, challenges persist in handling non-textual elements; (*XIN TAN. n.d.*) noted that 23% of FCOM queries involved tabular data, which existing NLP models often misinterpret.

Aviation has emerged as a critical testbed for file chatbots due to the complexity of operational manuals. The intelligent QA system developed by Tan et al. (*XIN TAN. n.d.*) processed 1,200 FCOM sections using a BERT-DPR retriever, achieving an 84.5% top-5 F1 score. The system's success hinged on preprocessing steps like entity recognition for aviation jargon (e.g., "V1 speed," "flap settings"), which improved semantic search precision by 31%.

Conversely, generic chatbots without domain adaptation failed to parse 68% of technical queries in the same dataset, highlighting the necessity of specialized training.

In customer experience management, file chatbots reduce reliance on human agents by resolving inquiries directly from policy documents. A study by Yan et al. (*DocChat n.d.*) deployed DocChat for IT support ticketing, where it resolved 72% of user issues using troubleshooting guides and FAQs. The framework's triggering mechanism filtered low-confidence responses, achieving a 92% user satisfaction rate. Similarly, (*Bouchra El Bakkouri , 2022, August 12*) documented a 40% reduction in escalations for banking chatbots that referenced updated compliance documents in real time.

Despite their advantages, file chatbots face scalability issues. Jatmika et al. (*Sunu Jatmika n.d.*) reported that BERT-based matching required  $4.2\times$  more processing time than Cosine Similarity, making it impractical for high-traffic applications. To address this, (*Sangita Pokhrel, 2021, August 20*) proposed lightweight embedding models like Sentence-BERT, which reduced latency by 57% without significant accuracy loss.

Most systems focus on textual data, neglecting tables, diagrams, and images prevalent in technical documents. (*XIN TAN. n.d.*) identified this as a major bottleneck, with only 11% of tabular queries answered correctly. Future solutions may require vision-language models like CLIP to parse multimodal content, though such integrations remain underexplored in peer-reviewed literature.

Next-generation RAG systems aim to incorporate continuous learning mechanisms. For instance, (*Sangita Pokhrel, 2021, August 20*) proposed reinforcement learning to refine retrieval strategies based on user feedback, potentially improving accuracy by 12–18%. Additionally, federated learning approaches could enable decentralized document updates across organizational silos without compromising data security.

Expanding file chatbots to non-English contexts presents both opportunities and challenges. While (*Sangita Pokhrel, 2021, August 20*) demonstrated preliminary success in Nepali and Hindi queries using mBERT, low-resource languages lack sufficient training data. Collaborative efforts to build multilingual corpora, as seen in the SQuAD 2.0 dataset, could bridge this gap.

File chatbots represent a paradigm shift in automated dialogue systems, offering unparalleled accuracy and adaptability in document-heavy domains. Peer-reviewed studies validate the efficacy of hybrid IR-NLP architectures, particularly RAG frameworks and cognition-based matching algorithms. However, persistent challenges in computational efficiency, multimodal processing, and multilingual support necessitate further innovation. As organizations increasingly adopt these systems, future research must prioritize real-time adaptability, cross-format compatibility, and ethical considerations in automated decision-making.

### 3. Methodology

#### 3.1 System Design

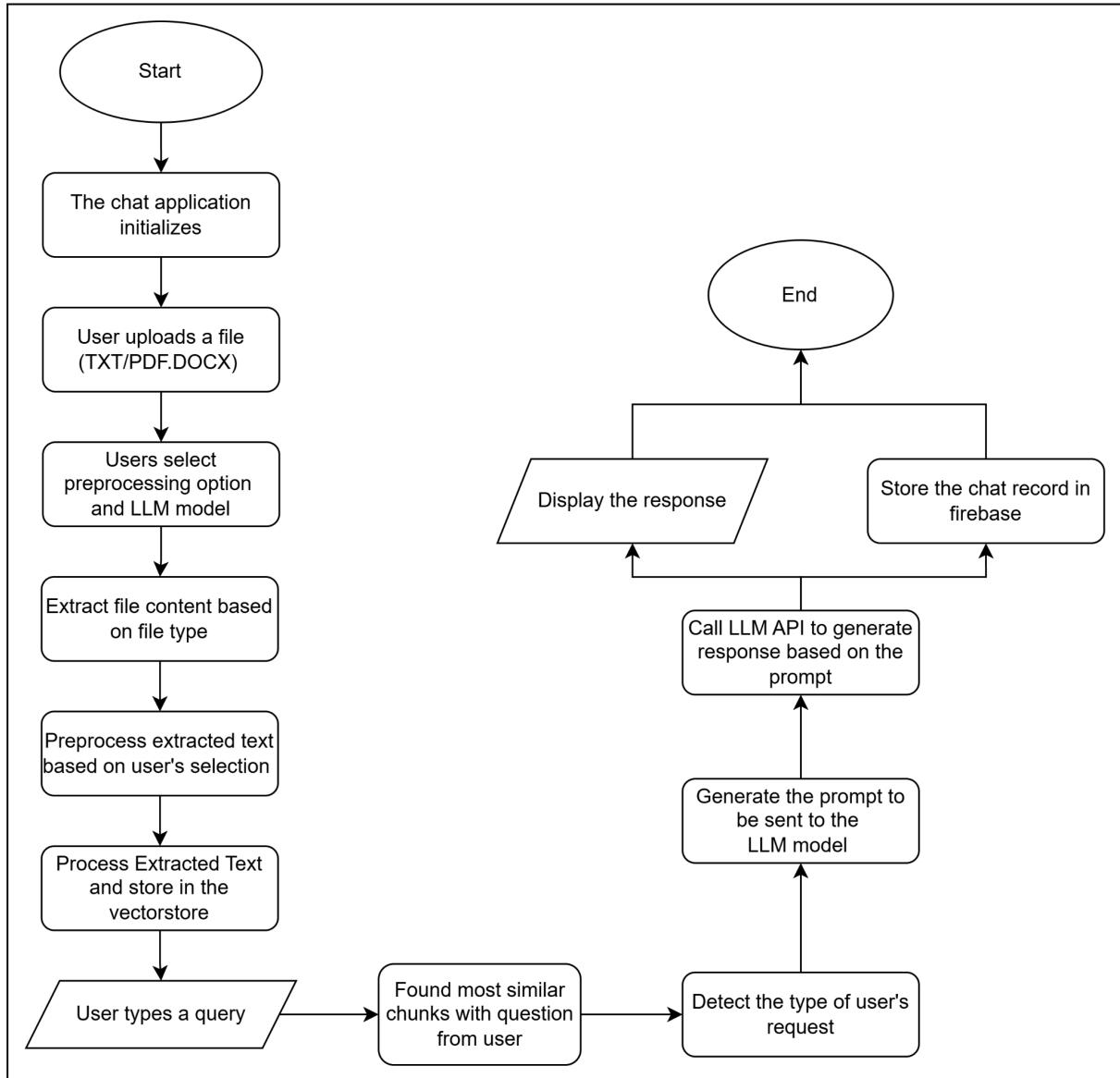
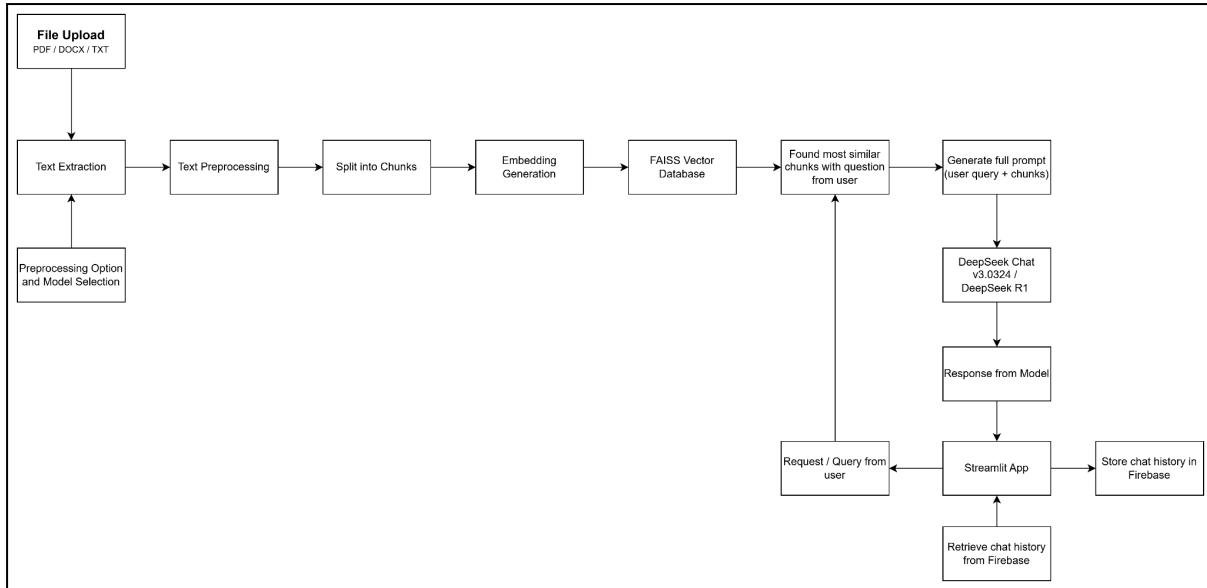


Figure 3.1 Flowchart for the basic chat feature



**Figure 3.2 Overall System Flow - Chatbot Architecture**

The diagram above illustrates the overall architecture of the chatbot system. When a new chat is initiated, a file is uploaded along with the user's selected preprocessing option and preferred LLM model. The system first extracts text from the uploaded file. Based on the selected preprocessing option, the corresponding preprocessing steps are then applied.

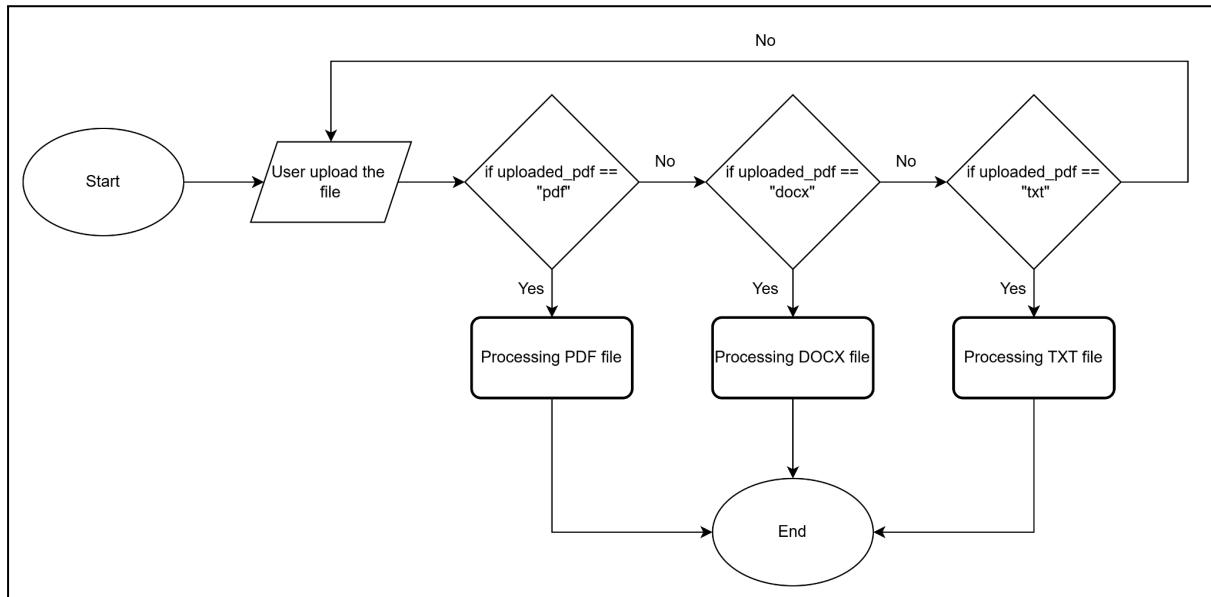
Next, the text is split into chunks and transformed into vector representations using the HuggingFace embedding model distiluse-base-multilingual-cased-v2. These embeddings are stored in a vector store. This initialization process occurs when a user uploads a file.

On the Streamlit frontend, when the user submits a query, the system retrieves the most relevant chunks from the vector store. It then constructs a complete prompt using the user query and the retrieved chunks. This prompt is sent to the selected LLM model which is chosen earlier by the user, via an API key. The generated response is then returned to Streamlit and displayed to the user. Meanwhile, the chat conversation is also stored in Firebase for persistence.

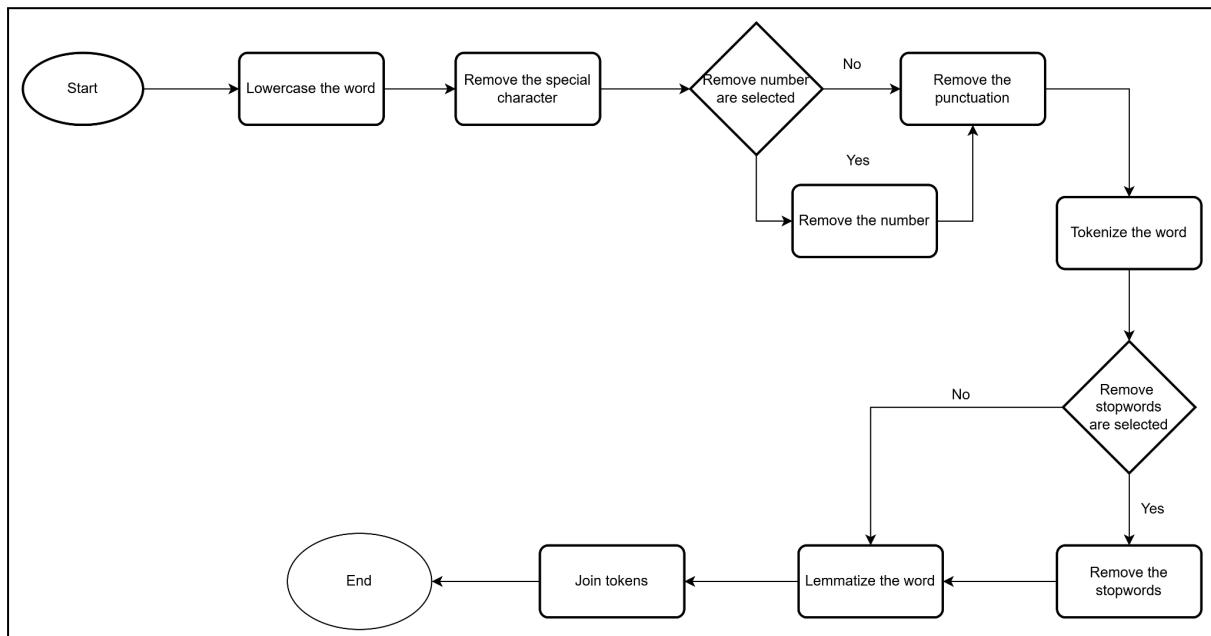
Upon Streamlit initialization, the system first retrieves any previous chat history from Firebase for previous chat displaying purposes.

## 3.2 Key Components

### 3.2.1 Document Processing Module



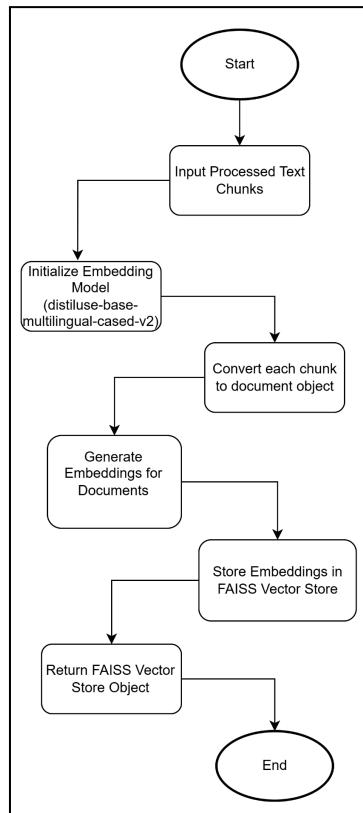
**Figure 3.3 Text Extraction Module from Uploaded File**



**Figure 3.4 Text Preprocessing Module**

This module extracts text from uploaded files and performs standardized preprocessing. First, the file extraction function can extract plain text content from PDF, Word (DOCX) and text (TXT) files. The system can automatically determine the file type and invoke the appropriate parsing method. Second, the text is preprocessed by converting it to lowercase, removing the special characters and punctuation, selectively removing numbers and stop words, while tokenizing for subsequent natural language processing tasks. In addition, this module considers various edge cases, such as files that cannot be read, unsupported formats, or files that are too large, and handles errors appropriately to ensure the system's robustness.

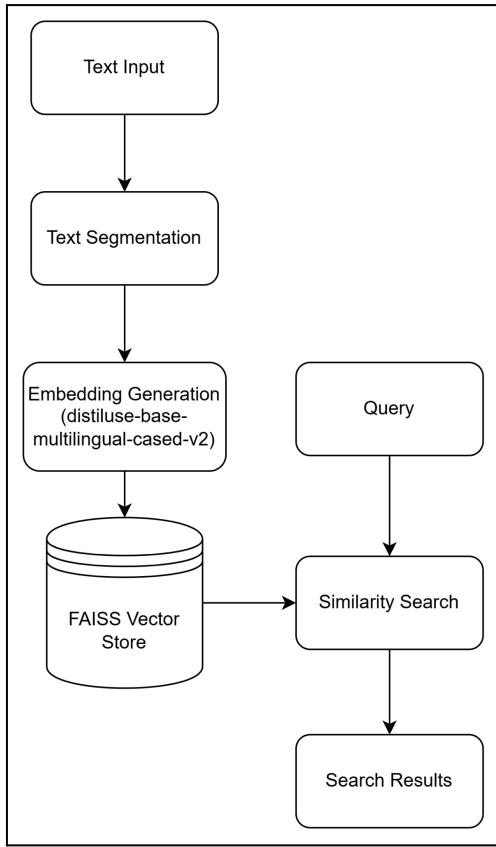
### 3.2.2 Embedding Generation



**Figure 3.5 Embedding Generation Module**

This module employs the distiluse-base-multilingual-cased-v2 model to transform processed text chunks into semantic embedding vectors. This model supports multiple languages and encodes sentences into 512-dimensional vectors, making it ideal for cross-language text analysis and semantic matching. Each piece of text is first packaged into a standard document object (Document) and then extracted using HuggingFaceEmbeddings. All generated embedding vectors will be saved in the vector database created by FAISS (Facebook AI Similarity Search) to facilitate future efficient similarity searches.

### 3.2.3 Vector Database



**Figure 3.6 Vector Database module**

This module stores text embedding vectors in an efficient vector database and supports semantic similarity retrieval. Key steps in the process include text segmentation, embedding generation, vector storage and similarity search.

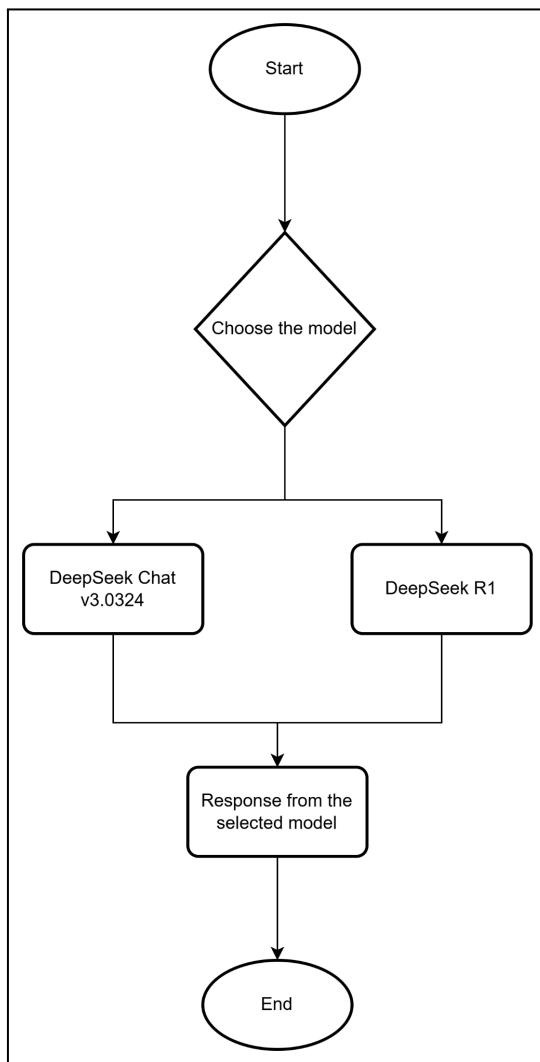
First, the system splits the text using `RecursiveCharacterTextSplitter`. Each text segment is approximately 1000 characters long and has some overlap (`chunk_overlap = 200`) to maintain contextual coherence. The `distiluse-base-multilingual-cased-v2` model is then used to generate each text block's semantic embedding, which is encapsulated as a document object and stored in the FAISS vector database.

After the user enters a query, the system will use the similarity search function (`similarity_search`) to retrieve several text fragments closest to the query from the database (the default is to return the first 10). This method supports multilingual queries and can perform semantic matching across languages, greatly improving the ability to obtain cross-language data.

Furthermore, the module includes built-in common command detection functions (such as “summarize”, “translate”, “conclusion”), which allow for the execution of more advanced text preprocessing tasks in response to user instructions. For example, if the user enters a command with keywords like “summarize” and “translate”, the system will return the summary and translation information.

Overall, this module combines semantic understanding with high-performance similarity search, providing a solid foundation for subsequent intelligent functions like question and answer, summary and translation.

### 3.2.4 LLM Integration



**Figure 3.7 General Flow for requesting response from LLM model**

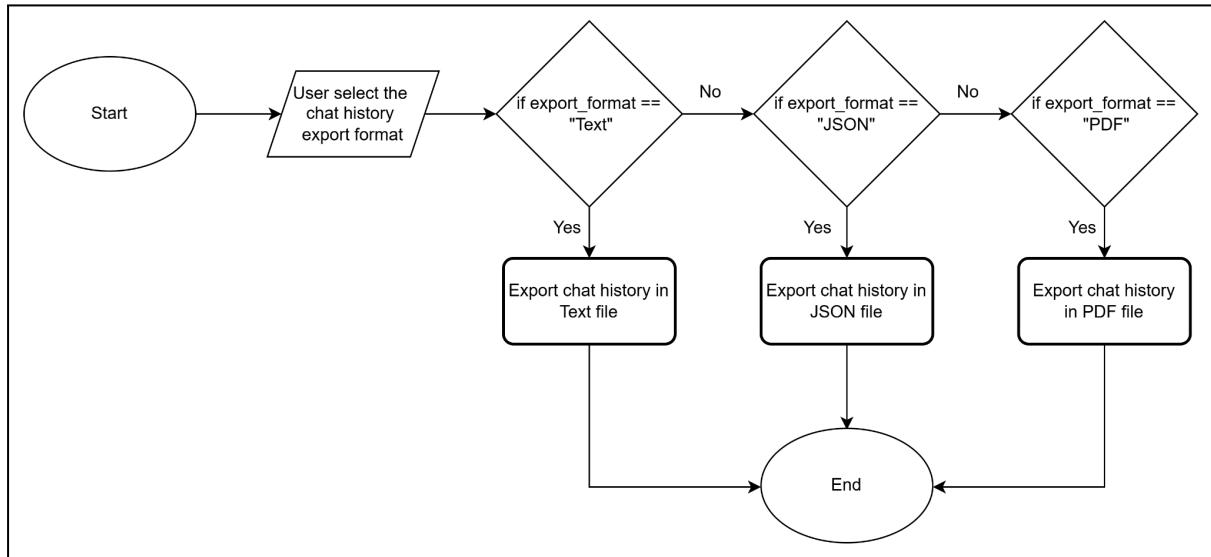
The system advanced large language models to provide a more natural and intelligent human-computer interaction. DeepSeek Chat v3.0324 and DeepSeek R1 are currently supported models, both of which use the powerful Transformer architecture and excel at multi-turn dialogue understanding, semantic reasoning and task execution.

After the user has completed the file uploading, text extraction and embedding generation, the system will send the user's query and the retrieved related text fragments to the large language model, which will generate the context-relevant natural language responses. DeepSeek Chat v3.0324 is best suited for general question-answering and multi-turn dialogues, while DeepSeek R1 is stable in language understanding and knowledge extraction,

as well as complex instruction processing tasks like summarizing, translating texts, and writing conclusions. The systems allow users to choose which language model to handle their requests as needed, thereby achieving a more flexible and personalized answer experience.

When combined with the vector database module, LLM can understand the semantic intent behind the user's question and provide accurate, context-relevant answers based on the document content, thereby improving the system's intelligence and practicality.

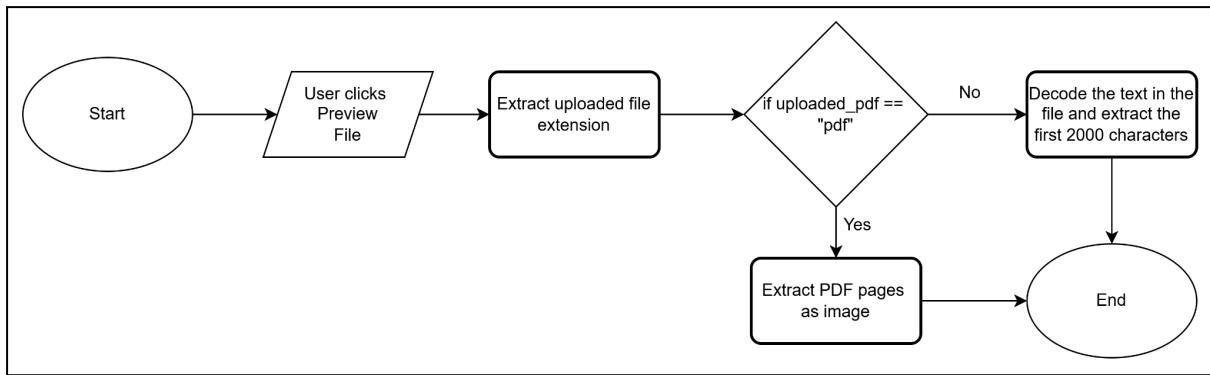
### 3.2.5 Export Chat Record



*Figure 3.8 Export Chat Function Module*

This module enables users to download their chat history in multiple formats for record-keeping or further analysis. When the user clicks the “Export Chat History” button, they are presented with three format options: Text, JSON, and PDF. If the user selects Text, the system compiles all messages into a simple plaintext file with clear labels distinguishing between user queries and assistant responses. For JSON exports, the module structures the data with metadata like timestamps and roles, creating a machine-readable file ideal for developers or data processing. The PDF option generates a visually formatted document complete with titles, section headers, and proper spacing, though it notes potential rendering issues with non-English characters. Regardless of the chosen format, the system dynamically generates the file in memory and triggers an automatic download, ensuring a seamless user experience. This feature enhances the application's utility by allowing users to preserve conversations in their preferred format while maintaining context through consistent labeling and metadata inclusion.

### 3.2.6 Preview Uploaded File Content



**Figure 3.9 Preview File Content Module**

This module is to show that the process begins when the user clicks “Preview file”, at which point the system extracts the uploaded file’s extension to determine the next processing method. If the file extension is detected as “.pdf”, the system will proceed and convert them to image format. This step is appropriate for the unique processing requirements of PDF files, ensuring that the content is presented visually. The process comes to a natural end once it is completed. However, if the uploaded file is not in PDF format, the system will decode the first 2000 characters. This operation works with non-PDF files and aims to quickly extract and display the file’s beginning content. Once the decoding is complete, the process ends.

## 3.3 Implementation Details

### 3.3.1 Backend

The system’s backend development uses the Python programming language. With its broad application and powerful ecology in natural language processing and machine learning, it lends strong support to the system’s rapid development and stable operation. The entire backend is built around the five core functions which are “document processing”, “text processing”, “semantic embedding generation”, “vector database construction” and “large language model integration”. It has a modular structure design, clear logic, and distinct functions, making it easy to maintain and expand later.

Firstly, the system can process a variety of document formats, which includes PDF, DOCX and plain text (TXT) files. After the file is uploaded, the plain text is extracted using the corresponding parser, and any file types or formats that are not recognized or supported are prompted for exception processing. To ensure the quality of the embedded generation, the extracted text will go through the processing stage, which is the standard natural language processing such as lowercase, special characters and number removal, word segmentation and stop word removal will be performed in order.

In terms of semantic understanding, HuggingFace’s distiluse-base-multilingual-cased-v2 model is used to build semantic vectors for multilingual text. Before vectorization, text is

sliced using RecursiveCharacterTextSplitter. Each slice is turned into a high-dimensional vector and saved in the FAISS vector database. FAISS efficient similarity search tool can swiftly match the user's query to the text fragment with the closest semantics.

In addition, the system integrates two huge language models, DeepSeek Chat v3.0324 and DeepSeek R1, which are employed for natural language creation in various task contexts. Users can select specific models based on their requirements to execute instruction-based tasks including translation, summary and conclusion writing. Using the command recognition mechanism, the system automatically analyzes user intentions and executes the related task.

The system's backend implementation mechanism ensures not only efficient document data processing and management, but also powerful semantic understanding and language generation capabilities, providing users with an accurate, smooth and context-related intelligent response experience.

### 3.3.2 Frontend

The system's frontend interface is built with streamlit, a lightweight web application framework. This tool is utilized in this system to quickly design an interactive user interface because of its quick deployment, user-friendly interface and flexible deployment. Streamlit offers real-time data interaction and can integrate smoothly with backend logic, significantly boosting the system's overall reaction efficiency and user experience.

In terms of user interface design, the system integrates several key interactive components:

- File upload widget
  - It allows users to upload documents in a variety of formats, which includes PDF, DOCX, and TXT and each of them cannot exceed 200 MB. The system detects the file type and invokes the appropriate extraction module to turn the content into processable plain text.
- Chat window
  - It has a text interaction area similar to a chat interface where users can enter the question, and the system's responses are shown in the form of messages one by one, simulating a natural conversation scenario.
- Send button
  - After entering the question, the user can initiate the system answer by pressing the enter button. The system will automatically query the vector database for semantic retrieval, and the language model will produce the appropriate response.

- Real-time response streaming
  - It allows the incremental display of answer content in a streaming manner, allowing users to obtain some information in real time without having to wait for complete generation, improving the immediacy and engagement experience.
- Navigation
  - The frontend interface includes a simple and easy navigation mechanism. Users can navigate to several module regions according to their requirements, such as document processing, language models. At the same time, the system also supports users to preview previous chart records to review historical conversation content.

### **3.3.3 Chat History Management**

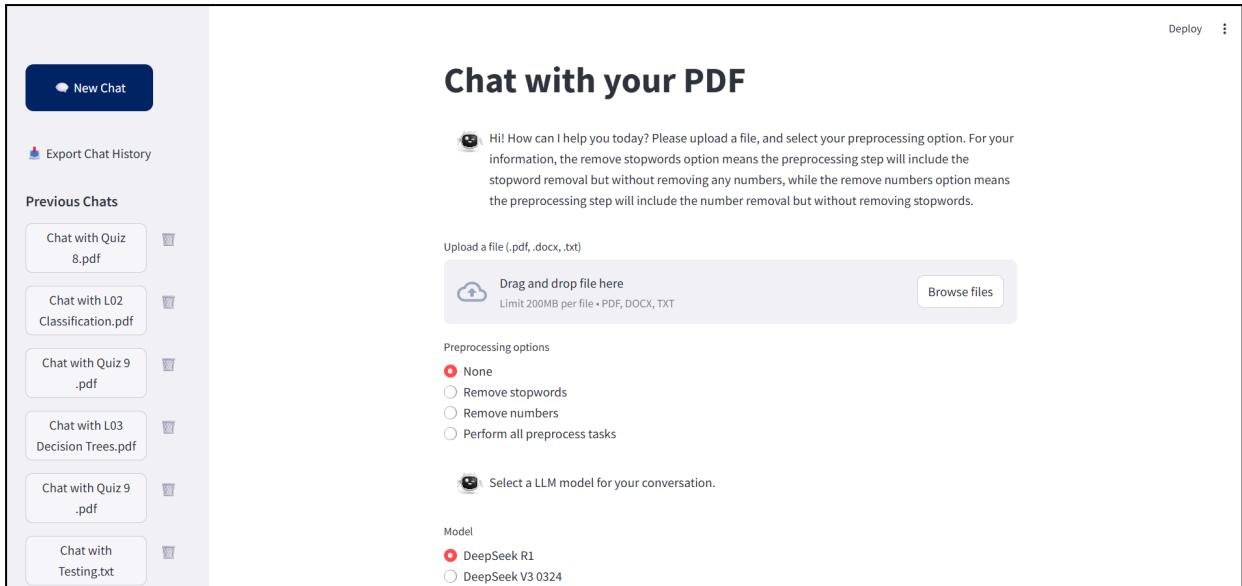
The backend database solution used by the system is Firebase Firestore Database, which allows for the persistent storing of chart records and user history query features. Firestore is a cloud-based document database provided by Google that has real-time synchronization, a flexible structure, and easy growth, making it ideal for efficient integration with frontend applications.

In this system, all interaction between the user and the language model which includes the user input and system response will be prepared and saved as a structured document in Firestore. This method enables users to examine previous talks in the interface, which improves information traceability and interaction continuity.

It is worth mentioning that, in order to keep system interaction simple, users can only see historical records and cannot ask questions based on previous interactions. Each inquiry is treated as a new session request, which helps to clarify the semantic context and simplifies session state management.

# 4. Results

## 4.1 User Interface



**Figure 4.1 Main Page of Chat Application**

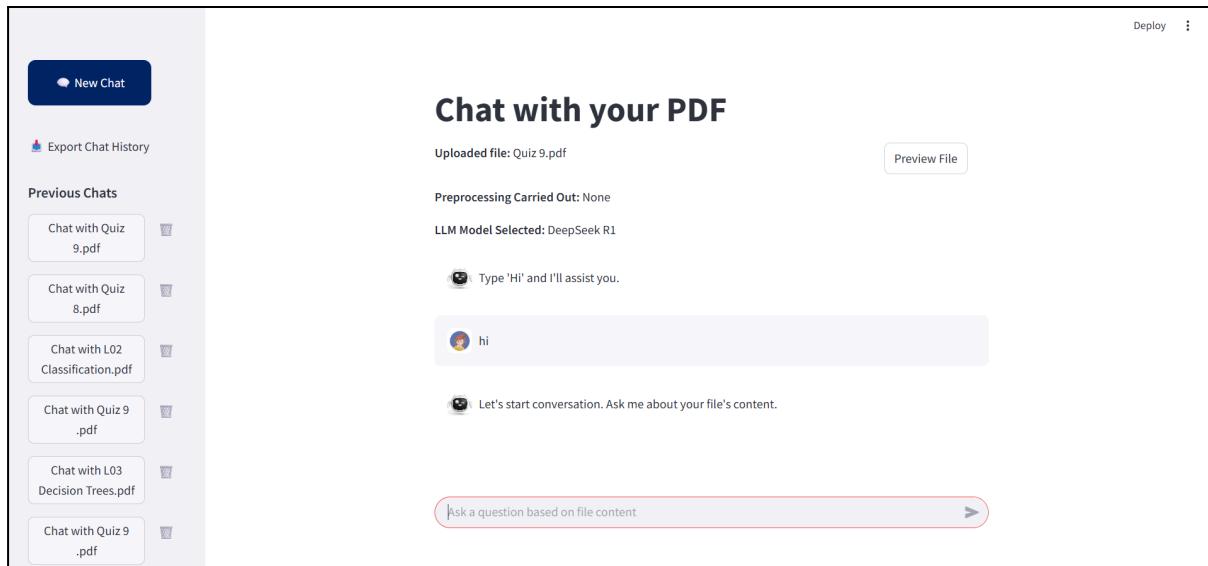
This user interface is used to interact with PDF, DOCX, TXT files, specially for uploading files before starting conversations. Here are the various components of the interface.

First, the left sidebar has a button ('New Chat') for the user to initiate a new conversation, a button ('Export Chat History') for exporting the chat history and a list of the user's previous chat history. The chat history is indicated by the title 'Previous Chats'. Below the title is a list of the user's previous chat history, in which each chat history is identified by using the name of the file uploaded as the title of the record, such as "Chat with TESTING.txt" and "Chat with L03 Decision Trees.pdf". The system also provides the option for users to delete unwanted chat history by clicking the trash can icon. Meanwhile, the user can also click on the previous chat title to view any of the previous chat history.

The main area beside the left sidebar is the main place for the user to interact with the chat application. When the user first initiates a new conversation, the chat application will first display an area for the user to upload the file in order to start a conversation. The title is "Chat with your PDF", while there is a message at the top of the interface that serves as a guidance for the user: "Hi ! How can I help you today? Please upload a file, and select your preprocessing option....". The message is then followed by a file uploading area, where users can upload files in PDF, DOCX, or TXT format, with a restricted maximum file size of 200 MB. Users are able to click the 'Browse files' button to select files to be uploaded from the device or directly drag a file to the area for uploading.

Then, the user is allowed to select the preprocessing options for the text extracted from the files. There are four options to be selected: 'None' which means there is no preprocessing

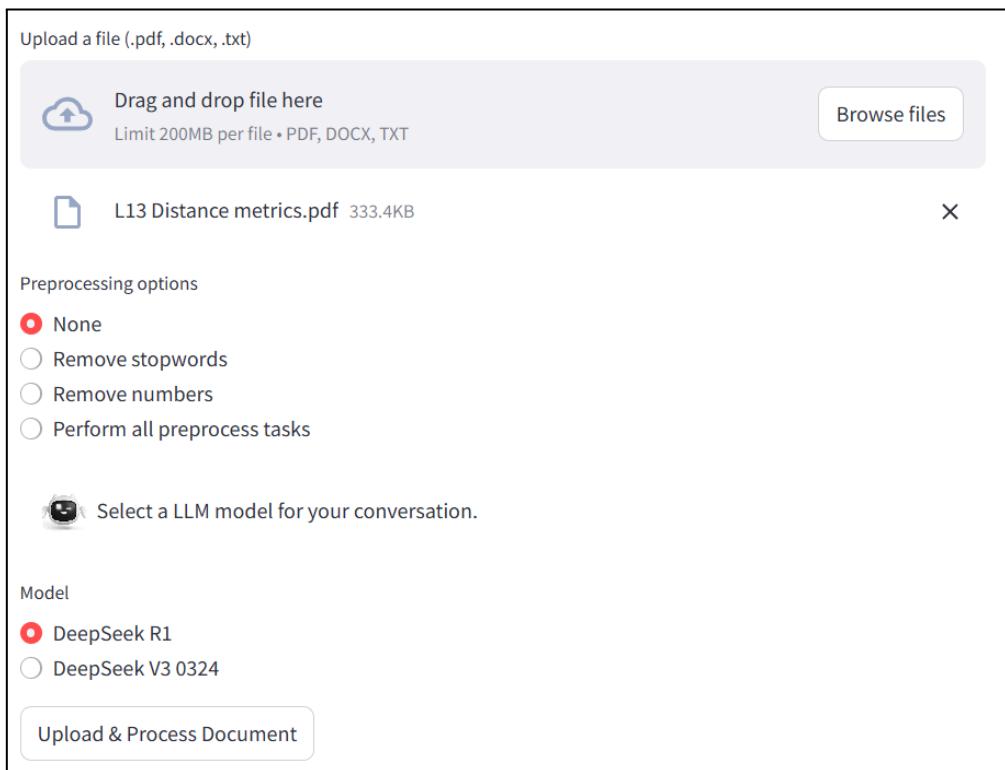
will be carried out, ‘Remove stopwords’ means that the preprocessing will include the step for removing the stopwords, the common meaningless words, but without removing any numbers from the text extracted, ‘Remove numbers’ means that the preprocessing will include the number removal step but without removing any stopwords from the extracted text, while ‘Perform all preprocess tasks’ means all preprocessing tasks will be carried out on the extracted text. Lastly, the user is given the option to select a LLM model for their conversation, in which there are two models to be chosen: ‘DeepSeek R1’ and ‘DeepSeek V3 0324’.



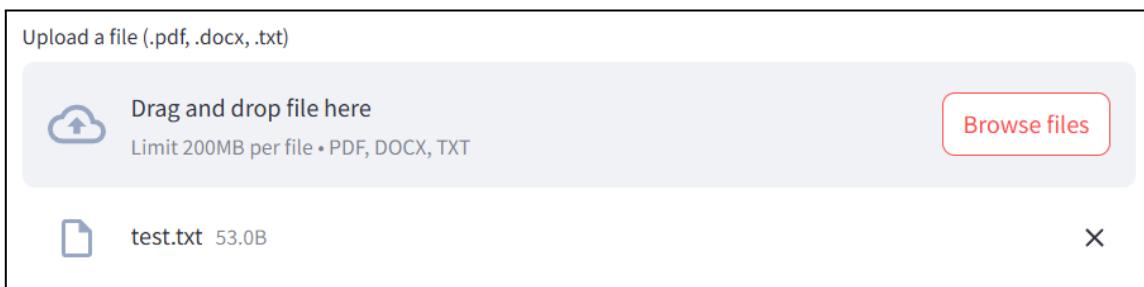
**Figure 4.2 User Interface of Chat Interaction with File Uploaded**

The figure above shows the user interface of the main feature of the chat application, in which the user can communicate with the system to ask or interact with the content of the file uploaded. At the top of the chatting interface, it will first show the uploaded file name with a button ('Preview File') that allows the user to preview the uploaded file content. Besides that, it will also show the choice for the preprocessing option and the LLM model selected for this conversation. User is able to enter any query or input to interact with the bot, while all the inputs or the responses generated will be kept showing at the message area (the middle area).

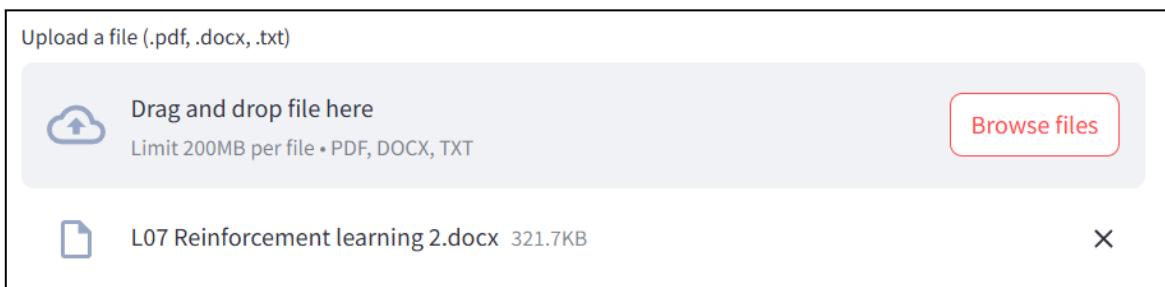
## 4.2 Accepting Multiple File Types (PDF, DOCS, TXT)



**Figure 4.3** User uploads pdf file and the ‘Upload & Process Document’ button shown

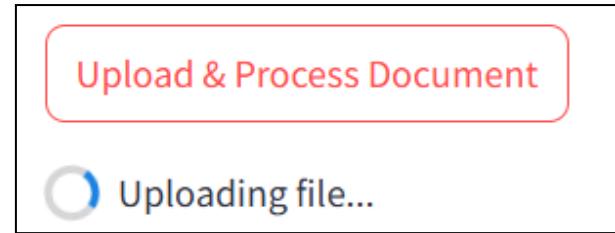


**Figure 4.4** User uploads txt file



**Figure 4.5** User uploads docx file

The three figures above show that the user is able to upload three types of file, which is the pdf file, docs file and txt file. When the file is successfully uploaded, the ‘Upload & Process Document’ will be shown. Users can click the button after selecting the corresponding preprocessing option and LLM model to start the conversation.



**Figure 4.6 File Processing Loading after clicking the button**

In order to maintain the smooth and intuitive user experience, a file processing loading will be shown below the button to inform the user about what operations that the system is carried out. This loading display is important as it archives one of the important aspects of UI design, minimal surprise, which means the user should be able to know what is happening now and able to know what will happen next. By having the loading status, the user can know that the file has been uploaded and the system is processing it, instead of just looking at the page that shows nothing until the file successfully be processed by the system.

The screenshot shows a user interface for file processing. At the top, a green banner displays the message "File uploaded successfully.". Below this, the heading "Extracted Raw Text from the file uploaded:" is shown, followed by the note "(Only shows the first 2000 characters)". A text area contains the following content:

BMCS2114 Machine Learning  
Quiz 8  
Q1. What is the role of an agent in reinforcement learning?  
a) It mimics the behavior of the environment.  
b) It determines the rewards in the environment.  
c) It performs actions to gain rewards in the environment.  
d) It maps states to actions in the environment.

At the bottom, there is a message from a chatbot: "Type 'Hi' and I'll assist you." Below this is a input field with the placeholder "Ask a question based on file content" and a right-pointing arrow icon.

**Figure 4.7 Message shown after file successfully uploaded**

When the system successfully processes the file uploaded, a message will show at the user interface to indicate that the file has been successfully uploaded. Besides that, the system will display the first 2000 characters of the extracted raw text from the uploaded file. This is used to let the user to view whether the file uploading and text extraction works correctly. Then, a message from the chatbot will be displayed to instruct the user how to start the chat in order to communicate with the chatbot about the uploaded file content. By typing any input, the user will be navigated to the actual chat user interface (**Figure 4.2**).

## 4.3 Validation of Responses to File-Related Questions (Summarization and Translation)

The figure below shows the response from the chatbot by using different preprocessing options (None or Perform all preprocess tasks) and the LLM model (Deepseek R1 or Deepseek V3 0324). The same file is used across the four situations, ‘Quiz 9.pdf’. Meanwhile, the exact same query is given across the four situations.

### Situation 1: No preprocessing carried out & Deepseek R1 Model

 Summarize the content of the document

 The document contains Quiz 9 from a Machine Learning course (BMCS2114) with five multiple-choice questions focused on **reinforcement learning**:

1. Q1 asks about characteristics of reinforcement learning, listing options like sequential decision-making (c) and environment-driven data (d).
2. Q2 explores feedback mechanisms, highlighting delayed feedback linked to the agent's actions (c).
3. Q3 distinguishes between reinforcement learning types, contrasting model-based (a) and model-free (b) approaches.
4. Q4 addresses the objective, with the goal being to determine an optimal policy (c).
5. Q5 clarifies what the agent receives post-action: a scalar reward (c).

The content strictly outlines conceptual themes in reinforcement learning without indicating correct answers.

 Response time: 24.06 seconds

**Figure 4.8 User asks for summarization**

 Translate the main point of the document into chinese

 翻译文档主要内容 (中文版):

BMCS2114 机器学习  
测验 9

Q1. 强化学习的特征是什么?  
a) 即时反馈  
b) 存在监督者  
c) 顺序决策  
d) 环境决定后续数据

Q2. 强化学习的反馈机制如何运作?  
a) 反馈即时且立即  
b) 由监督者提供反馈  
c) 反馈延迟，基于智能体的行为  
d) 由环境决定反馈

## 关键术语对照：

- 强化学习 (Reinforcement Learning)
- 顺序决策 (Sequential Decision Making)
- 无模型的强化学习 (Model-Free RL)
- 最优策略 (Optimal Policy)
- 标量奖励 (Scalar Reward)

翻译严格遵循原文结构与技术术语，未添加额外内容。

*Figure 4.9 User asks for translation in chinese*

### Situation 2: Perform all preprocess tasks & Deepseek R1 Model

Summarize the content of the document

Summary of Quiz 9 on Reinforcement Learning (BMCS2114):

1. Q1: Reinforcement learning is characterized by **sequential decision making** (c).
2. Q2: Feedback in reinforcement learning is **delayed and based on the agent's actions** (c).
3. Q3: **Model-free RL** does not use a model to solve problems (b).
4. Q4: The objective is to **determine the optimal policy for the agent** (c).
5. Q5: After performing an action, the agent receives a **scalar reward or reinforcement** from the environment (c).

Answers are derived directly from the quiz content and standard reinforcement learning principles.

Response time: 14.14 seconds

*Figure 4.10 User asks for summarization*

Translate the main point of the document into chinese

以下是文档内容的简体中文翻译：

---

BMCS2114 机器学习  
测验 9

问题 1：以下哪项是强化学习的特点？

- a) 即时反馈
- b) 存在监督者
- c) 序列化决策
- d) 环境决定后续数据

**问题 4：强化学习的目标是什么？**

- a) 最大化即时奖励
- b) 最小化长期回报
- c) 确定智能体的最优策略
- d) 模仿环境的行为

**问题 5：智能体执行动作后从环境中获得什么？**

- a) 环境的当前状态
- b) 下一步动作的策略
- c) 标量奖励或强化信号
- d) 环境模型

翻译严格遵循原文内容，未添加额外信息，术语与结构均与原始文档一致。

*Figure 4.11 User asks for translation in chinese*

### **Situation 3: No preprocessing carried out & Deepseek V3 0324**



Summarize the content of the document



Here is a summary of the content from the document:

The document is a quiz (Quiz 9) related to reinforcement learning in the course "BMCS2114 Machine Learning." It consists of five multiple-choice questions covering key concepts in reinforcement learning, including:

1. Characteristics of reinforcement learning (e.g., sequential decision-making, delayed feedback).
2. How feedback occurs in reinforcement learning (e.g., delayed and based on the agent's actions).
3. Types of reinforcement learning (e.g., model-free RL).
4. The objective of reinforcement learning (e.g., finding the optimal policy for the agent).
5. What the agent receives from the environment after performing an action (e.g., a scalar reward).

The questions focus on fundamental aspects of reinforcement learning, such as feedback mechanisms, model types, and the agent-environment interaction.

*Figure 4.12 User asks for summarization*



Translate the main point of the document into chinese



BMCS2114 机器学习



测验9

Q1. 以下哪项是强化学习的特点?

- a) 即时反馈
- b) 存在监督者
- c) 序列决策
- d) 环境决定后续数据

Q2. 强化学习中的反馈是如何发生的?

- a) 即时且立即
- b) 由监督者提供
- c) 延迟且基于智能体的动作
- d) 由环境决定

**Figure 4.13 User asks for translation in chinese**

#### **Situation 4: Perform all preprocess tasks & Deepseek R3 0324**



Summarize the content of the document



The document is a quiz (Quiz 9) on BMCS2114 Machine Learning, focusing on reinforcement learning (RL). It consists of five multiple-choice questions covering key RL concepts:

1. Q1: Highlights characteristics of RL, with the correct answer being *sequential decision making*.
2. Q2: Explains RL feedback as *delayed and based on the agent's actions*.
3. Q3: Identifies *model-free RL* as the type that does not use a model.
4. Q4: States the objective is to *determine the optimal policy for the agent*.
5. Q5: Describes the agent receiving a *scalar reward* from the environment after an action.

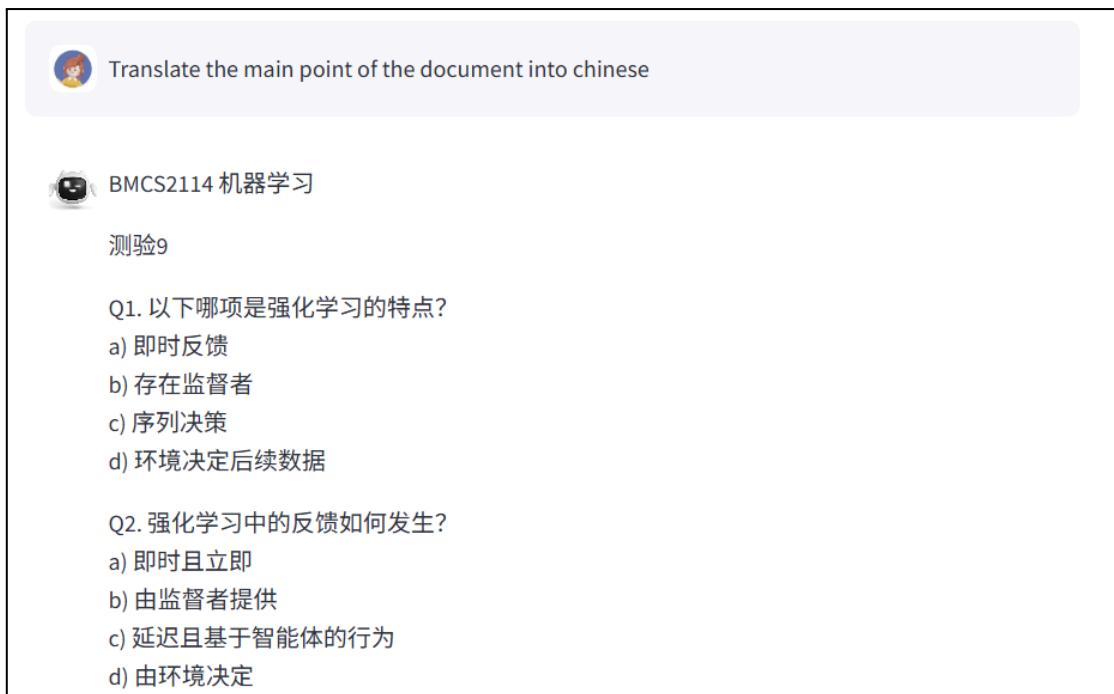
The quiz tests understanding of RL fundamentals, including feedback mechanisms, model types, and goals.

(Note: Summary strictly adheres to the provided document content.)



Response time: 5.31 seconds

**Figure 4.14 User asks for summarization**



*Figure 4.15 User asks for translation in chinese*

**Validation with the ChatGPT Answer (Reference):**

In order to validate whether the response given by the chatbot is relevant, I have used the current trending chatbot, ChatGPT to generate responses on the same file and with the same query used previously. The figure below shows the response from ChatGPT.

The screenshot shows a messaging interface. A user icon with a blue background and a white character is followed by the text: "Quiz 9.pdf PDF". Below this, the text "Summarize the content of the document" is displayed. The message then provides a summary of the document:  
The document titled "Quiz 9" from the course **BMCS2114 Machine Learning** contains a set of five multiple-choice questions focused on **Reinforcement Learning (RL)**. The quiz assesses understanding of key RL concepts, including:  

- Characteristics of RL**, emphasizing **sequential decision-making**.
- Nature of feedback**, highlighting that it is **delayed and based on agent actions**.
- Model-free vs. model-based approaches**, where **model-free RL** does not rely on an environmental model.
- Objective of RL**, which is to **determine the optimal policy**.
- Agent-environment interaction**, specifically that an agent receives a **scalar reward** after an action.

Would you like the correct answers or explanations for each question?

*Figure 4.16 Response from ChatGPT for the file summarization*

Translate the main point of the document into chinese

该文档是 BMCS2114 机器学习 课程的 第9次测验 (Quiz 9)，内容主要围绕 强化学习 (Reinforcement Learning, RL) 的核心概念展开。测验包括五道多项选择题，主要考察以下知识点：

1. 强化学习的特点，强调其是一个 序列决策过程。
2. 反馈的性质，指出其是 延迟的，并取决于智能体的行为。
3. 是否使用模型的问题，指出 无模型强化学习 (Model-free RL) 不依赖环境模型。
4. 强化学习的目标，是 确定最优策略 (optimal policy)。
5. 智能体与环境的交互，智能体在执行动作后会获得一个 标量奖励 (scalar reward)。

需要我把整份测验逐题翻译成中文吗？



*Figure 4.17 Response from ChatGPT for the chinese translation*

From the human perspective, by directly comparing the response generated by the ChatGPT with the chatbot generated response, it is found that the context in the generated response is quite similar, which means that the chatbot is able to generate a relevant response.

For a better validation, we have decided to use the cosine similarity to compare the response generated by the chatbot and ChatGPT. Before calculating the similarity, we have used the sentence transformer, ‘distiluse-base-multilingual-cased-v2’ to encode both the responses. The detailed code can be found at the ‘validation.ipynb’ that is submitted along with the source code.

*Table 4.1 Cosine Similarity Score for the Response Generated*

Situation	Cosine Similarity
No preprocess & Deepseek R1	Summarization: 0.7614 Translation: 0.6207
Perform all preprocess & Deepseek R1	Summarization: 0.7432 Translation: 0.6457
No preprocess & Deepseek V3 0324	Summarization: 0.7812 Translation: 0.6435
Perform all preprocess & Deepseek V3 0324	Summarization: 0.8314 Translation: 0.6583

Based on the result, we found that the cosine similarity scores at the four situations are having a moderate high score, in which all are above 0.6. This has also proven the previous human perspective analysis, in which the response from the chatbot is relevant to the document by comparing it to the reference response (ChatGPT’s response). Among the four situations, the fourth situation, in which all the preprocessing tasks are carried out and using

Deepseek V3 0324 model, the summarization and translation provided are most similar compared to the ChatGPT's response as it is having the highest score compared to the other three.

## 4.4 Answering General or Unrelated Questions

As most of the chatbot nowadays will perform preprocessing steps on the extracted text when interacting with the file uploaded by the user, hence the following demonstration will focus on the response from two models with the same preprocessing option ('Perform all preprocess tasks'). The same file is used across the two models, 'L02 Classification.pdf'. Meanwhile, the exact same query will be given to the two models.

### **Situation 1: Asking Context in the Document**



What is the classification



**Answer:**

Classification in machine learning is a supervised learning task that predicts the class category of a new observation based on patterns learned from previous examples. It involves assigning predefined labels to instances by analyzing their features. Below are the key components and types of classification tasks, as detailed in the context:

### **Key Elements of Classification**

1. **Features:** Quantifiable variables (numerical or categorical) that help distinguish different classes. Examples include transaction amount, credit score, or purchase frequency.
2. **Labels:** Known outcomes (e.g., "approved/denied" for loans, "rose/lily" for flowers) used to train supervised models.
3. **Similarity Measures:** Metrics like Euclidean distance, cosine similarity, or Jaccard similarity to compare observations.

## Bank Scenario Example

- **Task:** Predict loan approval ("approved" or "denied").
- **Features:** Income, credit score, age, loan amount.
- **Label:** Binary outcome ("approved" or "denied").
- **Similarity Measure:** Euclidean distance or cosine similarity (algorithm-dependent).
- **Challenge:** Imbalanced data due to fewer denied loans, requiring techniques like resampling or weighted loss functions.

This structured approach ensures accurate predictions by leveraging historical data, feature analysis, and appropriate algorithms tailored to the classification type.

⌚ Response time: 28.72 seconds

**Figure 4.18 Partial Chatbot Response (Deepseek R1 Model)**



What is the classification



The context provided discusses several types of classification tasks in machine learning, which are used to predict categories or labels for new observations based on previous examples. Here are the detailed types of classification mentioned:

1. **Binary Classification:**

- Classifies items into one of two categories.
- Examples:
  - Spam detection (spam or not spam).
  - Fraud detection (fraudulent or not fraudulent).
  - Sentiment analysis (positive or negative).
- Output: Binary decision (e.g., yes/no, true/false).

2. **Multiclass Classification:**

- Classifies items into one of three or more categories.
- Examples:
  - Image classification (e.g., classifying images into objects, scenes, or activities).

## Key Elements for Developing a Classification Model:

1. **Feature Extraction:** Identifying relevant features (e.g., income, credit score, age for loan applications).
2. **Labeling:** Using known labels (e.g., approved or denied for loan applications).
3. **Similarity Measure:** Using metrics like Euclidean distance or cosine similarity to compare observations.
4. **Model Training:** Using algorithms like logistic regression, decision trees, or neural networks.
5. **Model Evaluation:** Testing the model on unseen data to measure accuracy.
6. **Prediction:** Using the trained model to predict new observations.

This summary covers the classification types and their applications as described in the provided context.

 Response time: 15.87 seconds

**Figure 4.19 Partial Chatbot Response (Deepseek V3 0324 Model)**

By looking at the response generated by two models on the same question, we are able to observe that both the responses are having a quite high similarity. Their explanations are related to each other, and strictly based on the document's context. The only difference may be the structure formatted, both the models are giving the similar context, but they may answer in different structure or explanation form. The other aspect that we can observe is the time aspect. Based on the response time shown below the response generated, we found out the Deepseek V3 0324 Model is generating response more faster than the Deepseek R1 Model.

## **Situation 2: Asking a question exists in the document**



What is the multi-label classification task in the context of the bank's scenario



In the context of the bank's scenario, a **multi-label classification task** would involve assigning **multiple labels** (e.g., binary or categorical outcomes) to a single loan application instance. For example:

- **Task:** Predicting whether a loan applicant is **likely to default on the loan** and whether they are **likely to make timely payments**. Here, both labels (default risk and payment punctuality) are assigned to the same applicant, even though these outcomes may not be mutually exclusive.

### **Key Details:**

- **Feature Used:** Applicant information such as income, credit score, age, and other relevant factors.
- **Label:** Binary labels like "default/no default" and "timely payment/delayed payment" applied simultaneously.
- **Method:** Machine learning algorithms trained to identify patterns in historical data to predict multiple outcomes for a single instance.

This task helps the bank manage risks by evaluating multiple aspects of a loan applicant's behavior simultaneously.



Response time: 14.56 seconds

***Figure 4.20 Chatbot Response (Deepseek R1 Model)***



What is the multi-label classification task in the context of the bank's scenario



In the context of the bank's scenario, a multi-label classification task would involve assigning multiple labels to a single loan application based on various characteristics or outcomes.

For example, the bank might want to predict:

- Whether the loan applicant is likely to default on the loan (Yes/No).
- Whether the applicant is likely to make payments on time (Yes/No).
- Whether the applicant qualifies for additional financial products (e.g., credit card, insurance).

Here, each loan application can have multiple labels (e.g., "likely to default," "unlikely to make payments on time," "qualifies for credit card"), making it a multi-label classification problem. The output would be a set of binary or categorical labels for each instance (loan application), rather than a single class label.

Key details:

1. **Features used:** Applicant information like income, credit score, age, etc.
2. **Labels:** Multiple binary or categorical outcomes (e.g., default risk, payment behavior, product eligibility).
3. **Similarity measure:** Machine learning algorithms would identify patterns in historical data to assign these labels, using metrics like Jaccard similarity for multi-label scenarios.

This differs from binary (single yes/no outcome) or multiclass (single label from multiple categories) classification.

⌚ Response time: 8.76 seconds

**Figure 4.21 Chatbot Response (Deepseek V3 0324 Model)**

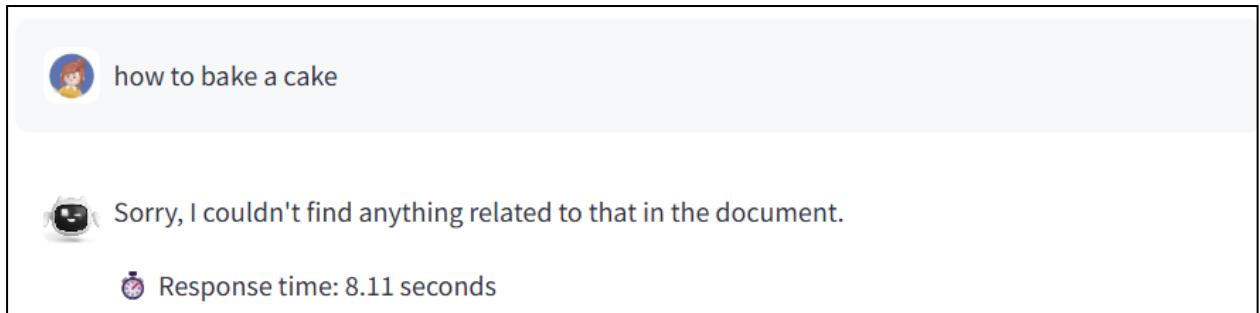
- viii) Give one example of a multi-label classification task in the context of the bank's scenario.
- viii) To predict whether a loan applicant is likely to default on their loan and whether they are likely to make their payments on time.

**Figure 4.22 Context of the asking question in the uploaded file**

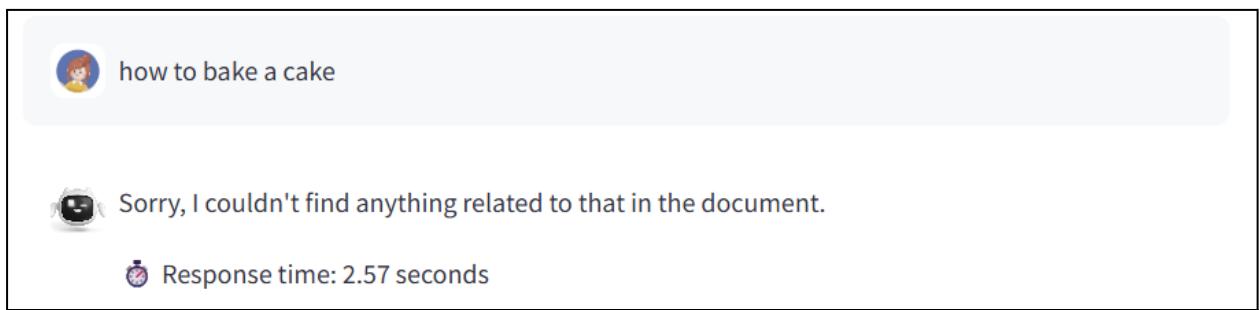
In the second situation, the given input is a question that exists in the uploaded file. Same like the first situation, the response from the two models are showing the high similarity, and strictly related to the uploaded file's context. The response not only gives what the document writes, the model is also able to generate some explanation on it. However, if we compare to the answer written in the uploaded document, the response from Deepseek R1 is exactly the same as the document's answer; meanwhile the response from Deepseek V3 0324 has added one point on the label (product eligibility). It does not show that the Deepseek V3 0324's performance is not good, as it is able to provide more context on the basis of the provided

context. The result will be only considered as not accurate if the user is requesting an exactly same answer or explanation that exists in the provided context, while in this situation, the additional context may confuse the user. Moreover, at the time aspect, the Deepseek V3 0324 model is still generating the response faster than the Deepeek R1 model.

#### **Situation 3: Asking something that is not related to the uploaded file content**



***Figure 4.23 Chatbot Response (Deepseek R1 Model)***



***Figure 4.24 Chatbot Response (Deepseek V3 0324 Model)***

When the user is asking something that is not related to the uploaded file context, the system will prompt a message to indicate that the requested information is not found in the document. It will not directly prompt any other that is relevant to the user's request, all the responses generated by the chatbot will be strictly following the provided context.

## **4.5 Handling Spelling Mistakes or Typo**

Based on the figures below, both the models are able to handle the query with spelling errors. Although the user entered something that is having a spelling error, the chatbot still can understand the context and provide the relevant response.



wa is da multilabell clasificatio



**Answer:**

Multi-label classification is a type of machine learning task where **multiple labels (or categories)** can be assigned to a single instance (e.g., a data point or object). Unlike binary or multi-class classification, which assign only one label per instance, multi-label classification allows for **multiple, non-exclusive labels** to be associated with the same input.

## Key Details from the Context:

### 1. Definition:

- Assigns **multiple labels** to a single instance (e.g., tagging a blog post with multiple topics or an image with multiple objects).
- Output is a **set of binary labels** (e.g., "yes/no" for each possible category).

## Example from the Context:

A bank might use multi-label classification to predict whether a loan applicant is **likely to default** (label 1) and whether they are **eligible for a special offer** (label 2) based on their financial history.

If you need further clarification, feel free to ask!



Response time: 31.97 seconds

*Figure 4.25 Partial Chatbot Response (Deepseek R1 Model)*



wa is da multilabell clasificatio



Multilabel classification is a type of classification task where multiple labels can be assigned to a single instance. Unlike binary or multiclass classification, where each instance is assigned to only one class or label, multilabel classification allows for multiple labels to be associated with a single instance.

#### Examples from the context:

1. **Text Categorization:** Assigning multiple tags or categories to a given text (e.g., a blog post or news article that covers multiple topics).
2. **Image Tagging:** Assigning multiple labels to an image (e.g., an image might contain a dog, a park, and sunshine).
3. **Music Genre Classification:** Assigning multiple genres to a song (e.g., a song could be both "rock" and "pop").

**Output:** The output is a set of labels, which can be binary (e.g., yes/no for each label) or multiclass (e.g., multiple categories).

#### Example in the Bank Scenario:

- **Multilabel Classification Task:** Predicting whether a loan applicant is likely to default on the loan **and** whether they are likely to make payments on time. Here, both "default" and "on-time payment" can be labels assigned to the same applicant.

#### Key Points:

- Labels are not mutually exclusive.
- Used when an instance can belong to multiple categories simultaneously.
- Common in tasks like content tagging, where multiple descriptors apply.

If you need further clarification, feel free to ask!

⌚ Response time: 7.64 seconds

**Figure 4.26 Partial Chatbot Response (Deepseek V3 0324 Model)**

## 4.6 Handling Mixed Language

Based on the figures below, both the models are able to handle the query with mixed language. Although the user has used multiple language to express the request, the chatbot still can understand and provide the relevant response.



bolehkah anda bagi saya 意思 of the multiclass classification



多类分类（Multiclass Classification）是一种机器学习任务，旨在将数据样本分类到三个或更多类别中的其中一个。与二分类（仅区分两个类别）不同，多类分类适用于更复杂的场景，需要模型从多个可能的类别中确定最合适的标签。以下是其核心要点：

### 定义与特点

- **分类目标：**将每个输入数据（如文本、图像、用户行为等）分配到多个预定义类别中的唯一一个类别。
- **输出形式：**结果为单一类别标签（如“猫”“狗”“鸟”），而非多个标签或数值。

### 示例（根据上下文）

- **电商场景：**将商品分类为“电子产品”“服装”“家居”等，以优化网站展示。
- **医疗场景：**根据患者症状判断疾病类型（如区分流感、过敏或肺炎）。

### 挑战

- **类别平衡：**需处理不同类别样本数量不均的问题（如某些类别数据极少）。
- **特征选择：**需提取能有效区分多个类别的特征（如文本中的关键词、图像中的纹理）。

若需具体实现细节或更多案例，可进一步说明！



Response time: 26.85 seconds

Figure 4.27 Partial Chatbot Response (Deepseek R1 Model)



bolehkah anda bagi saya 意思 of the multiclass classification



Multiclass classification ialah suatu tugas dalam pembelajaran mesin di mana item diklasifikasikan ke dalam satu dari tiga atau lebih kategori. Contohnya termasuk:

1. **Klasifikasi Gambar:** Mengklasifikasikan gambar ke dalam berbagai objek, pemandangan, atau aktivitas.
2. **Klasifikasi Topik:** Mengkategorikan teks ke dalam berbagai topik.
3. **Identifikasi Bahasa:** Mengidentifikasi bahasa dari teks yang diberikan.

**Output** dari multiclass classification adalah keputusan kategorikal berupa label kelas atau deskripsi teks. Contoh lainnya termasuk:

- **Pengenalan Tulisan Tangan:** Mengklasifikasikan digit tulisan tangan (0-9).
- **Pengenalan Ekspresi Wajah:** Mengenali ekspresi wajah seperti senang, sedih, atau marah.
- **Klasifikasi Penyakit:** Mengklasifikasikan jenis kanker atau infeksi tertentu.

Dalam konteks bank, contoh multiclass classification adalah memprediksi apakah pemohon pinjaman akan **disetujui, ditolak, atau ditawarkan jenis pinjaman berbeda** (misalnya, dengan suku bunga lebih tinggi atau jumlah pinjaman lebih rendah).

Fitur yang digunakan bisa mencakup pendapatan, skor kredit, usia, dan faktor lainnya, sementara **label** diketahui berdasarkan hasil pinjaman sebelumnya. **Metode pengukuran kesamaan** (seperti jarak Euclidean) digunakan untuk melatih model dalam mengenali pola dan mengklasifikasikan aplikasi pinjaman baru.



Response time: 9.56 seconds

**Figure 4.28 Partial Chatbot Response (Deepseek V3 0324 Model)**

Based on the response generated, we observed that the response from the Deepseek R1 model is in Chinese, while the response from Deepseek V3 0324 model is in Malay. By looking at the response generated, both the models are able to provide the response that is highly related to the context. The reason that causes response in different languages may be the different query interpretation by the two models. Deepseek R1 may take consider the word ‘意思 (meaning)’ has the highest weightage, so the response is generated in Chinese word. Meanwhile, the Deepseek V3 0324 model may take in consideration the Malay word appearing at the beginning of the query, so it generates the response in Malay word.

## 4.8 Error Handling Performance

### Chat with your PDF



Hi! How can I help you today? Please upload a file, and select your preprocessing option.

Upload a file (.pdf, .docx, .txt)



Drag and drop file here

Limit 200MB per file • PDF, DOCX, TXT

Browse files



books\_data\_file.csv text/csv files are not allowed. File must be 200.0MB or smaller. !



**Figure 4.29 User uploads an invalid file type and the file size exceeds 200MB**

Upload a file (.pdf, .docx, .txt)



Drag and drop file here

Limit 200MB per file • PDF, DOCX, TXT

Browse files



authors\_data\_file.csv text/csv files are not allowed. !



Preprocessing options

- None
- Remove stopwords
- Remove numbers
- Perform all preprocess tasks



Select a LLM model for your conversation.

Model

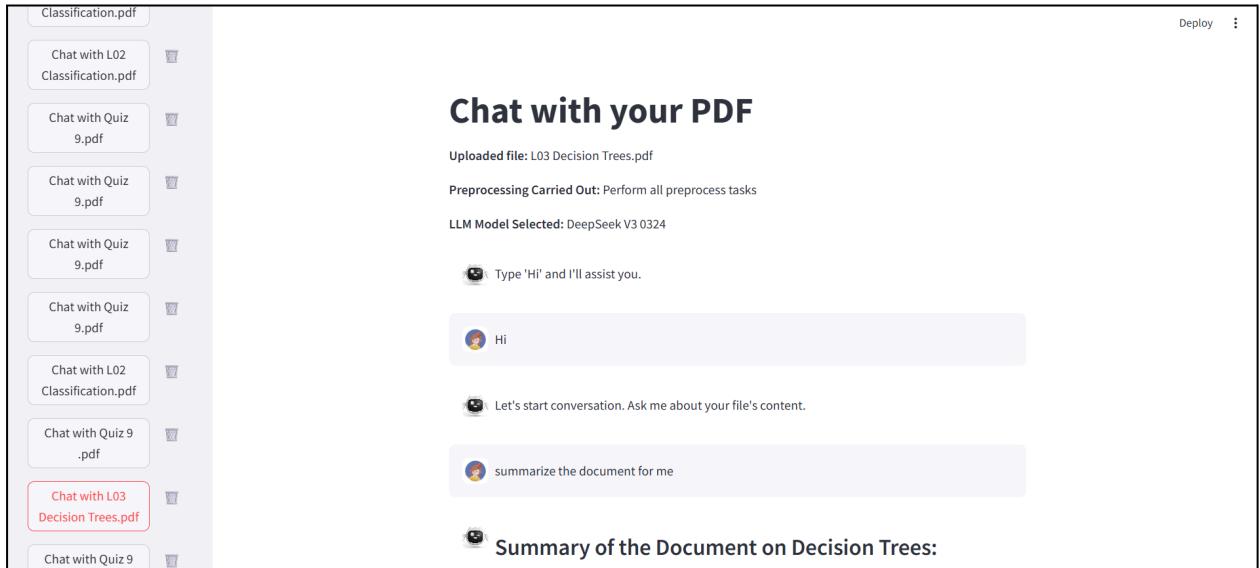
- DeepSeek R1
- DeepSeek V3 0324

**Figure 4.30 User upload an invalid file type and ‘Upload & Process Document’ button not shown**

As the chat application is only accepting the file uploaded in PDF, DOCS or TXT file format, hence the file that is in another format (csv file/json file) will not be accepted. Meanwhile, there is also a restriction on the uploaded file size, in which the user is only allowed to upload a file that does not exceed 200 MB. If the user uploads an invalid file format or the uploaded

file exceeds 200 MB, a message in red color will be displayed. The ‘Upload & Process Document’ button will also not be displayed unless the user uploads a valid file.

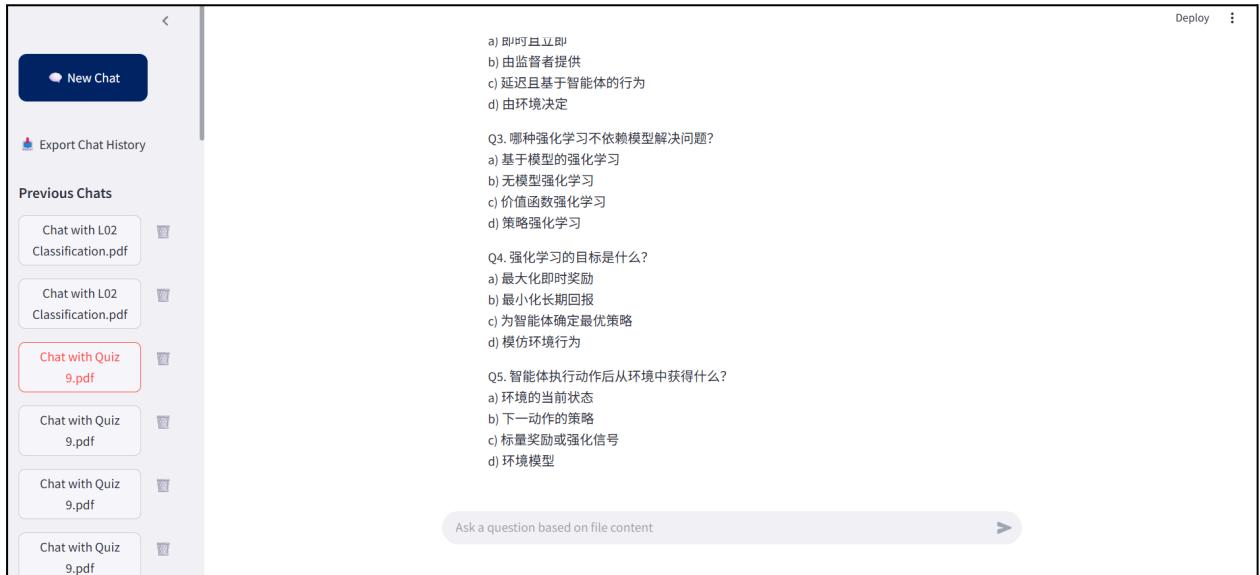
## 4.9 Chat History Management



*Figure 4.31 User click on any chat title and the chat history will be displayed*

This is a past session. To chat with a document, start a new chat session.

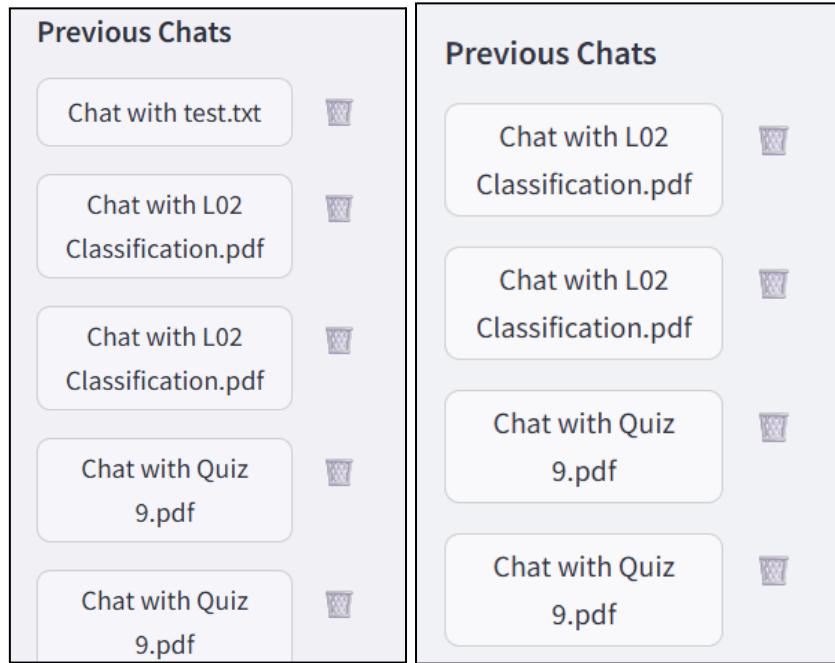
*Figure 4.32 A message that indicate this is a past chat session*



*Figure 4.33 User is allowed to enter input if the chat is still active*

Every chat record will be stored as the history records, and a list of the previous chat records is displayed at the left sidebar. When the user clicks on any chat title at the list, the corresponding chat history will be displayed. If the chat is an inactive chat, which means the

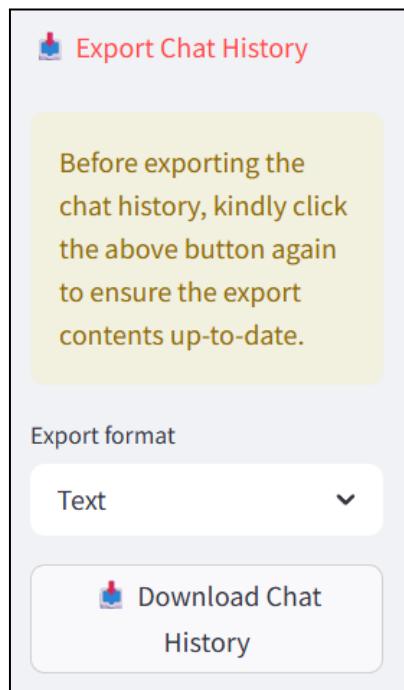
history is loaded from the database, a message (**Figure 4.32**) will be displayed to inform the user that this is a past chat session that is not allowed for continuing chatting. If the chat is an active chat, the user is allowed to continue the chatting session. The active chat means those chat within the same session while the web browser is opening. For instance, the user first starts a conversation with file A, then the user clicks the new chat button to start a new conversation with file B, when the user clicks back to chat with file A, the user is allowed to continue the chatting as it is within the same session.



**Figure 4.34 The chat history list before and after deleting the ‘Chat with test.txt’**

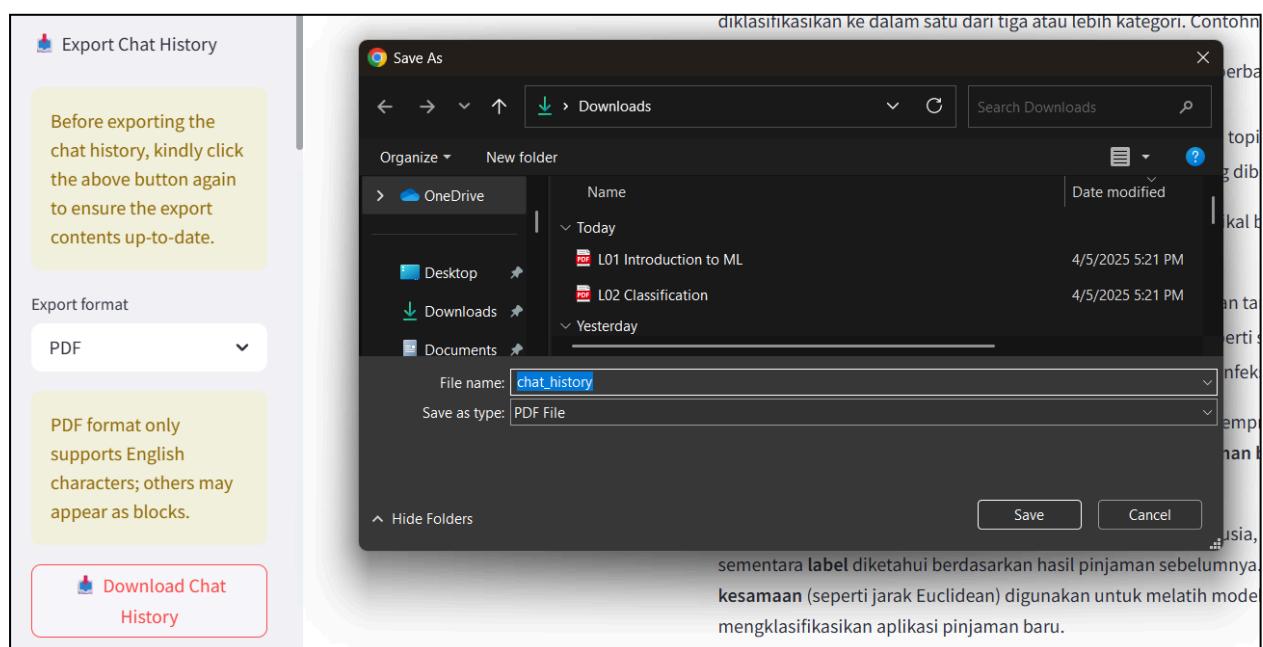
The user is also allowed to delete the previous chat history record by simply clicking on the trash can icon beside the chat title.

## 4.10 Export Chat Record



**Figure 4.35 Export Option show after clicking the button**

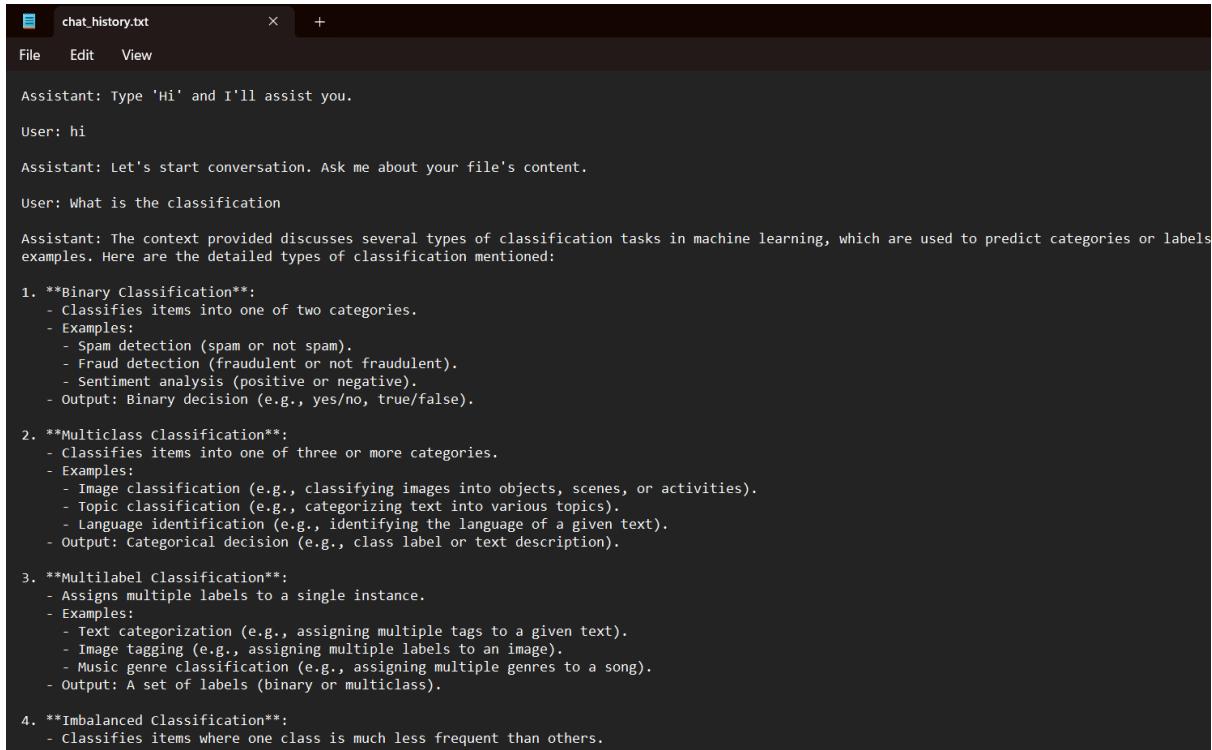
When the user clicks on the ‘Export Chat History’ button, the export option will be displayed, along with a reminder message for the user to refresh the download record before exporting it. The user is given the option to export the chat record in text, json or pdf format.



**Figure 4.36 A message displayed when export format is pdf and Action for downloading the chat record by clicking the ‘Download Chat History’ button**

For exporting chat records in pdf format, a message will be displayed to inform the user that the other characters beside English character will be displayed as the blocks in the downloaded pdf file.

The figures following show the sample exported chat record in the three available export formats.



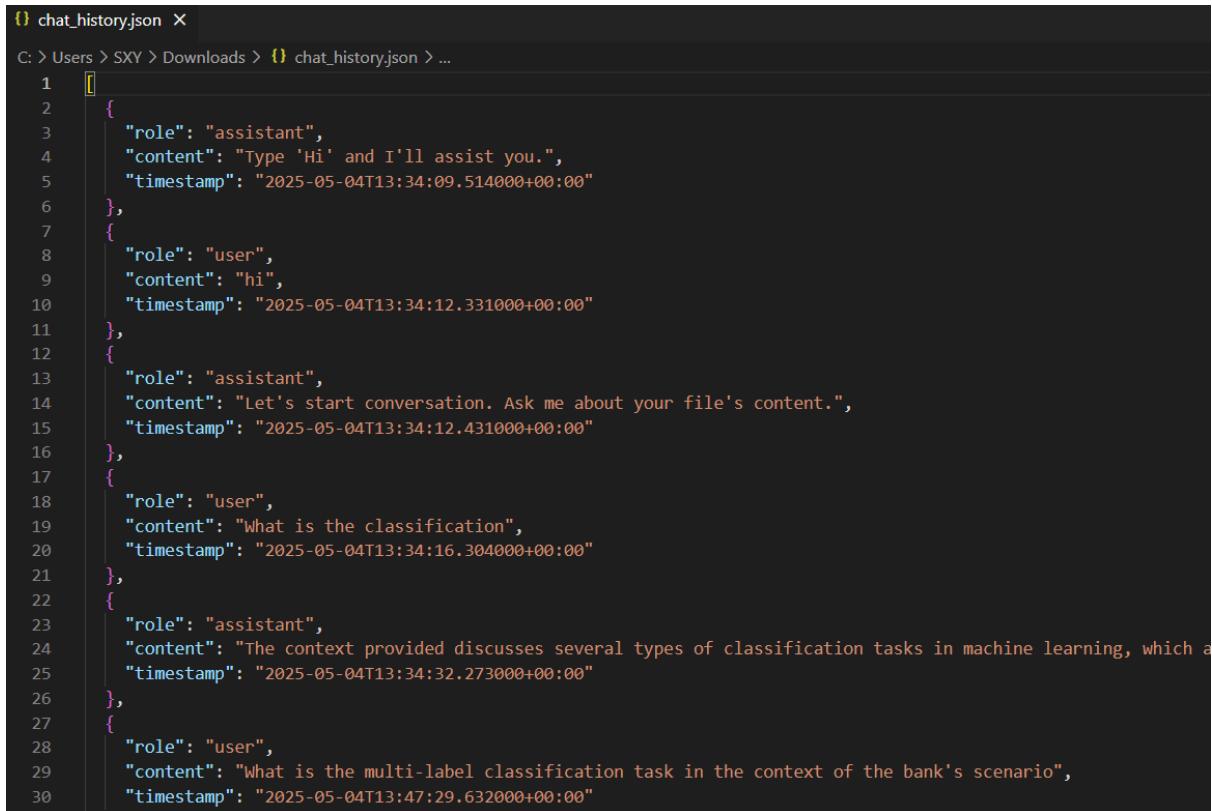
```
chat_history.txt
File Edit View
Assistant: Type 'Hi' and I'll assist you.
User: hi
Assistant: Let's start conversation. Ask me about your file's content.
User: What is the classification
Assistant: The context provided discusses several types of classification tasks in machine learning, which are used to predict categories or labels examples. Here are the detailed types of classification mentioned:
1. **Binary Classification**:
   - Classifies items into one of two categories.
   - Examples:
     - Spam detection (spam or not spam).
     - Fraud detection (fraudulent or not fraudulent).
     - Sentiment analysis (positive or negative).
   - Output: Binary decision (e.g., yes/no, true/false).

2. **Multiclass Classification**:
   - Classifies items into one of three or more categories.
   - Examples:
     - Image classification (e.g., classifying images into objects, scenes, or activities).
     - Topic classification (e.g., categorizing text into various topics).
     - Language identification (e.g., identifying the language of a given text).
   - Output: Categorical decision (e.g., class label or text description).

3. **Multilabel Classification**:
   - Assigns multiple labels to a single instance.
   - Examples:
     - Text categorization (e.g., assigning multiple tags to a given text).
     - Image tagging (e.g., assigning multiple labels to an image).
     - Music genre classification (e.g., assigning multiple genres to a song).
   - Output: A set of labels (binary or multiclass).

4. **Imbalanced Classification**:
   - Classifies items where one class is much less frequent than others.
```

**Figure 4.37 Exported Chat Records in Text format (txt)**



```
chat_history.json
C: > Users > SXY > Downloads > chat_history.json > ...
1 [
2 {
3   "role": "assistant",
4   "content": "Type 'Hi' and I'll assist you.",
5   "timestamp": "2025-05-04T13:34:09.514000+00:00"
6 },
7 {
8   "role": "user",
9   "content": "hi",
10  "timestamp": "2025-05-04T13:34:12.331000+00:00"
11 },
12 {
13   "role": "assistant",
14   "content": "Let's start conversation. Ask me about your file's content.",
15   "timestamp": "2025-05-04T13:34:12.431000+00:00"
16 },
17 {
18   "role": "user",
19   "content": "What is the classification",
20   "timestamp": "2025-05-04T13:34:16.304000+00:00"
21 },
22 {
23   "role": "assistant",
24   "content": "The context provided discusses several types of classification tasks in machine learning, which are used to predict categories or labels examples. Here are the detailed types of classification mentioned:",
25   "timestamp": "2025-05-04T13:34:32.273000+00:00"
26 },
27 {
28   "role": "user",
29   "content": "What is the multi-label classification task in the context of the bank's scenario",
30   "timestamp": "2025-05-04T13:47:29.632000+00:00"
```

**Figure 4.38 Exported Chat Records in json format**

## Chat History

**File Name:** L02 Classification.pdf

**Chat ID:** JK0kvly0WbxKq1vCS1l9

**Exported On:** 2025-05-04 22:32:27

**Assistant:**

Type 'Hi' and I'll assist you.

**User:**

hi

**Assistant:**

Let's start conversation. Ask me about your file's content.

**User:**

What is the classification

**Assistant:**

The context provided discusses several types of classification tasks in machine learning, which are used to predict categories or labels for new observations based on previous examples. Here are the detailed types of classification mentioned:

1. \*\*Binary Classification\*\*:

- Classifies items into one of two categories.
- Examples:
  - Spam detection (spam or not spam).
  - Fraud detection (fraudulent or not fraudulent).
  - Sentiment analysis (positive or negative).
  - Output: Binary decision (e.g., yes/no, true/false).

*Figure 4.39 Exported Chat Records in pdf format*

## 4.11 Uploaded File Preview

Within the chat session, the user is also given the option to preview the uploaded file content by simply clicking on the 'Preview File' button beside the uploaded file name at the above of the chat session (the area that shows uploaded file name, preprocessing option and LLM model selected).

If the uploaded file is a pdf file, a zoom slider will be shown for allowing the user to zoom on the picture previewing of the uploaded pdf file. A download button is also available for the user when they want to view the details of the uploaded file without the need to manually search the file at the file storage in the devices.

**Uploaded file:** L02 Classification.pdf

**Preview File**

Zoom Level



*Figure 4.40 Zoom Slider for previewing uploaded pdf file*

Download PDF

Download to view detail.

BMCS2114 Machine Learning

L02: Classification

Classification predicts the class or category of a new observation based on previous examples. Several key elements are needed to develop a classification model:

1. **Features that can be quantified:** Identify the relevant features or variables that can help distinguish between different classes. These features, such as numerical or categorical data, should be measurable and quantifiable to facilitate analysis.
2. **Labels are known:** In supervised machine learning, labelled data is required to train the model. These labels should accurately reflect the true class or category of the observations, allowing the model to learn from these examples and make predictions on new data.
3. **Method to measure similarity:** measure the similarity or distance between observations, such as *Euclidean distance, cosine similarity, or Jaccard similarity*.

Figure 4.41 Preview for the uploaded pdf file

Chat with your AI

Uploaded file: L02 Classification.pdf

Zoom Level: 100

Download PDF

Download to view detail.

File name: L02 Classification

Save as type: PDF File

Name	Date modified	Type	Size
.ipynb_checkpoints	2/5/2025 6:40 PM	File folder	
__pycache__	4/5/2025 12:30 AM	File folder	
temp_uploads	4/5/2025 10:18 PM	File folder	

Save Cancel

Figure 4.42 Action of Downloading the uploaded pdf file by clicking the button

If the uploaded file is a text file, the preview will be showing the first 2000 characters of the extracted text from the text file.

Uploaded file: test.txt

Preview File

apple is me apple is you

Figure 4.43 Preview for Text File

## 4.12 User Feedback and Satisfaction

For better evaluation of the chat application developed, we construct a questionnaire and distribute it to friends and family members. We first let them test on the chat application, upload files and try to communicate with it, or perform any available feature of the chat application. After they have done testing, they will fill in the questionnaire based on the previous interaction with the chat application. The questionnaire that we conducted is mainly focused on some few aspects of the chat application, in which the chatbot's performance, how well the chatbot understands the user's request, UI design and the response time.

### **Questions asked in the Google Form:**

- Q1 : What is your educational background?
- Q2 : How do you think our system performance?
- Q3 : Rate the usefulness of our file chatbot
- Q4 : Was the chatbot easy to use?
- Q5 : Did the chatbot understand and process your file correctly?
- Q6 : Were the chatbot's answers accurate and relevant to your file?
- Q7 : How do you think our UI design?
- Q8 : How quickly did the chatbot respond to your queries?
- Q9 : Did you encounter any errors or bugs?
- Q10 : If yes, describe what happened
- Q11 : How would you rate your overall experience with the chatbot?

### **The Response Collected from Non-IT background:**

Q1: Others

*Table 4.2 Respondents from Non-IT Background*

Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
5	5	Yes	Yes	Yes	5	Very fast	No		5
5	5	Yes	Yes	Yes	4	Very fast	No		4
4	3	Yes	Yes	Yes	3	Just right	No		5
5	4	Yes	Yes	Yes	4	Very fast	No		5
4	5	Yes	Yes	Yes	4	Very fast	No		5
4	5	Yes	Yes	Yes	4	Just right	No		5
5	5	Yes	Yes	Yes	5	Very fast	No		5
5	5	Yes	Yes	Yes	5	Very fast	No		5
4	5	Yes	Yes	Yes	5	Very fast	No		4
5	3	Yes	Yes	Yes	3	Just right	No		4
4	4	Yes	Yes	Yes	4	Just right	No		5
4	5	Yes	Yes	Yes	5	Just right	No		4
5	5	Yes	Yes	Yes	5	Just right	No		5
4	4	Yes	Yes	Yes	4	Too slow	No		4

5	4	Yes	Yes	Yes	4	Very fast	No		4
---	---	-----	-----	-----	---	-----------	----	--	---

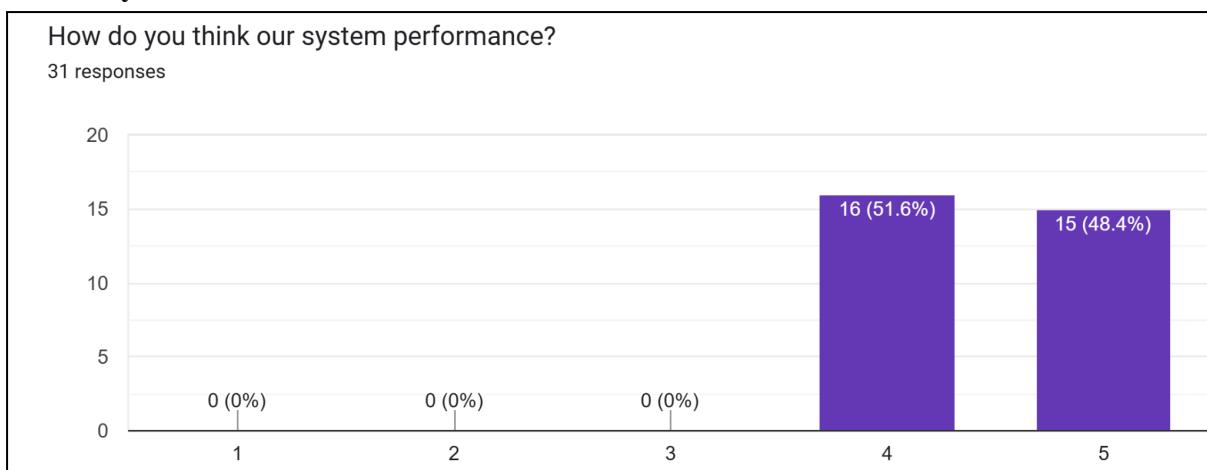
### **The Response Collected from IT background:**

Q1: IT related

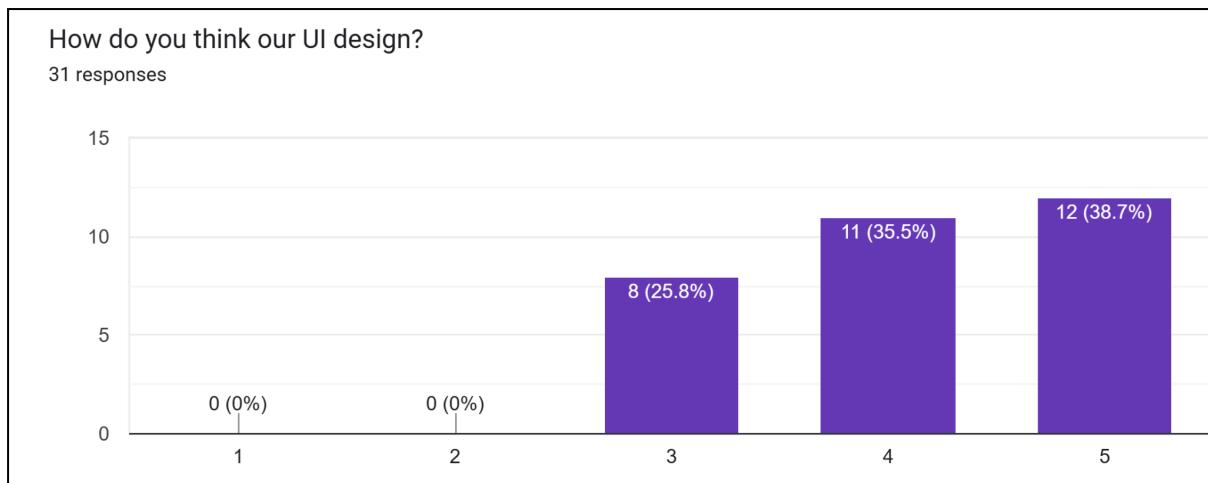
***Table 4.3 Respondents from IT Background***

Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
5	5	Yes	Yes	Yes	5	Very fast	No		5
4	4	Yes	Yes	Yes	3	Just right	No		5
4	5	Yes	Yes	Yes	3	Very fast	No		4
4	5	Yes	Yes	Yes	3	Very fast	Yes	when preview pdf, if zoom out, then the pdf preview will be disappear	4
5	5	Yes	Yes	Yes	5	Just right	No		5
5	5	Yes	Yes	Yes	4	Just right	No	D:	5
4	5	Yes	Yes	Yes	4	Just right	No		4
4	4	Yes	Yes	Yes	4	Just right	No		4
4	5	Yes	Yes	Yes	3	Very fast	No		4
5	5	Yes	Yes	Yes	4	Very fast	No		5
4	5	Yes	Yes	Yes	3	Very fast	No		4
5	4	Yes	Yes	Yes	3	Very fast	No		4
5	5	Yes	Yes	Yes	5	Very fast	No		4
4	4	Yes	Yes	Yes	5	Just right	No		5
4	3	Yes	Yes	Yes	5	Just right	No		4
5	5	Yes	Yes	Yes	5	Very fast	No		5

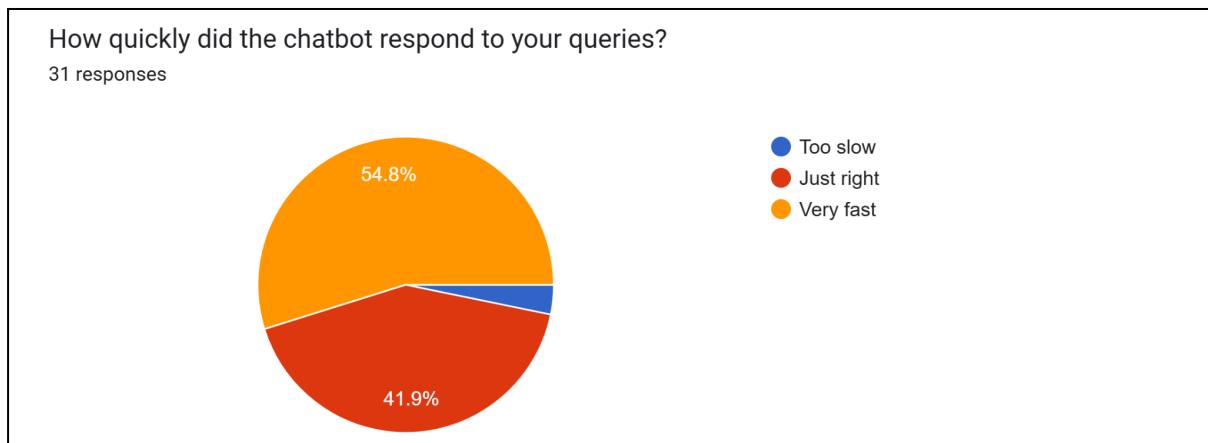
### **Summary Chart:**



***Chart 4.1 Rate on System Performance***



**Chart 4.2 Rate on UI Design**



**Chart 4.3 Proposition of Response Time**

The final responses that we were able to collect are 31 responses, with 15 respondents from non-IT background, the other 16 respondents are from the IT background. Based on the feedback, most of the respondents rate above 3 for the system performance, which means that the chat application is able to work well and perform what it requires to be performed. Although one respondent reports that he/she faces a bug during testing the chat application, the overall feedback from the other respondents shows that there is no major bug in the chat application. By looking at the bug that was reported by the respondents, after we check through the chat application, we have found the problem and solved it. In the UI design aspect, the average score is around 4.13, which means that the UI design can still be modified to become better. In the aspect of response time, most of the respondents think that the chat application is able to provide response very fast or the speed for the chatbot to generate response is just right. Overall, based on the collected feedback, we can conclude that the current chat application has performed well on what it should be performed. However, as this is still a prototype, and the respondents are tested with the guidance from us, it may cause bias on testing the context that can be easily identified, but those special or complex contexts may be ignored. Hence, although the overall feedback is good, this chat application still requires further improvement so that it can perform well when deployed into the real world scenario.

## 4.14 Test Case

The table below shows some of the test cases that were carried out to test the chatbot's features. The result shows that the chatbot's functionality works well, ensuring high usability and reliability for diverse use cases.

No	Description	Precondition	Steps	Expected Results	Actual Results	Status
1	Test file upload and processing	Ensure file (PDF, DOCS, TXT) is available	1. Open Chatbot 2. Upload a supported file 4. Confirm the successful upload	File is uploaded successfully and content is accurately extracted	User is able to upload the file in PDF, DOCS, or TXT successfully	PASS
2	Test summarization feature	A valid file has been uploaded	1. Upload a file 2. Ask for summarization ('Summarize the content')	Chatbot provides an accurate and precise summary.	A precise summary is generated strictly based on the uploaded file context and able to provide the important points	PASS
3	Test multilingual translation	A valid file has been uploaded	1. Upload a file 2. Ask for translation ('Translate the content into Chinese')	Chatbot provides a translated content that is correct and follows the requested language	The chatbot is able to translate the content of which languages that user want correctly	PASS
4	Test error handling for unsupported files	An unsupported file is available	1. Open Chatbot 2. Attempt to upload an unsupported file (csv file)	A message will be shown to the user that the file format is unsupported and the upload button will not be displayed	User unable to upload the file in any other formats beside of PDF, TXT and DOCS	PASS
5	Test chat history saving and exporting	Chat Record Exists	1. Interact with the chatbot 2. Export the chat history as PDF/text/json	Chat history is saved accurately and the file is generated correctly.	Users are able to save the chat history in pdf, text or json format.	PASS
6	Preview Uploaded Text	A valid file has been uploaded	1. Start a chat session 2. Click the preview file button	The preview content for the uploaded file will be displayed	User is able to view the content of the uploaded file at the chatbot UI	PASS

# **5. Discussion and Conclusion**

## **5.1 Achievement of Objectives**

Our project successfully achieved its core objectives by developing an interactive chat application that transforms static documents into dynamic conversational partners. Our file chatbot allows users to upload files in PDF, DOCX, or TXT formats and engage with the content through natural language queries. A user-friendly interface, built using Streamlit, ensures seamless file uploads and real-time interactions, including an interactive file preview feature that supports zooming in and out for better readability and an option to download the original PDF. The chatbot provides accurate responses, supporting tasks such as summarization, translation, precise information extraction from the uploaded documents. The multilingual translation feature operates exclusively through conversational commands, users can directly instruct the chatbot to translate specific content (e.g., "Summarize the pdf written in chinese"). This on-demand approach enhances accessibility and efficiency for users, particularly non-native speakers and busy professionals.

To maintain focus, the system restricts interactions to questions related to the uploaded file, ensuring relevance. Each chat session is tied to a single document, with new files requiring a new session. The interface allows users to start a new chat while mid-conversation, with the ability to return and continue the previous chat seamlessly as long as the user does not close the web browser as the chat session will stay active for the same browsing session. Chat history is stored using Firebase, allowing users to review past conversations, though continuity is limited to the current session for security and simplicity. The application also handles edge cases gracefully, including unsupported formats, unreadable text, and ambiguous queries, while maintaining low-latency responses for a smooth user experience. Preliminary testing indicates high satisfaction with the chatbot's accuracy and usability. Overall, the project delivers on its promise of bridging the gap between static documents and interactive AI-driven conversations, setting a foundation for future enhancements such as cross-session persistence and advanced multimodal support.

## **5.2 The Significance of Result**

The successful implementation of our interactive file chatbot represents a transformative advancement in human-document interaction, delivering both practical utility and technical innovation. By converting static files (PDF, DOCX, TXT) into dynamic conversational partners, our solution redefines traditional document processing, shifting from passive reading to active, intelligent engagement. This paradigm shift not only enhances efficiency but also democratizes access to complex documents, enabling users of all technical backgrounds to interact with content naturally and intuitively.

A key strength of our system is its multilingual capability, which effectively eliminates language barriers. The on-demand translation feature proves invaluable for international business users and researchers working with foreign-language materials, fostering seamless cross-lingual collaboration. Furthermore, our carefully designed interface, featuring intuitive chat switching and interactive document preview, ensures a fluid user experience that maintains workflow continuity without disruption.

From a technical standpoint, our chatbot demonstrates remarkable versatility, handling tasks ranging from summarization to precise data extraction with high accuracy. The architecture achieves an optimal balance between usability and security, leveraging Firebase integration for convenient chat history access while safeguarding sensitive information.

Ultimately, this innovation paves the way for a new era of document interaction, one where information is not just stored, but truly understood and utilized with unprecedented ease.

## **5.3 Limitations**

Our current prototype has several limitations that impact its functionality and usability. First, the absence of login and logout functionality means chat histories are not privately tied to individual accounts, raising privacy concerns for sensitive documents. While the system stores chat history for review, users cannot resume or build upon past conversations, they may only view previous interactions without continuing the dialogue. Users are also not able to view and download their originally uploaded files within the chat history. This lack of contextual continuity forces users to restart discussions even when referencing earlier chats, disrupting workflow efficiency.

Additionally, the system restricts interactions to one file per session. Switching documents requires starting a new chat, further fragmenting the user experience. On the technical side, the chatbot processes only textual content, ignoring non-text elements like images, tables, or diagrams. This significantly limits its utility for files where visual data is critical, such as research papers with figures or financial reports with embedded charts. Moreover, the current technique used does not support the chatbot to answer the user's request by considering also the past few questions. Each request for the user is only considered as a single part, without considering the previous question about what. Hence, it may require the user to restate the prerequisite first before asking for a new response. This is causing a time-consuming problem and is not user-friendly. These constraints highlight areas for future improvement to enhance user experience and functionality.

## **5.4 Future Improvements**

Our file chatbot system will go through several major improvements to significantly expand its capabilities and user experience. A foundational improvement will be the implementation of secure login and logout functionality with user authentication, enabling personalized sessions while ensuring data privacy. This authentication system will integrate with cloud-based storage to maintain complete chat histories across sessions, allowing users to not just view but actively continue previous conversations with full contextual understanding. We plan to introduce multi-document processing capabilities, enabling users to upload and query multiple files simultaneously for cross-document analysis without interrupting their workflow. The system's contextual awareness will be substantially upgraded through enhanced memory architecture, preserving conversation threads and references across sessions. To handle large-scale documents, we will optimize backend processing using advanced vectorization techniques and intelligent chunking strategies for efficient handling of 100+ page files. Recognizing the need for reliable access, we will develop an offline mode that maintains core functionality during internet disruptions. Moreover, we are also considering to include the OCR (Optical Character Recognition) technology in our chat application so that the chatbot can not only understand the textual content, it can also analyze the image and the table that is included in the file. These collective improvements, particularly the continuous chat experience and context preservation features, are designed to eliminate current workflow interruptions while delivering more natural, productive document interactions. The upgraded system will not only address existing limitations but also establish a foundation for future expansions in document intelligence and multimodal processing capabilities.

# References

- AskYourPDF: The Best PDF AI Chat App.* (2024). AskYourPDF. <https://askyourpdf.com/>
- Bouchra El Bakkouri. (2022, August 12). *The role of Chatbots in Enhancing Customer Experience: Literature Review.* Procedia Computer Science. <https://www.sciencedirect.com/science/article/pii/S1877050922006627>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., & Hesse, C. (2020). Language Models Are Few-Shot Learners. *Arxiv.org*, 4. <https://arxiv.org/abs/2005.14165>
- Chatbot | Definition, History, & Facts | Britannica.* (2024, March 13). Www.britannica.com. <https://www.britannica.com/topic/chatbot>
- ChatPDF - Chat with any PDF!* (2023). Www.chatpdf.com. <https://www.chatpdf.com/>
- Deepseek vs. Chatgpt vs. Gemini: Best AI model compared.* 365mechanix. (2025, February 18). <https://www.365mechanix.com/blogs/2025/deepseek-vs-chatgpt-vs-gemini-which-ai-model-is-best/>
- Deepseek. (n.d.). *DeepSeek v3 0324 (free) - API, providers, stats.* OpenRouter. <https://openrouter.ai/deepseek/deepseek-chat-v3-0324:free>
- Deepseek. (n.d.-b). *R1 (free) - API, providers, stats.* OpenRouter. <https://openrouter.ai/deepseek/deepseek-r1:free>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018, October 11). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* ArXiv.org. <https://arxiv.org/abs/1810.04805>
- DocChat: An information retrieval approach for chatbot ... (n.d.). <https://aclanthology.org/P16-1049.pdf>

Guinness, H. (2025, February 19). *The Best Large Language Models (llms) in 2025*. Automate your work today. <https://zapier.com/blog/best-llm/#gemini>

Jurafsky, D., & H. Martin, J. (2018). *Speech and Language Processing*. Stanford.edu. <https://web.stanford.edu/~jurafsky/slp3/>

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). <https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf>

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). Large Language Models: A Survey. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2402.06196>

OpenAI. (2023). *GPTs can read and understand documents*. Retrieved from <https://openai.com/chatgpt>

OpenAI. (2023). *How GPTs work with your files*. Retrieved from <https://openai.com/chatgpt>

Paperpal. (2025, March 20). *Top 7 AI tools for research in 2025 (compared)*: Paperpal Blog. <https://paperpal.com/blog/news-updates/top-7-ai-tools-for-research/amp>

Pogeaant, N. (2023, January 19). *Exploring NLP's performance-evaluation and metrics as the compass*. Medium. <https://medium.com/@npogeaant/exploring-nlps-performance-evaluation-and-metrics-as-the-compass-db2266422130>

*Pretrained models*. Pretrained Models - Sentence Transformers documentation. (n.d.). [https://www.sbert.net/docs/sentence\\_transformer/pretrained\\_models.html](https://www.sbert.net/docs/sentence_transformer/pretrained_models.html)

Rahultiwari. (2024, September 23). *Unlocking the power of sentence embeddings with all-minilm-L6-V2*. Medium. <https://medium.com/@rahultiwari065/unlocking-the-power-of-sentence-embeddings-with-all-minilm-l6-v2-7d6589a5f0aa>

Sangita Pokhrel, Bina K C., & Prashant Bikram Shah. (2021, August 20). *A Practical Application of Retrieval-Augmented Generation for Website-Based Chatbots*:

*Combining Web Scraping, Vectorization, and Semantic Search.* Formplus.  
<https://www.formpl.us/blog/academic-chrome-extensions>

*Sentence-transformers/distiluse-base-multilingual-cased-V2*      *hugging*      *face.*  
sentence-transformers/distiluse-base-multilingual-cased-v2 · Hugging Face. (n.d.).  
<https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2>

Sharma, A. (2025, April 21). *GPT 4 vs GPT 3: Differences you must know in 2025*. K21 Academy. <https://k21academy.com/ai-ml/gen-ai/gpt-4-vs-gpt-3/>

Sunu Jatmika, Syaad Patmanthara, Aji Prasetya Wibawa, & Fachrul Kurniawan. (n.d.). Cognition-based document matching within the chatbot ...  
<https://pdfs.semanticscholar.org/e044/e07f46ca30094e9f4f6a0076e2658de41405.pdf>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*. <https://arxiv.org/pdf/1706.03762>

What is the role of cosine similarity in embeddings?. Milvus. (n.d.).  
[https://milvus.io/ai-quick-reference/what-is-the-role-of-cosine-similarity-in-embeddin  
gs](https://milvus.io/ai-quick-reference/what-is-the-role-of-cosine-similarity-in-embeddings)

XIN TAN. (n.d.). Design and research of intelligent QA system ....PDF - -Orca.  
<https://orca.cardiff.ac.uk/id/eprint/149082/1/Design%20and%20research%20of%20intelligent%20QA%20system%20....pdf>

Yen-Ting Lin, & Yun-Nung Chen. (2021, January 27). *An Empirical Study of Cross-Lingual Transferability in Generative Dialogue State Tracker.*  
<https://arxiv.org/pdf/2101.11360>