Advanced RAG for Financial Information

Submitted in partial fulfillment of the requirements
for the degree of

B.E. Information Technology

By

Sunny Nishar	**211084**
Gaurav Prajapati	**211096**
Rylen Lobo	**221254**
Vinita More	**211076**

Supervisor
Ms. Alvina Lopes
Assistant Professor

Department of Information Technology
St. Francis Institute of Technology
(Engineering College)
An Autonomous Institute, Affiliated to University of Mumbai

2024-2025

SFIT LIBRARY 2025

CERTIFICATE

This is to certify that the project entitled **"Advanced RAG for financial information"** is a bonafide work of **"Sunny Nishar (211084), Gaurav Prajapati (211096), Rylen Lobo (221254), Vinita More (211076)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology.

02-5-2025

Ms. Alvina Lopes

(Project Guide)

Mr. Devesh Rajadhyax

(Industry Mentor)

Dr. Prachi Raut

(Head Of Department)

Dr. Sincy George

(Principal)

SFIT LIBRARY 2025

Project Approval Report for B.E.

This project report entitled *"Advanced RAG for Financial Information"* by ***Sunny Nishar (211084), Gaurav Prajapati (211096), Rylen Lobo (221254) & Vinita More (211076)*** is approved for the degree of ***Bachelors of Engineering in Information Technology***.

1. _____ 2/5/25

2. _____ 02-5-2025 Ms. Alvina Lopes.

Date: 02/05/2025

Place: Borivali, Mumbai

SFIT LIBRARY 2025

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Sunny Nishar (211084)

Gaurav Prajapati (211096)

Rylen Lobo (221254)

Vinita More (211076)

Date: 02/05/2025

SFIT LIBRARY 2025

ABSTRACT

Financial documents are essential for understanding a company's performance, but their interpretation remains challenging due to their large volume, unstructured format, and domain-specific language. Traditional methods and even some Large Language Model (LLM)-based solutions often struggle with issues like hallucination, context loss, and difficulty processing complex elements such as tables and charts. This project presents an advanced Retrieval-Augmented Generation (RAG) based chatbot system specifically designed to overcome these limitations. The system takes financial documents in PDF format as input and leverages a pipeline that includes Query Enhancement, Semantic Chunking, and Reranking to optimize the accuracy and relevance of responses. Using tools like Marker for PDF-to-Markdown conversion, LlamaIndex for parsing and context management, HuggingFace embeddings for semantic understanding, and Pinecone for efficient vector storage, the system delivers accurate, context-aware answers to user queries. By integrating metadata and structuring document content into meaningful units, the proposed solution significantly reduces hallucination, mitigates misinformation risks, and enhances the quality of information retrieval. The result is a robust and intelligent chatbot capable of assisting users in understanding complex financial documents with improved precision and reliability.

SFIT LIBRARY 2025

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Problem Formulation 2

1.3 Description 2

1.4 Motivation 3

1.5 Proposed Solution 3

1.6 Scope of the Project 4

2 Literature Review **6**

2.1 Existing Work 6

2.1.1 Literature review related to existing system/methodology 7

2.1.2 Literature review related to existing algorithms 8

2.1.3 Literature review related to tools/technology/framework 9

2.2 Research Gaps identified 10

3 System Analysis **11**

3.1 Functional Requirements 11

3.2 Non-Functional Requirements 12

3.3 Specific Requirements 13

3.4 Use-Case Diagrams 13

4 Analysis Modeling **20**

4.1 Activity Diagram 20

4.2 Data Flow Diagram 21

4.2.1 DFD:Level 0 21

4.2.2 DFD:Level 1 22

4.3 Timeline Chart 23

5 System Architecture **24**

5.1 System Architecture 24

SFIT LIBRARY 2025

5.2 Graphical User Interface 27

6 Implementation **29**
6.1 Implementation 29
6.1.1 User Authentication with Supabase 29
6.1.2 PDF to Markdown Conversion using Marker 29
6.1.3 Markdown Parsing using LlamaIndex Node Parser 30
6.1.4 Summary Extraction with SummaryExtractor 30
6.1.5 Vector Embedding with HuggingFace Models 30
6.1.6 Vector Storage using Pinecone 30
6.1.7 Query Enhancement 30
6.1.8 Reranking with Cross-Encoder 30
6.1.9 Contextual Chat Engine with LlamaIndex 30
6.2 Application Performance 31
6.3 Code 32

7 Testing **37**
7.1 Test Cases 37
7.2 Testing Methods Used 38
7.2.1 Unit Testing 38
7.2.2 Integration Testing 38
7.2.3 Usability Testing 38
7.2.4 Smoke and Sanity Testing 39

8 Result and Discussion **41**
8.1 Results and Discussion 41
8.1.1 Evaluation Results 41
8.1.2 Metric Definitions and Formulas 42
8.1.3 Discussion 43

9 Result and Discussion **44**
9.1 Conclusion 44
9.2 Future Scope 44

10 Acknowledgement **46**

SHIT LIBRARY 2025

List of Figures

3.1 Use Case Diagram 15

4.1 Activity Diagram for User. 21
4.2 DFD Level 0. 21
4.3 DFD Level 1. 22
4.4 Timeline details and chart 23

5.1 Advanced RAG system 26
5.2 Data extraction in Table format 27
5.3 Factual questions answered by system 28
5.4 Mathematical question answering 28

7.1 Test case structure 40
7.2 Test case results 40

SFIT LIBRARY 2025

List of Tables

2.1 Comparison of methodology 7
2.2 Comparison of algorithms 8
2.3 Comparison of tools/technology/framework 9

3.1 Use Case Diagram Description 14
3.2 Use Case: Login Functionality 16
3.3 Use Case: Input Query 16
3.4 Use Case:Input Document 17
3.5 Use Case: Index Document/ Query 17
3.6 Use Case: Embed Document 18
3.7 Use Case: Update vector database 18
3.8 Use Case: Retrieve documents 19
3.9 Use Case: Display output 19

7.1 Test Cases 37

SFIT LIBRARY 2025

List of Abbreviations

LLM Large Language Model

AI Artificial Intelligence

RAG Retrieval Augmented Generation

PDF Portable Document Format

10-K Annual Report

10-Q Quaterly Report

8-K Current Report

S-1 Registration Statement

SEBI Securities and Exchange Board of India

BSE Bombay Stock Excahnge

NSE National Stock Exchange

OCR Optical Character recognition

QA Question Answering

BLEU Bilingual Evaluation understudy

ROUGE Recall-Oriented Understudy for Gisting Evaluation

API Application Programming Interface

GUI Graphical User Interface

CPU Central Processing Unit

GPU Graphics Processing Unit

RAM Random Access Memory

SITT LIBRARY 2025

Chapter 1

Introduction

1.1 Introduction

In today's data-driven world, information has become a cornerstone for decision-making across all sectors, and the finance sector is no exception. Financial documents serve as a vital source of data, providing a snapshot of a company's financial health, performance, operations, cash flow, and overall standing in the market. These documents, such as 10-K, 10-Q, 8-K, S-1, and Earnings Call Transcripts, are essential for assessing a company's performance. While some financial documents are publicly available on platforms like SEBI, BSE, NSE, and other financial news websites, accessing meaningful insights from them remains a significant challenge. The sheer volume, complexity, and unstructured nature of this data make it difficult for professionals and non-experts alike to derive actionable insights efficiently. The problem is exacerbated by the presence of domain-specific terminologies and financial jargon, making interpretation of these documents a daunting task for individuals outside the finance industry. Furthermore, the manual processing of vast financial data is both time-consuming and prone to errors, with traditional query methods often failing to retrieve the necessary information due to the documents' unstructured nature. Existing Large Language Models (LLMs), although trained to process such documents, struggle with outdated knowledge bases, hallucinations, and lack of contextual awareness, rendering them inefficient for real-time, accurate financial analysis. This paper proposes an Advanced Retrieval-Augmented Generation (RAG) system to enhance the capabilities of financial LLMs by addressing the core issues of hallucination, loss of context, and inadequate handling of complex financial structures like tables and images. By integrating techniques such as query enhancement, semantic chunking, and re-ranking, the proposed system aims to provide accurate, up-to-date, and context-aware answers for financial document queries. Leveraging Pinecone for vector storage and LlamaIndex for structured document parsing, the system will ensure efficient retrieval of relevant data and seamless, coherent responses. Additionally, the Deepeval evaluation framework will be employed to rigorously assess the system's performance, focusing on metrics that gauge both accuracy and contextual

SFIT LIBRARY 2025

relevance. The goal of this system is to significantly improve the accessibility and accuracy of financial information retrieval, enabling stakeholders, investors, and financial analysts to make more informed decisions with confidence. By minimizing the risks of misinformation and improving the precision of retrieved data, this system stands to revolutionize the way financial documents are understood and analyzed in real-time.

1.2 Problem Formulation

Despite advancements in technology, the interpretation and analysis of financial documents remain a complex and tedious task. Companies now make their financial information readily available through digital platforms, promoting transparency and fostering trust among stakeholders. However, with the vast amount of unstructured data and the presence of domain-specific terminologies, extracting meaningful insights from financial documents is challenging. The accuracy of such interpretations is critical, as they significantly influence decisions made by investors and stakeholders. Given this, ensuring accessibility and precision in retrieving and analyzing financial information is of utmost importance in the finance domain, which calls for an advanced system that can effectively manage and process these documents while providing accurate and context-aware insights.

1.3 Description

The financial sector requires timely, accurate, and easily accessible information to make informed decisions. Financial documents, such as annual reports, balance sheets, and quarterly filings, provide critical insights into a company's financial health. However, the process of extracting, interpreting, and analyzing these documents remains complex and labor-intensive. The unstructured nature of these documents, coupled with domain-specific jargon and intricate data like tables, charts, and figures, makes manual analysis prone to errors and inefficiencies.

This project aims to develop an Advanced Retrieval-Augmented Generation (RAG) system for efficient financial information extraction and question answering. The system will leverage Large Language Models (LLMs) and advanced retrieval techniques to process and interpret financial documents accurately and contextually. By integrating PDF-to-Markdown conversion, semantic chunking, vector embeddings, and re-ranking, the system will enhance query understanding and provide precise, up-to-date answers.

The core functionalities of this system include:

- Financial Document Preprocessing – Converting financial reports from PDF to structured Markdown format and breaking them into semantic chunks for efficient processing.
- Semantic Search & Contextual Retrieval – Enabling users to query financial data and

CET LIBRARY 2025

retrieve relevant insights based on vector-based similarity search, enhancing accuracy and context-awareness.

- AI-Driven Query Enhancement & Re-ranking – Improving query understanding and re-ranking potential answers to ensure the retrieval of the most relevant, contextually accurate information.
- Contextual Chat Interface – Providing users with an intuitive platform to interact with financial data, allowing for conversational queries and responses with high contextual fidelity.

By combining these techniques, the system will streamline the process of retrieving and analyzing financial documents, ultimately empowering analysts, investors, and stakeholders to make more informed decisions based on precise, context-aware financial data.

1.4 Motivation

Traditional financial reporting methods often lack clarity and accessibility, resulting in users feeling overwhelmed by complex data and terminology. This project aims to develop an Advanced Retrieval-Augmented Generation (RAG) system tailored for financial information to tackle these critical challenges. The main aim of using Advanced RAG is to enable the integration of Large Language Models with proprietary data from organizations, allowing for detailed insights into a company's finances based on its annual financial reports. By leveraging AI, the Advanced RAG system will provide an intuitive platform for users to easily extract, analyze, and understand company financial documents, such as annual reports. This innovative approach will simplify the interpretation of financial data, enabling both individual investors and institutions to make smarter, more informed investment decisions.

1.5 Proposed Solution

The proposed solution is a web-based Retrieval-Augmented Generation (RAG) system tailored for intelligent interaction with financial documents. It combines document parsing, semantic chunking, vector-based retrieval, and LLM-powered response generation into a unified architecture.

Key components of the solution include:

- **Document Conversion:** Financial PDFs are converted to structured Markdown for easier parsing.
- **Semantic Indexing:** Parsed documents are chunked and embedded using transformer-based models, with vectors stored in Pinecone for efficient semantic search.

SIT LIBRARY 2025

- **Enhanced Querying:** User queries are improved and matched with relevant document segments using similarity search and reranked for contextual relevance.
- **LLM-based Response Generation:** Retrieved content is used to generate accurate and context-aware answers via a large language model.
- **Conversational Interface:** A chat interface supports multi-turn interactions, enabling seamless document querying.
- **User Data Management:** Supabase is used for secure user authentication, and to persist chat history and document metadata, ensuring personalized and traceable interactions.

This architecture ensures reliable, scalable, and secure financial information access, supporting better decision-making through enriched document understanding.

1.6 Scope of the Project

This section outlines the scope of the Advanced Retrieval-Augmented Generation (RAG) system for financial information retrieval. The system is designed to provide accessible and efficient financial insights for the general public, investors, and small businesses by analyzing annual reports. While the project offers a robust solution, there are certain aspects that are deliberately excluded from its current capabilities.

- **Target Users:** The system is intended for individuals, small businesses, and investors who need quick access to financial insights without relying on expert financial advisors. It caters to users who may not have the budget for high-end solutions but need help in understanding financial documents.
- **Document Support:** The system supports only **annual reports**, which are static documents uploaded by the user. Any other types of financial documents, such as quarterly reports, filings, or investment presentations, are not supported in the current version of the system.
- **Data Understanding Capabilities:** The system processes and interprets **textual content**, **tables**, and **images** within uploaded documents. However, it does not handle **graphs, charts, or any visual analytics** due to limitations in displaying graphical content in the current Next.js version. This functionality will not be supported until the issue is resolved.
- **Language Support:** The system is designed to support **English language only**. Non-English documents or queries will not be processed, making it unsuitable for non-English speaking users at this time.

SIT LIBRARY 2025

- **Offline and Web-Based Functionality:** The system operates based on uploaded static reports and does not perform **web-based searches** or pull information from external sources. It cannot provide real-time market data or respond to queries outside the scope of the uploaded documents.
- **Multimodal Input Limitations:** The current system does not support **scanned documents**, **handwritten notes**, or any other form of input that requires **Optical Character Recognition (OCR)**. Only digital, text-based PDFs are accepted.
- **Cross-Device Synchronization:** The system is not equipped with **cloud synchronization** features, and users cannot access their chat history or documents across multiple devices. The platform does not support **collaborative features**, meaning users cannot share documents or insights with others within the system.
- **Real-Time Data Retrieval:** The system is designed for static document analysis and does not provide real-time data updates or streaming. It cannot handle dynamic, real-time data retrieval or updates from external sources.
- **Limitations in Answering Queries:** The system does not perform inference based on information outside the uploaded financial documents. It will not provide answers or insights beyond the scope of the provided reports and will not attempt to extrapolate or guess information beyond its dataset.

SFIT LIBRARY 2025

Chapter 2

Literature Review

SEIT LIBRARY 2025

2.1 Existing Work

Paper [1] advances RAG techniques for financial QA by addressing limitations like ineffective chunking and relevance-based retrieval. It proposes improved chunking, query expansion, metadata use, and re-ranking to boost response accuracy and contextual integrity. System [2] applies a naive RAG method for diagnosing coffee leaf diseases but suffers from poor retrieval quality, resulting in unreliable recommendations. Paper [3] uses RAG with Wikipedia data to improve science exam QA. While effective for typical queries, it struggles with atypical questions, emphasizing the need for more diverse training data. Study [4] evaluates retrieval strategies, advocating element-based chunking for financial documents using the FinanceBench dataset, but notes a lack of domain-specific chunking research. Paper [5] proposes MedGraphRAG for medical QA using hierarchical knowledge graphs. However, high computational cost and lack of response transparency reduce its practicality.

Article [6] highlights re-ranking's role in refining retrieval, but notes added system complexity and resource demands as trade-offs. Article [7] emphasizes the importance of multimodal RAG for processing complex documents with text, tables, and visuals. Integration across modalities remains challenging due to varied model requirements. System [8] by Tee Meng Kiat uses Advanced RAG to answer basic financial queries, focusing on term clarification and simple calculations for user-friendly explanations. Article [9] discusses Advanced RAG components and contrasts them with naive RAG but lacks implementation details and evaluation metrics. Article [10] introduces LlamaIndex for building RAG pipelines, offering useful tools but posing challenges for beginners due to complex workflows and debugging difficulties. Article [11] presents Langchain, a modular framework for LLM applications. While flexible, its poorly maintained community packages and risk of data loss pose development risks.

2.1.1 Literature review related to existing system/methodology

Table 2.1: Comparison of methodology

Sr. No.	Title of Paper	Review	Analysis/Limitations
1	Improving Retrieval for RAG based Question Answering Models on Financial Documents	The paper highlights limitations in evaluating RAG models, as existing metrics (like BLEU and ROUGE) do not capture the semantic accuracy of complex answers [1].	Standard chunking approaches in RAG pipelines often break documents into uniform chunks without considering the document's structure, which can result in critical information loss
2	Overcoming LLM Challenges using RAG-Driven Precision in Coffee Leaf Disease Remediation	The system is developed with an aim to predict disease on coffee leaves and give remediation measures for the same [2].	The naive RAG approach relies on basic retrieval methods that may not effectively fetch the most relevant documents, lowering response quality.
3	Science Exam Question Answering based on Retrieval Augmented Generation	RAG is used to tackle the problem of limited training data of LLM. It uses a huge dataset from Wikipedia to help the LLMs find relevant information [3].	While the approach is designed to adapt to a diverse array of questions, it may struggle with those significantly different from the training data.
4	Financial Report Chunking for Effective Retrieval Augmented Generation	The methodology evaluates chunking strategies for financial reports, focusing on an element-based approach that divides documents into chunks by structure to enhance RAG performance [4].	The element-based chunking may still face challenges in ensuring factual accuracy and relevance in generated responses, particularly with complex financial data.

SFIT LIBRARY 2025

2.1.2 Literature review related to existing algorithms

Table 2.2: Comparison of algorithms

Sr. No.	Title of Paper	Review	Analysis/Limitations
1	Medical Graph RAG: Towards Safe Medical Large Language Model Via Graph Retrieval Augmented Generation.	The system uses a RAG framework to enhance LLMs for private medical data, boosting response accuracy and relevance. It employs document segmentation, entity extraction, and a hierarchical knowledge graph to connect user data with core medical knowledge [4].	The current approach to building and merging knowledge graphs in MedGraphRAG is computationally intensive and inefficient, limiting the system's responsiveness.The system fails to provide clear explanations for how responses are generated.
2	Enhancing Advanced RAG Systems Using Re-ranking with LangChain	Re-ranking involves re-evaluating and rearranging the retrieved documents or data based on their relevance to the query. It prioritizes retrieval results most relevant to the query[5].	Re-ranking adds complexity to the system, requiring more computational power and memory, which can increase response times and operational costs, especially in large-scale applications.
3	Multi-Modal Retrieval-Augmented Generation (RAG) Systems: An In-Depth Overview and Implementation	This system enhances the capability to extract meaningful insights from complex documents, such as PDFs, which often contain a mixture of textual, tabular, and visual information [6].	Multi-modal RAG systems need high-quality, diverse data; poor data degrades performance, and varying modalities may require different architectures, complicating integration.
4	Mengtee	It is a system that uses Advanced RAG to explain financial documents. It uses llama index, Vectara and Together AI API [8].	The system answers basic questions, retrieves requested values, and performs simple calculations using the provided document.

SFIT LIBRARY 2025

2.1.3 Literature review related to tools/technology/framework

Table 2.3: Comparison of tools/technology/framework

Sr. No.	Title of Paper	Review	Analysis/Limitations
1	Advanced RAG Techniques: Unlocking the Next Level	The article focuses on Advanced RAG and the techniques used under it. It also brings to light the drawbacks of Naive RAG.It explains the components of Advanced RAG and also describes the techniques to enhance the output generated by Advanced RAG [9].	The article is limited to the description and does not state how the techniques can be implemented, also there is no information about the accuracy of the techniques mentioned.
2	LlamaIndex	LlamaIndex is a framework designed to help developers create and manage large language models effectively, particularly in the context of retrieval-augmented generation (RAG). The website states what tools LlamaIndex provides and also guides through the implementation [10].	The overall workflow of LlamaIndex is complex and hence difficult for beginners. Also due to complex workflows debugging issues related to data retrieval and context augmentation can be challenging.
3	Langchain	Langchain is a framework for building applications with large language models (LLMs). It offers a modular architecture for language models, retrieval systems, agents, and memory, facilitating the creation and production of data-aware LLM applications[11].	Langchain includes community packages that are often poorly maintained, leading to potential compatibility issues and bugs. Additionally, its ability to connect with external sources may permanently alter them, risking data loss.

SFIT LIBRARY 2025

2.2 Research Gaps identified

- The standard RAG pipeline does not give satisfactory results due to efficient chunking of documents [1].
- The naive RAG approach lacks the capability to integrate different types of data (e.g., text, images), limiting its overall effectiveness in providing comprehensive recommendations [2].
- The system in [3] has a complex implementation The methodology involves multiple steps and components, such as encoding, retrieval, and prompt construction, which could complicate the implementation and require careful tuning.
- The use of knowledge graph to provide more contextual answers poses issues of scalability also no reasoning behind the answer received reduces the trust of users [5].
- Integrating re-ranking techniques can result in latency due to additional processing steps [6].
- The system requires exceptionally high computational resources due to integration of multi-modal advanced retrieval augmented generation [7].
- The application in [8] is trained only on 3 financial reports which is quite less as there are numerous conditions that may not be applicable to all sorts of companies and hence information about those conditions will not be available to the system.
- LlamaIndex lacks built-in capabilities for dynamic user interactions, such as conversational flows or multi-turn dialogues which can degrade the user experience [10].
- The learning curve for Langchain is steep. Additional integration of APIs needs extra efforts [11].

SFIT LIBRARY 2025

Chapter 3

System Analysis

3.1 Functional Requirements

- The developer has to set up all the necessary dependencies and use up-to-date software versions (e.g., Marker, LlamaIndex, LangChain, Ollama, HuggingFace, Pinecone) to ensure smooth running of the project.
- The developer has to collect and store all the financial reports used for training and querying within the system.
- The system should authenticate the user before providing them with any financial information about a company.
- The system should take user input in only PDF and plain text.
- The developer should provide an option for the user to upload the financial document of the desired company.
- The system should convert uploaded PDF documents into structured Markdown format using Marker, preserving elements like tables, equations, and code blocks.
- The system should parse the Markdown content into structured nodes using the MarkdownNodeParser from LlamaIndex, maintaining hierarchy and metadata for each section.
- The system should generate concise summaries for each node using the SummaryExtractor from LlamaIndex powered by the locally run Ollama 3.2:3b model.
- The system should embed the parsed and summarized nodes into numerical vector representations using HuggingFace's hkunlp/instructor-large model.
- The developer must ensure these embeddings are stored in the Pinecone vector database for efficient retrieval using hybrid search.

SIT LIBRARY 2025

- The system should apply query enhancement techniques to clean and improve user input queries for better search relevance.
- The system must re-rank the search results using a cross-encoder model (cross-encoder/ms-marco-MiniLM-L-6-v2) to prioritize the most contextually relevant answers.
- The system should support an interactive, context-aware chat interface using the LlamaIndex contextual chat engine to maintain the flow of conversation across multiple turns.
- The system must ensure that all critical processes (summarization, embedding, and inference) occur locally to maintain user data privacy and avoid third-party API dependencies.
- The user will be able to request additional functionalities to be integrated into the system as per their requirements.

3.2 Non-Functional Requirements

- Performance: The system should retrieve, process, and respond to financial queries within a few seconds, even when handling large PDF documents and embeddings.
- Accessibility: The application should be accessible on both Google Play Store and Apple App Store, ensuring cross-platform availability.
- Usability: The application should be user-friendly, featuring a clean and intuitive graphical user interface (GUI) with smooth navigation and onboarding for new users.
- Maintenance: The modular system architecture should allow for seamless updates and component-level maintenance without disrupting the entire application.
- Security: User data, financial documents, query logs, and embeddings must be securely stored with encryption and access control. Sensitive operations like summarization and embedding must be performed locally to ensure data privacy.
- Reliability: The system should maintain an up-time of at least 99.95%, particularly during high-demand periods such as quarterly or annual financial reporting seasons.
- Scalability: The system must scale horizontally to support multiple users concurrently and handle increasing volumes of documents and queries without performance degradation.
- Extensibility: The system should support future enhancements such as document comparison, trend detection, and visual analytics without requiring major architectural changes.
- Responsiveness: The contextual chat engine should maintain dialogue coherence and respond naturally and promptly across multi-turn conversations.

SIT LIBRARY 2025

3.3 Specific Requirements

Hardware :

- CPU : 64 Bit Intel or AMD Processor .
- GPU : Minimum 8 GB Graphics Memory with DX10.
- RAM : 8GB or above.
- Memory : Minimum 10 GB for installation and additional project files.
- Operating System : Windows 7 or above.
- User : With a Android phone having minimum 2 GB RAM and atleast 100 MB free space.

Software :

- LlamaIndex : Used for data indexing and data retrieval.
- Python : For the implementation of Advanced RAG.
- Django : For integartion with llamaIndex to support in data indexing and data retrieval.
- Nextjs : For building the frontend.
- VScode : Code editor used during the project.

3.4 Use-Case Diagrams

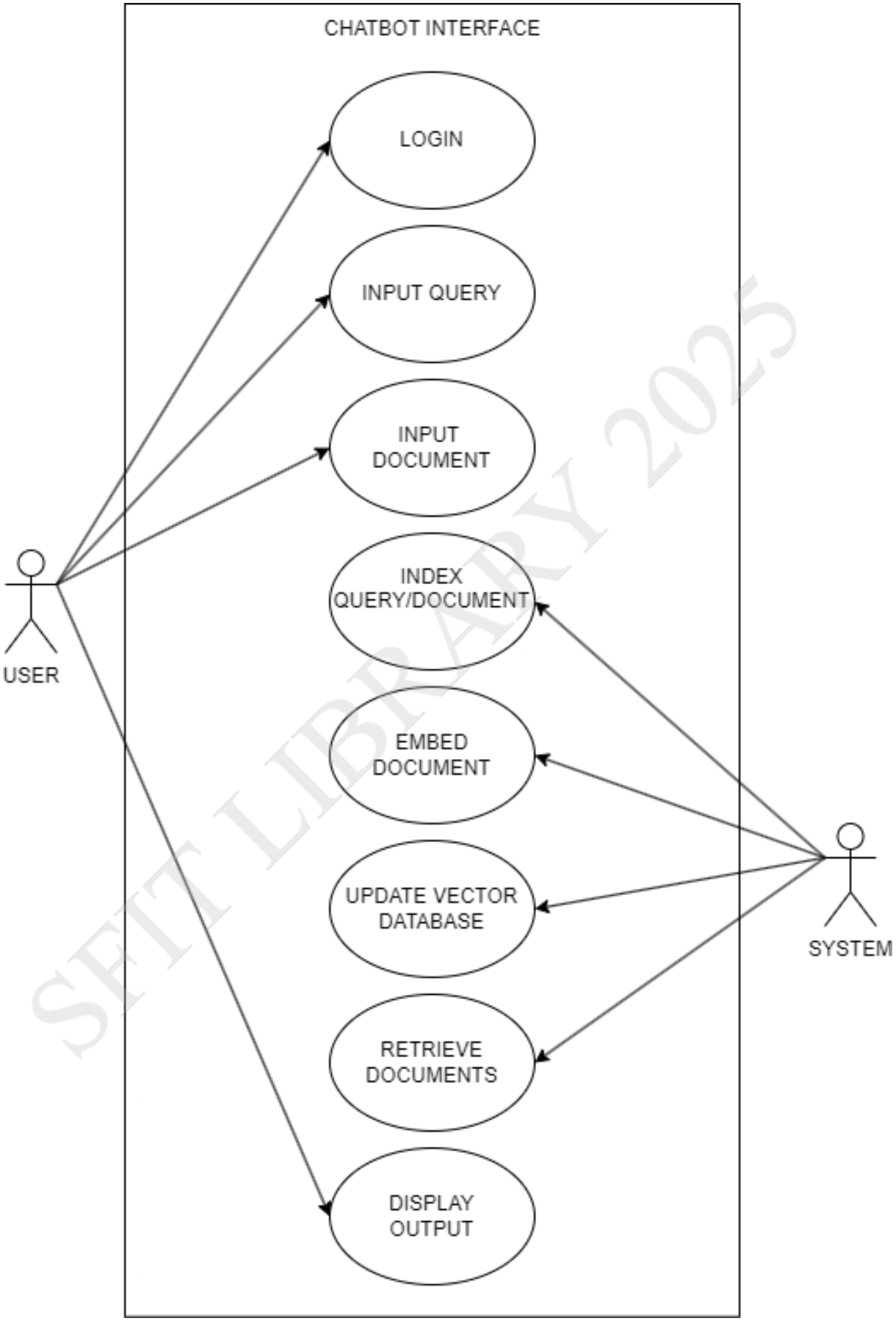
Use case diagrams are visual representations that depict the interactions between a system and its external users. They provide a high-level overview of the system's functionality and how users can interact with it to achieve their goals. The figure 4.1 depicts the use case diagram for a system using Advanced RAG for financial information.

SIT LIBRARY 2025

Table 3.1: Use Case Diagram Description

Actors	Description
User	The user initiates the chat bot interaction by providing a query or inputting a document. The system processes the user's input and processes it as indexed documents, embedded documents, or retrieved documents. The user also has the access to the generated output.
System	The system is responsible for all the processing tasks, including indexing documents, embedding documents, updating the vector database, retrieving documents, and displaying the output. It interacts with the user through the chat bot interface, receiving input and providing responses. The system employs natural language processing and machine learning techniques to understand user queries and provide accurate and relevant information.

SFIT LIBRARY 2025



SEIT LIBRARY 2025

Figure 3.1: Use Case Diagram

Use Case	**Login Functionality**
Primary Actor	User
Goal in Context	The Login is used by the user to retrieve his/her account and previous chat histories
Preconditions	Data should be fed into the system.
Trigger	-
Scenario	When the user wants to access his account
Exception	The user doesn't have an account signed up
Post condition	The user credentials are added in the database

Table 3.2: Use Case: Login Functionality

Use Case	**Input Query**
Primary Actor	User
Goal in Context	Input of the queries by the user
Preconditions	Data should be fed into the system.
Trigger	-
Scenario	When the user is logged in it can input the desired query
Exception	Does not give any input query
Post condition	The user should be indicated about the query being inputted

Table 3.3: Use Case: Input Query

SFIT LIBRARY 2025

Use Case	**Input Document**
Primary Actor	User
Goal in Context	Accept the document by the user as the dataset
Preconditions	The system should be running
Trigger	-
Scenario	The user inputs the document as the financial data which is fed into the system
Exception	Does not provide any document for the data
Post condition	The document gets feed into the system

Table 3.4: Use Case:Input Document

SJBIT LIBRARY 2025

Use Case	**Index Document/ Query**
Primary Actor	System
Goal in Context	-
Preconditions	Data should be fed into the system.
Trigger	The document/query is provided as input by the user
Scenario	When the user wants to access his account
Exception	The Document is unavailable
Post condition	The entire document is indexed

Table 3.5: Use Case: Index Document/ Query

Use Case	**Embed Document**
Primary Actor	User
Goal in Context	The data is converted into vectors and represented in high dimensional space
Preconditions	Data should be fed into the system.
Trigger	When data is inputted by the user
Scenario	When the data is inputted by the user and after the indexing of the document the data is converted into the vectors and stored.
Exception	The document is not indexed is properly
Post condition	The document should be ready to put into the update vector database

Table 3.6: Use Case: Embed Document

Use Case	**Update vector database**
Primary Actor	System
Goal in Context	The data converted into vectors needs to be stored in the vector database
Preconditions	Data should be fed into the system.
Trigger	The Vector data is available to be updated after the embedding is done
Scenario	When the user has new information that is to be added to the database for use during response generation
Exception	There is no vector data available to update in the database
Post condition	The database should be updated with the new data

Table 3.7: Use Case: Update vector database

SEIT LIBRARY 2025

Use Case	**Retrieve documents**
Primary Actor	System
Goal in Context	To make all the relevant document data available to the user to make the queries.
Preconditions	Data should be fed into the system.
Trigger	The embedding created from the input is matching with some documents in the vector database.
Scenario	When the user wants answer to a query the system searches for relevant documents in the vector database and retrieve the documents which are relevant to the context of the query
Exception	The matching document is not available
Post condition	The data that is retrieved shouldn't alter the information in the database

Table 3.8: Use Case: Retrieve documents

Use Case	**Display output**
Primary Actor	System
Goal in Context	To give a human understandable output
Preconditions	Data/ Query should be fed into the system.
Trigger	Query/ document is provided by the user
Scenario	When the user wants to answer for a finance related query and input for the same is provided the system will generate response
Exception	The system does not have satisfactory output for the provided query
Post condition	The user if satisfied should be able to end the chat

Table 3.9: Use Case: Display output

SFIT LIBRARY 2025

Chapter 4

Analysis Modeling

An activity diagram is a type of Unified Modeling Language (UML) flowchart that shows the flow from one activity to another in a system or process

4.1 Activity Diagram

The Activity Diagram outlines the steps involved in interacting with an Advanced RAG (Retrieval-Augmented Generation) system for financial queries. The process begins with the user starting the application, which leads to the home screen where the user can choose between submitting a new financial query or performing a follow-up query. For a new query, the system uses an embedding model to convert the question into a vector representation and searches the vector database for relevant financial information. The system retrieves this information, and the Large Language Model (LLM) processes the data to provide insightful answers.

If the user is satisfied with the generated insights, the process ends. However, if they need further information, they can ask a follow-up question, with the system retaining the context of the previous interaction. The updated insights are then retrieved, and the process continues until the user is satisfied or exits. The diagram shows a seamless flow between querying, retrieving data, and refining responses, focusing on user satisfaction through iterative interactions.

SHIT LIBRARY 2025

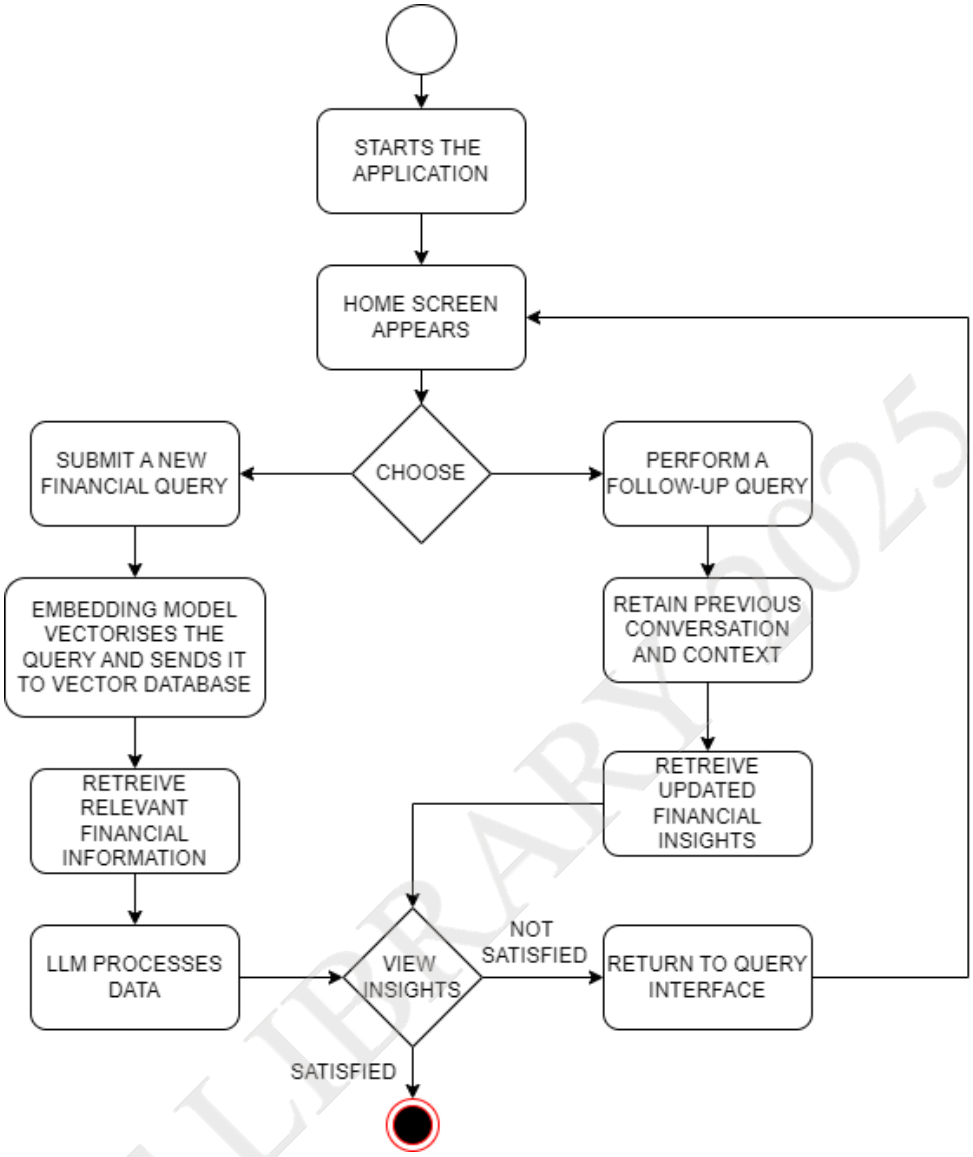


Figure 4.1: Activity Diagram for User.

4.2 Data Flow Diagram

4.2.1 DFD:Level 0

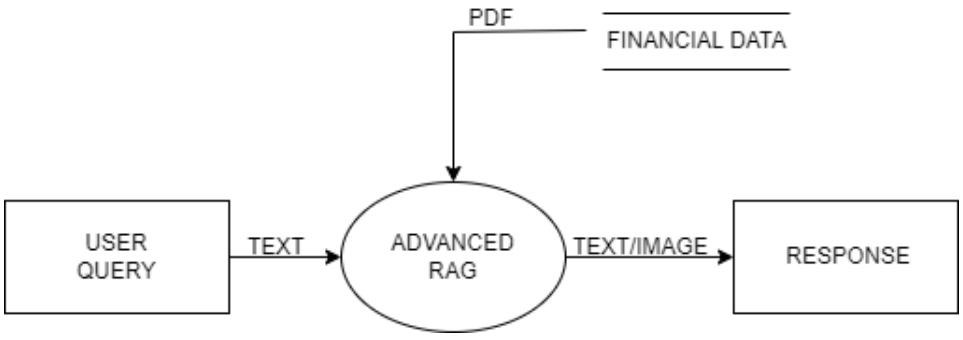


Figure 4.2: DFD Level 0.

SFIT LIBRARY 2025

The Data Flow Diagram (DFD) depicts how an Advanced RAG (Retrieval-Augmented Generation) system processes a user's query. The user submits a text query, which is fed into the Advanced RAG system. The system retrieves relevant financial data from sources like PDFs and processes the information. It then generates a response in the form of text or images, which is sent back to the user.

4.2.2 DFD:Level 1

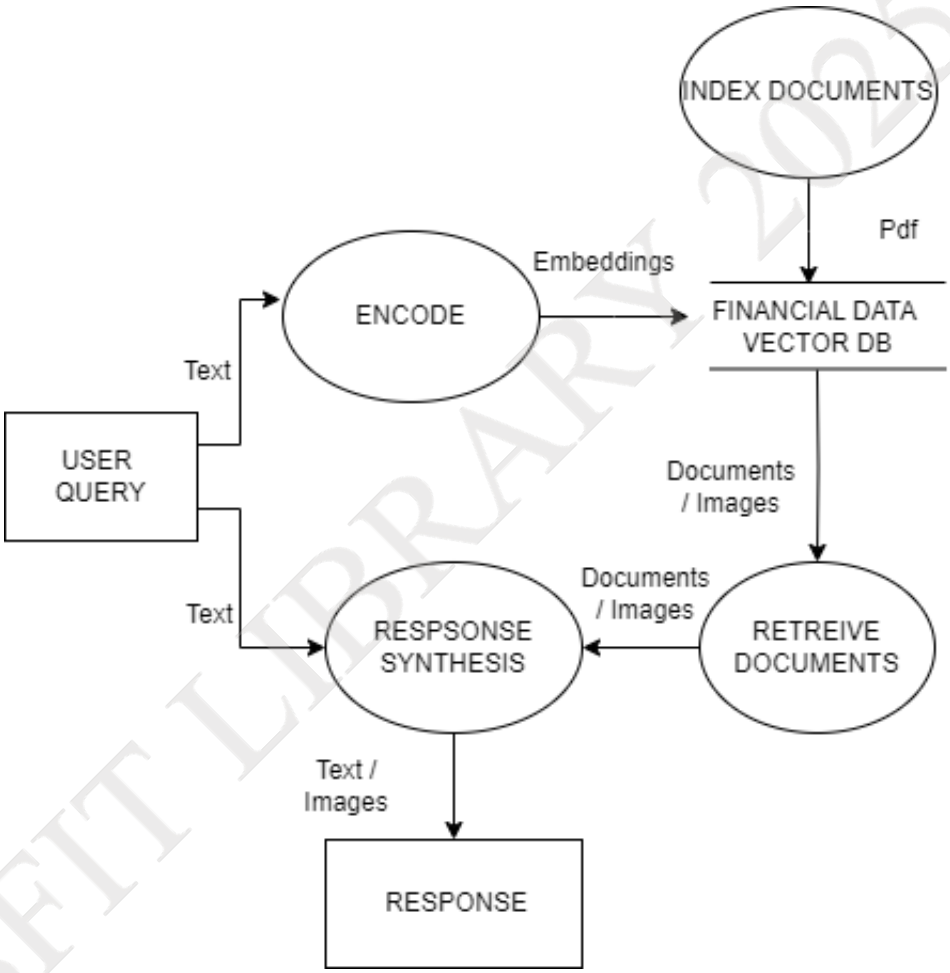
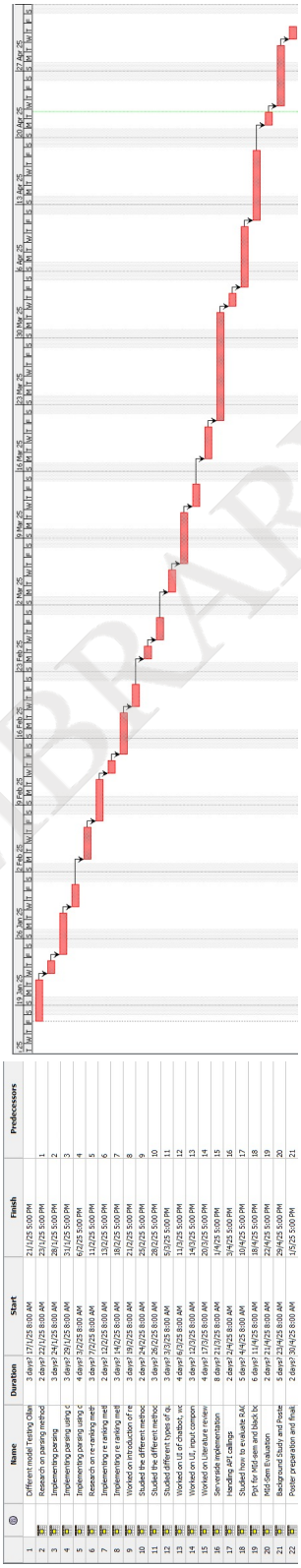


Figure 4.3: DFD Level 1.

The diagram illustrates a process for answering user queries related to financial data. It involves encoding the query, retrieving relevant documents from a vector database, and synthesizing a response.

SFIT LIBRARY 2025

4.3 Timeline Chart



		Name	Duration	Start	Finish	Predecessors
1		Different model Testing (Qlan	3 days?	17/1/25 8:00 AM	21/1/25 5:00 PM	
2		Research on parsing method	2 days?	22/1/25 8:00 AM	23/1/25 5:00 PM	1
3		Implementing parsing	3 days?	24/1/25 8:00 AM	28/1/25 5:00 PM	2
4		Implementing parsing using c	3 days?	29/1/25 8:00 AM	31/1/25 5:00 PM	3
5		Implementing parsing using c	4 days?	3/2/25 8:00 AM	6/2/25 5:00 PM	4
6		Research on re-ranking meth	3 days?	7/2/25 8:00 AM	11/2/25 5:00 PM	5
7		Implementing re ranking meth	2 days?	12/2/25 8:00 AM	13/2/25 5:00 PM	6
8		Implementing re ranking meth	3 days?	14/2/25 8:00 AM	18/2/25 5:00 PM	7
9		Worked on introduction of re	3 days?	19/2/25 8:00 AM	21/2/25 5:00 PM	8
10		Studied the different methoc	3 days?	24/2/25 8:00 AM	26/2/25 5:00 PM	9
11		Studied the different methoc	3 days?	26/2/25 8:00 AM	28/2/25 5:00 PM	10
12		Studied different types of q	3 days?	3/3/25 8:00 AM	5/3/25 5:00 PM	11
13		Worked on UI of chatbot, inc	4 days?	6/3/25 8:00 AM	11/3/25 5:00 PM	12
14		Worked on UI, input compon	3 days?	12/3/25 8:00 AM	14/3/25 5:00 PM	13
15		Worked on Literature review	4 days?	17/3/25 8:00 AM	20/3/25 5:00 PM	14
16		Serverside implementation	8 days?	21/3/25 8:00 AM	1/4/25 5:00 PM	15
17		Handling API callings	2 days?	2/4/25 8:00 AM	3/4/25 5:00 PM	16
18		Studied how to evaluate RAC	5 days?	4/4/25 8:00 AM	10/4/25 5:00 PM	17
19		Ppt for mid-sem and black bc	6 days?	11/4/25 8:00 AM	18/4/25 5:00 PM	18
20		Mid-Sem Evaluation	2 days?	21/4/25 8:00 AM	22/4/25 5:00 PM	19
21		Revise content of ppt and Poste	2 days?	23/4/25 8:00 AM	24/4/25 5:00 PM	20
22		Poster preparation and final	2 days?	30/4/25 8:00 AM	1/5/25 5:00 PM	21

Figure 4.4: Timeline details and chart

SFIT LIBRARY 2025

Chapter 5

System Architecture

5.1 System Architecture

This chapter illustrates the end-to-end architecture of the proposed system designed to enable intelligent retrieval and analysis of financial documents using a Retrieval-Augmented Generation (RAG) framework. The system is specifically optimized to process annual reports, extract meaningful data, and provide accurate, context-aware answers to user queries.

The architecture is broadly divided into four stages: (1) Document Preprocessing, (2) Vector Database Construction, (3) Query Handling and Retrieval, and (4) Response Generation. Figure 5.1 presents a high-level flow of the complete pipeline.

Document Preprocessing

- The system begins by parsing the uploaded financial PDF using a specialized PDF parser (Marker), which extracts text, tables, and footnotes. The extracted content is then converted into Markdown format, which standardizes structure and improves downstream parsing.
- The Markdown is split into semantically meaningful chunks using a Markdown Text Splitter. Each chunk undergoes further parsing to identify logical sections such as risk factors, business overview, and financial highlights.

2. Embedding and Storage

- Each chunk is embedded into a high-dimensional vector space using a powerful embedding model (e.g., `hkunlp/instructor-large`). Summary representations are also generated via a summary extractor using Gemini LLM to condense and enhance chunk quality.
- These embeddings are stored in a vector database (Pinecone), enabling efficient similarity-based retrieval during the question answering process.

3. Query Processing and Retrieval

SIIT LIBRARY 2025

- User queries are first refined using Gemini LLM's query refinement mechanism, which restructures the question for better retrieval performance. The refined query is embedded using the same embedding model to maintain vector space consistency.
- The system then performs a similarity search against the stored document chunks using cosine similarity and retrieves the top-k relevant chunks. These retrieved results are further re-ranked using a cross-encoder model (`cross-encoder/ms-marco-MiniLM-L-6-v2`) to ensure that the most relevant chunks are prioritized.

4. Prompt Augmentation and Response Generation

- The re-ranked context is merged with the user query to form an augmented prompt. This enriched prompt is passed to Gemini LLM, which generates a grounded and context-specific response. This approach ensures responses are factually supported by source documents, minimizing hallucinations.

The described architecture effectively combines LLM generative power with retrieval-based grounding. It ensures that user queries about financial documents are answered with high factual accuracy, providing a reliable and efficient solution for investors, analysts, and regulatory bodies.

SFIT LIBRARY 2025

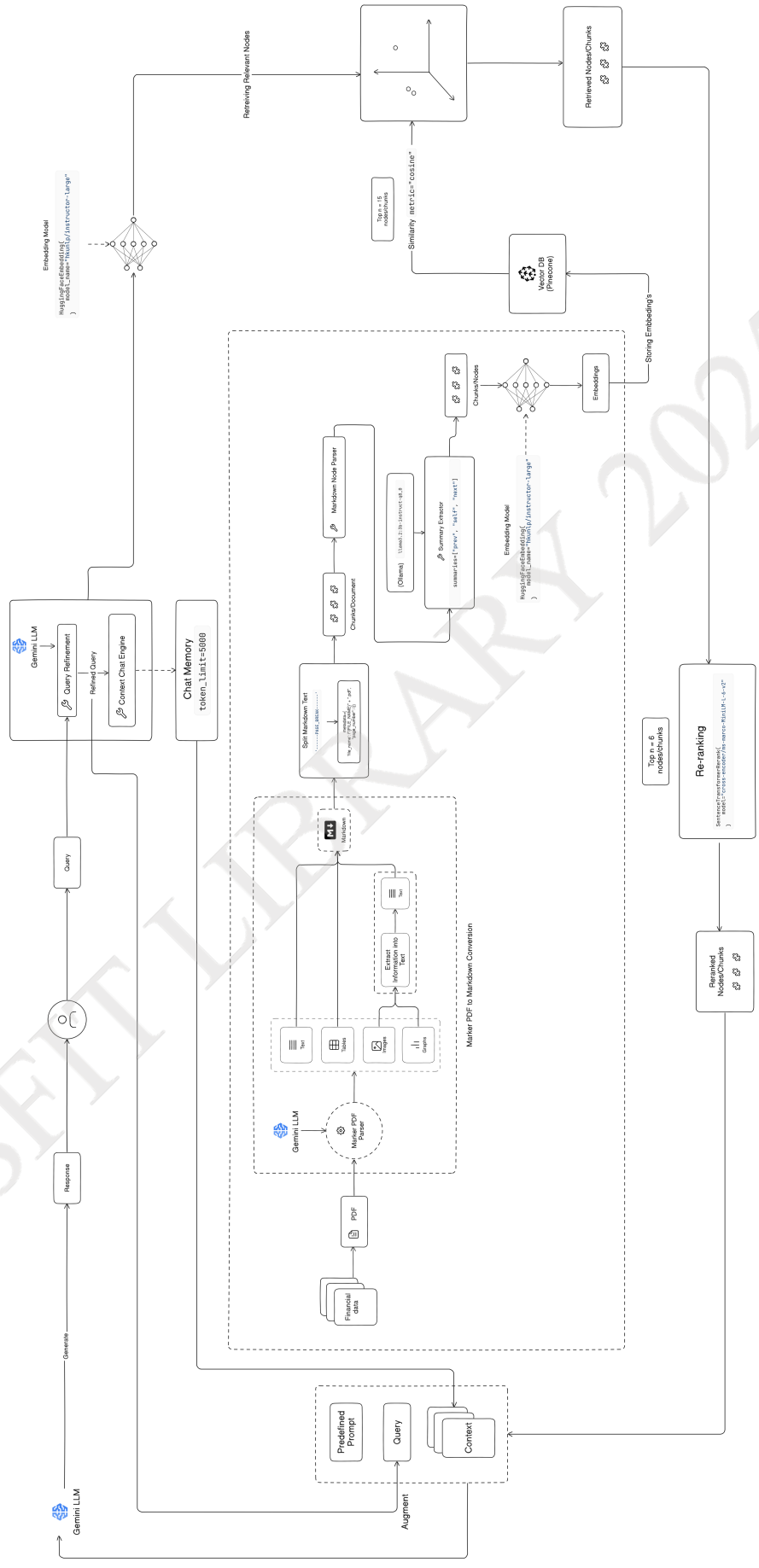


Figure 5.1: Advanced RAG system

SETT LIBRARY 2025

5.2 Graphical User Interface

This section presents the GUI screenshots of the Advanced RAG system, showcasing key features such as table data extraction, document summarization, factual question answering, and general conversational interaction. The screenshots highlight the system's ability to handle a wide range of financial queries through a clean, user-friendly interface, demonstrating both functionality and ease of use.

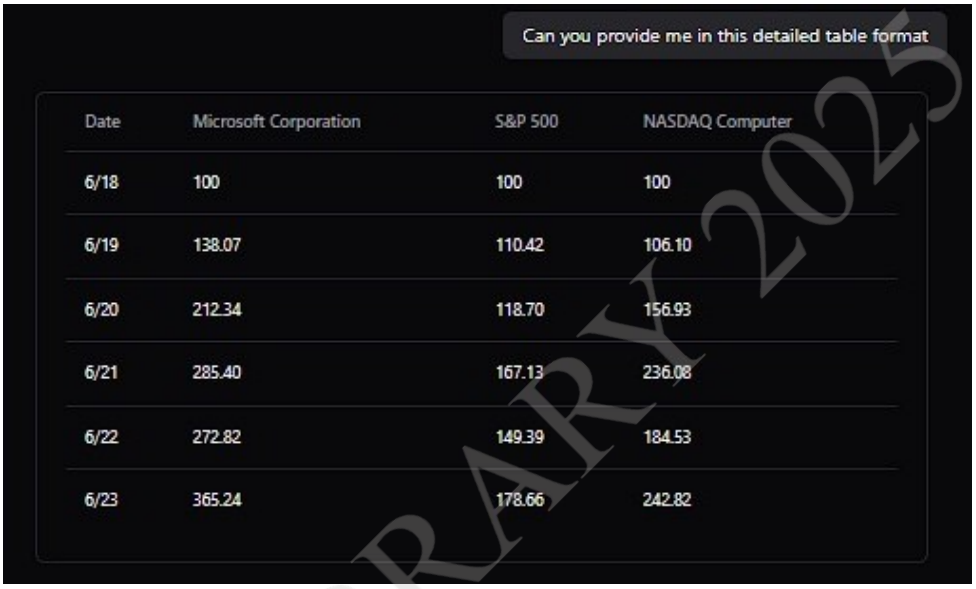


Figure 5.2: Data extraction in Table format

This figure demonstrates the system's ability to accurately extract and interpret numerical and structured data from tables within financial documents. The extracted values are presented clearly without distortion or loss of context. This is particularly useful for retrieving metrics such as revenue, expenditure, or capital expenditure directly from complex reports.

SFIT LIBRARY 2025

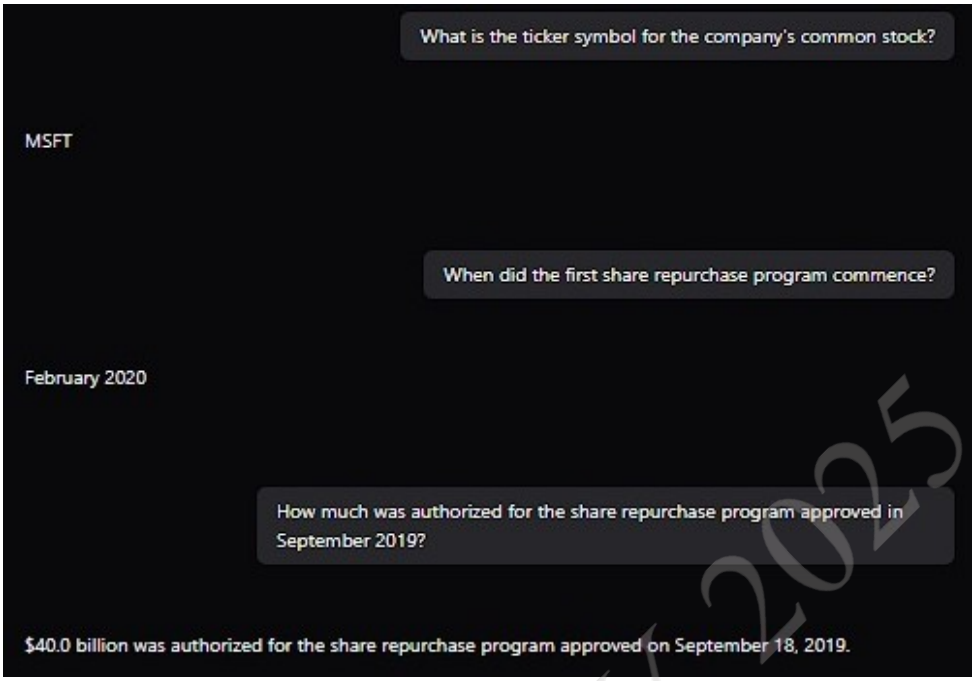


Figure 5.3: Factual questions answered by system

Here, the system responds to a fact-based query by retrieving precise information directly from the document. The answer is accompanied by the relevant context or reference, ensuring transparency. This feature supports quick decision-making by allowing users to ask targeted questions and receive accurate, document-backed answers.

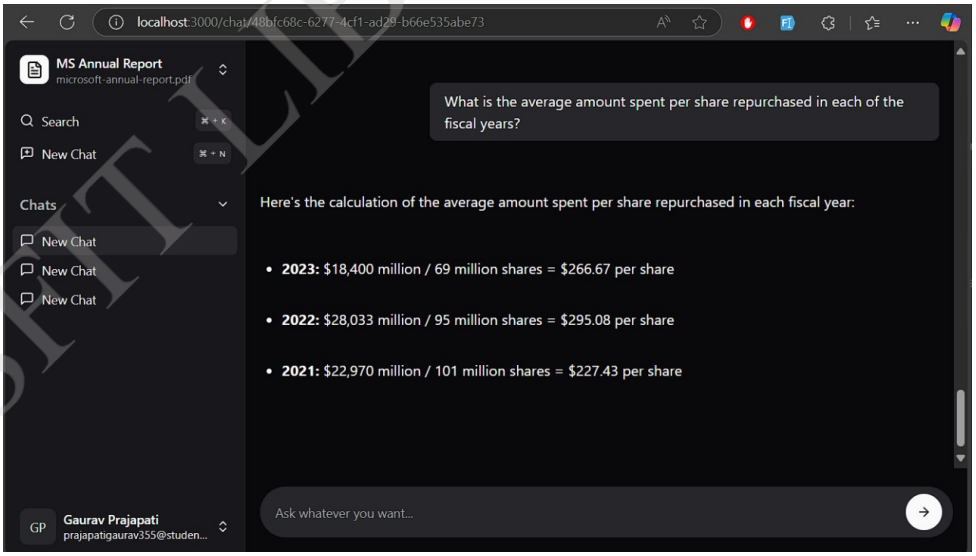


Figure 5.4: Mathematical question answering

The system supports basic mathematical operations, allowing users to ask questions involving addition, subtraction, percentage calculations, and comparisons. It accurately interprets and computes values based on numerical data from the document.

SPIT LIBRARY 2025

Chapter 6

Implementation

6.1 Implementation

The implementation of the proposed system involves a combination of document preprocessing, semantic chunking, vector storage, enhanced querying, reranking, and interactive chat functionalities. Each step is designed to solve a specific challenge in financial document analysis and retrieval.

6.1.1 User Authentication with Supabase

The system begins with a secure user login mechanism, implemented using **Supabase**. Supabase provides a backend-as-a-service platform with built-in authentication and database services. It is used for:

- **Email and Password Authentication:** Users are authenticated via a secure login interface connected to Supabase's authentication module.
- **User Data Management:** Basic user details (such as name, email, login timestamps) are stored in Supabase's PostgreSQL database.
- **Chat History Storage:** All queries posed by the user and responses generated by the system are stored in Supabase, allowing users to revisit previous conversations and enabling system improvement via feedback analysis.

6.1.2 PDF to Markdown Conversion using Marker

Financial documents, primarily in PDF format, are first processed using the **Marker** tool. Marker is an advanced document parser that converts complex PDFs (with tables, headers, and multi-column formats) into Markdown, preserving semantic structure. This Markdown format is more suitable for chunking and parsing in subsequent stages.

SIT LIBRARY 2025

6.1.3 Markdown Parsing using LlamaIndex Node Parser

The Markdown files are then parsed using the **MarkdownNodeParser** from **LlamaIndex**. This parser breaks the document into semantically meaningful nodes based on headers and sections. Each node is enriched with metadata, such as section titles, page numbers, and hierarchy, which facilitates structured indexing and retrieval.

6.1.4 Summary Extraction with SummaryExtractor

Each parsed node is summarized using the **SummaryExtractor** from LlamaIndex. This process generates short, meaningful summaries that are stored as metadata and later used to assist in ranking and filtering relevant nodes during query time. The summary generation is performed locally using **Ollama 3.2:3b**, ensuring privacy and efficiency.

6.1.5 Vector Embedding with HuggingFace Models

To perform semantic search, each node (along with its summary) is converted into an embedding vector using **HuggingFace's hkunlp/instructor-large** model. These vectors capture the semantic meaning of the text and enable similarity-based retrieval.

6.1.6 Vector Storage using Pinecone

The resulting embedding vectors are stored in **Pinecone**, a vector database designed for real-time, scalable similarity search. Pinecone allows for fast retrieval of the most relevant document chunks during query processing based on cosine similarity with the user query embedding.

6.1.7 Query Enhancement

Before retrieval, user queries are processed through a **Query Enhancement Module**, which corrects grammatical errors, expands abbreviations, and rephrases incomplete questions. This ensures better semantic alignment between the query and stored document nodes.

6.1.8 Reranking with Cross-Encoder

The top retrieved document nodes are reranked using the **cross-encoder/ms-marco-MiniLM-L-6-v2** model. Cross-encoders jointly encode the query and document pair to evaluate contextual relevance, allowing more accurate prioritization of retrieved chunks.

6.1.9 Contextual Chat Engine with LlamaIndex

The reranked document chunks are fed into the **Contextual Chat Engine** of LlamaIndex. This engine supports conversational memory, allowing the system to maintain context across

SIT LIBRARY 2025

multi-turn interactions. It enhances user experience by delivering coherent and relevant answers in the form of a chatbot.

6.2 Application Performance

The performance of the Advanced Retrieval-Augmented Generation (RAG) system for financial information was assessed based on its ability to retrieve accurate insights, respond efficiently to queries, and provide a smooth user experience when interacting with complex financial documents. The system has been designed with a focus on reliability, precision, and ease of use in financial analysis and decision-making contexts.

- Query Responsiveness: The system delivers responses typically within 3–6 seconds, depending on the complexity of the question and the size of the financial document. This is achieved through the use of precomputed vector embeddings and efficient similarity searches via a vector database (Pinecone), enabling quick retrieval of relevant content for the LLM to process.
- Handling of Financial Data: The system accurately extracts and processes both textual and numerical data from financial reports, including complex tables. It supports basic financial calculations such as differences, ratios, and percentage change, making it useful for interactive analysis directly from the source documents.
- Accuracy and Transparency: The system consistently provides context-aware and relevant answers, maintaining alignment with the source content. It also includes reference links or excerpts with each response, ensuring transparency and enabling users to verify the answer's origin—a critical feature for financial professionals.
- Range of Query Types: Extensive testing showed the system handles a wide variety of questions, including table data extraction, summary generation, factual Q&A, and general explanatory queries. This demonstrates the versatility of the system in understanding and adapting to different user intents within the financial domain.
- User Interface and Usability: The interface allows users to easily upload PDFs, enter queries, and view structured responses in a clean and conversational format. Recent enhancements like tooltips and response feedback have improved clarity and guided users through error handling or insufficient results, enhancing the overall usability.

SET LIBRARY 2025

6.3 Code

```
import os
import time
import multiprocessing
import nest_asyncio
from llama_index.core import Document, Settings
from llama_index.core.node_parser import MarkdownNodeParser
from llama_index.core.extractors import SummaryExtractor,
    QuestionsAnsweredExtractor
from llama_index.core.schema import MetadataMode
from llama_index.core.ingestion import IngestionPipeline
from llama_index.vector_stores.pinecone import PineconeVectorStore
from pinecone import Pinecone, ServerlessSpec

from src.services.pdf_service import PDFToMarkDownTextProcessor

# Dictionary to track processing tasks
processing_tasks = {}

class DocumentProcessor:
    """Service for processing documents and creating vector store
       indices"""

    def __init__(self, llm, embed_model, pinecone_client):
        nest_asyncio.apply()
        self.llm = llm
        self.embed_model = embed_model
        self.pc = pinecone_client

    def process_document(self, file_path, doc_id=None):
        """Process a document and store it in Pinecone"""
        # Check for cancellation before starting
        if doc_id and processing_tasks.get(doc_id, {}).get('cancelled
            ', False):
            raise Exception("Processing cancelled by user")

        # Converting To MD Text using a separate process that can be
            terminated
        print(f"Starting PDF to Markdown conversion for {file_path}")

        # Use Manager for more reliable cross-process communication
        manager = multiprocessing.Manager()
        result_queue = manager.Queue()

        # Create and start the process
```

SFIT LIBRARY 2025

```
            md_processor = PDFToMarkDownTextProcessor()
            process = multiprocessing.Process(
                target=md_processor.process_with_timeout,
                args=(file_path, result_queue)
            )

            # Store the process in processing_tasks to be able to
               terminate it
            if doc_id:
                processing_tasks[doc_id]['process'] = process

            process.start()
            print(f"Started PDF conversion process with PID: {process.pid
               }")

            # Wait for the process to complete or be cancelled
            process_timeout = 600  # 10 minutes max
            process_wait_interval = 1  # Check every second

            for i in range(process_timeout):
                # Check if process is done
                if not process.is_alive():
                    print(f"PDF conversion process completed after {i}
                       seconds")
                    break

                # Check for cancellation
                if doc_id and processing_tasks.get(doc_id, {}).get('
                   cancelled', False):
                    print(f"Terminating PDF conversion process for {
                       doc_id}")
                    process.terminate()
                    process.join(5)  # Give it 5 seconds to terminate
                    if process.is_alive():
                        process.kill()  # Force kill if still alive
                    raise Exception("Processing cancelled by user")

                # Wait a bit before checking again
                time.sleep(process_wait_interval)

            # If we got here and process is still running, it timed out
            if process.is_alive():
                print(f"PDF conversion timed out after {process_timeout}
                   seconds")
                process.terminate()
                process.join(5)
                if process.is_alive():
```

SFIT LIBRARY 2025

```
                process.kill()
            raise Exception("PDF conversion timed out")

        # Get the result from the queue with timeout
        try:
            print("Waiting for result from PDF conversion...")
            # Wait max 10 seconds for result
            md_text = result_queue.get(block=True, timeout=10)
            print("Got result from PDF conversion queue")

            # Check if there was an error
            if isinstance(md_text, str) and md_text.startswith("ERROR
                :"):
                print(f"PDF conversion reported error: {md_text}")
                raise Exception(md_text)
        except Exception as e:
            if "queue empty" in str(e).lower():
                print("Queue was empty, PDF conversion failed to
                    produce result")
                raise Exception("PDF conversion failed to produce
                    output")
            else:
                print(f"Error getting result from queue: {str(e)}")
                raise

        # Check for cancellation after markdown conversion
        if doc_id and processing_tasks.get(doc_id, {}).get('cancelled
            ', False):
            raise Exception("Processing cancelled by user")

        print(f"Creating document nodes")
        # Create document nodes
        # Extract just the filename without the path
        filename = os.path.basename(file_path)
        documents = [Document(text=part.strip(), metadata={
            'file_name': filename,
            'page_number': i
        }) for i, part in enumerate(md_text.split('------PAGE_BREAK
            ------')) if part.strip()]

        # Check for cancellation after creating document nodes
        if doc_id and processing_tasks.get(doc_id, {}).get('cancelled
            ', False):
            raise Exception("Processing cancelled by user")

        # Parse and extract metadata
        print(f"Parsing and extracting metadata")
```

SET LIBRARY 2025

```
        node_parser = MarkdownNodeParser(show_progress=True)
        extractors = [SummaryExtractor(summaries=["prev", "self", "
            next"], llm=self.llm),
                      QuestionsAnsweredExtractor(questions=3, llm=
                          self.llm, metadata_mode=MetadataMode.EMBED)]

        pipeline = IngestionPipeline(
            transformations=[node_parser, *extractors])
        nodes = pipeline.run(
            nodes=documents, in_place=False, show_progress=True)

        # Check for cancellation after pipeline run
        if doc_id and processing_tasks.get(doc_id, {}).get('cancelled
            ', False):
            raise Exception("Processing cancelled by user")

        # Generate embeddings
        print(f"Generating embeddings")
        for i, node in enumerate(nodes):
            # Check for cancellation periodically during embedding
                generation
            if doc_id and i % 5 == 0 and processing_tasks.get(doc_id,
                {}).get('cancelled', False):
                raise Exception("Processing cancelled by user")

            node.embedding = self.embed_model.get_text_embedding(
                node.get_content(metadata_mode=MetadataMode.EMBED)
            )

        # Check for cancellation before Pinecone storage
        if doc_id and processing_tasks.get(doc_id, {}).get('cancelled
            ', False):
            raise Exception("Processing cancelled by user")

        # Store in Pinecone
        print(f"Storing in Pinecone")
        index_name = os.path.splitext(os.path.basename(file_path))[0]
        if index_name not in self.pc.list_indexes().names():
            self.pc.create_index(
                str(index_name).lower(),
                dimension=768,
                metric="euclidean",
                spec=ServerlessSpec(cloud="aws", region="us-east-1")
            )
        pinecone_index = self.pc.Index(index_name)
        vector_store = PineconeVectorStore(pinecone_index=
            pinecone_index)
```

SFIT LIBRARY 2025

```
            vector_store.add(nodes)

            # Final cancellation check
            if doc_id and processing_tasks.get(doc_id, {}).get('cancelled
                ', False):
                raise Exception("Processing cancelled by user")

            print(f"Document processing complete")

    # Export the processing_tasks dictionary

    def get_processing_tasks():
        return processing_tasks
```

SFIT LIBRARY 2025

Chapter 7

Testing

7.1 Test Cases

Table 7.1: Test Cases

Test Case ID	Test Scenario	Test Steps	Status
TC01	PDF Upload Functionality	Upload a valid PDF file .	PASS
TC02	Large PDF Upload	Upload file upto 30 MB.	PASS
TC03	Multiple File Upload	Upload four valid PDFs simultaneously.	PASS
TC04	Invalid File Type	Try uploading a non-PDF file (e.g., .txt or .jpg).	PASS
TC05	Ask a Simple Question	Ask a direct factual question from the uploaded PDF.	PASS
TC06	RAG Output Integration	Query based on content from uploaded PDFs using local retrieval.	PASS
TC07	Query with Invalid/Out-of-Document Dates	Ask a question containing a date not present in the document (e.g., ”What was the revenue in 2030?”).	PASS

SFIT LIBRARY 2025

7.2 Testing Methods Used

7.2.1 Unit Testing

Unit testing is focused on validating the individual components of the system in isolation, such as the PDF parser, Markdown node generator, embedding model, and query enhancer. These components form the backbone of the RAG architecture and must work reliably to ensure accurate and efficient financial data retrieval. Unit tests were conducted to evaluate individual components in isolation, such as:

- PDF-to-Markdown conversion
- Node parsing and summarization
- Embedding generation
- Query enhancement
- Vector retrieval from Pinecone Each unit test passed successfully, with zero failures and errors, as reflected in the test summary logs. The total runtime remained efficient, and tests covered both expected and edge scenarios to ensure robustness at the function level.

7.2.2 Integration Testing

Integration testing validated the interactions between subsystems—ensuring smooth data flow from PDF ingestion to user response delivery. This included end-to-end testing from document input, chunk creation, metadata handling, embedding, storage in the vector database, and final output generation via the LLM. The test results show successful integration with all components working in unison. No functional mismatches or system failures were observed. The test results log confirmed a 100% pass rate for integration tests.

7.2.3 Usability Testing

Usability testing was conducted to assess the ease of use, intuitiveness, and overall user experience of the Advanced RAG system for financial information retrieval. The objective was to ensure that users—regardless of their technical background—could comfortably interact with the system, upload documents, and retrieve accurate responses without confusion.

Test participants were asked to perform typical user actions, such as uploading a financial PDF, submitting queries related to its content, and reviewing the system's answers. The testing evaluated navigation flow, system responsiveness, clarity of interface elements, and overall satisfaction. The results indicated that users were able to complete all tasks successfully and found the interface clean and straightforward.

SFIT LIBRARY 2025

Based on earlier feedback, improvements such as tooltips for buttons and detailed system feedback when queries yield insufficient results were implemented prior to testing. These enhancements significantly improved clarity and user confidence during interaction. Testers appreciated the conversational flow of the chatbot and the relevance of the responses generated. Overall, the system was found to be highly usable, with an effective balance of functionality and simplicity, confirming that it meets usability standards and provides a smooth experience for end users.

7.2.4 Smoke and Sanity Testing

Smoke and sanity testing were critical in validating the stability and core functionality of the Advanced Retrieval-Augmented Generation (RAG) system for financial information retrieval. Smoke testing was carried out immediately after the initial deployment to ensure that the essential workflows of the system—such as text query submission, financial PDF upload, document parsing, embedding generation, and chatbot interaction—were functioning as expected. During this phase, a financial report was successfully uploaded through the interface and processed without any issues. A confirmation message verified that the document had been parsed and embedded correctly, and the system returned accurate, context-aware responses from the uploaded file. This validated that the core pipeline—upload, process, retrieve, and respond—was stable and operational.

Following smoke testing, sanity testing was performed to ensure that recent updates or bug fixes did not introduce new issues or break existing functionality. Specific modules, including document chunking, metadata summarization, vector generation, vector store retrieval, and LLM-based response generation, were tested independently. Each component maintained its functionality and integrated seamlessly within the overall system. The application responded reliably to both standard and edge-case inputs, confirming its consistency, robustness, and readiness for further testing stages. Together, these tests confirmed that the system was structurally sound and ready for real-world use.

SFIT LIBRARY 2025

```
----------------------------------------------------------------------
Ran 7 tests in 7.575s

OK

======================================================================
Test Summary: PASSED
Total tests: 7
Failures: 0
Errors: 0
Skipped: 0
Time: 12.73 seconds
Results saved to test_results\test_results_20250423_010058.json
```

Figure 7.1: Test case structure

SMIT LIBRARY 2025

```
======================================================================
OVERALL TEST RESULTS
======================================================================
Unit Tests: PASSED
Integration Tests: PASSED
Smoke Tests: PASSED
Sanity Tests: PASSED
Usability Tests: PASSED

Overall Result: PASSED
(myenv) PS G:\Coding\AIML\Major Project\Advanced-RAG-Finance\server\tests>
```

Figure 7.2: Test case results

Chapter 8

Result and Discussion

8.1 Results and Discussion

The performance of the proposed Advanced Retrieval-Augmented Generation (RAG) system was evaluated using standard metrics provided by the DeepEval framework. These metrics assess different dimensions of the system's effectiveness, including the relevance and faithfulness of responses, contextual understanding, summarization capability, and safety in terms of bias and toxicity.

8.1.1 Evaluation Results

- **Answer Relevancy:** 0.97
- **Faithfulness:** 1.0
- **Contextual Precision:** 0.64
- **Contextual Recall:** 0.75
- **Summarization Quality:** 0.075
- **Bias Score:** 0.0
- **Toxicity Score:** 0.0

These results indicate a highly reliable and accurate system for financial question answering. The high values for Answer Relevancy and Faithfulness ensure that the system generates responses that are both relevant to the query and grounded in the source document. Contextual Precision and Recall show that the system is capable of retrieving semantically appropriate chunks, although there is room for improvement, particularly in summarization, as shown by its relatively low score.

The bias and toxicity scores being 0.0 validate the safety and neutrality of the system, which is especially crucial in the finance domain where objective analysis is essential.

SFIT LIBRARY 2025

8.1.2 Metric Definitions and Formulas

DeepEval, open source evaluation framework for LLMs with 14+ evaluation metrics, it enables creation and modifying of metrics according to project needs.It is used in the project for evaluation. The metric used for the project are as follows:

Answer Relevancy Answer relevancy metric is used to measure the quality of the RAG pipeline's generator by evaluating how relevant the actual_output of the LLM application is compared to the provided input. [12]

$$AnswerRelevancy = \frac{NumberofRelevantStatements}{TotalNumberofStatements} \tag{8.1}$$

Contextual Recall The extracted statements made in the expected_output is used to classify whether each statement can be attributed to nodes in the retrieval_context using the Contextual Recall metric. [13]

$$ContextualRecall = \frac{NumberofAttributableStatements}{TotalNumberofStatements} \tag{8.2}$$

Contextual Precision The RAG pipeline's retriever is evaluated by the nodes in the retrieval_context that are relevant to the given input and are ranked higher than irrelevant ones using the contextual precision metric which uses LLM-as-a-judge. [14]

$$ContextualPrecision = \frac{1}{NumberofRelevantNodes} \sum_{k=1}^{n} \left(\frac{NumberofRelevantNodesUptoPositionk}{k} \times r_k \right) \tag{8.3}$$

Summarization The summarization metric uses LLM-as-a-judge to determine whether the LLM is generating factually correct summaries based on the necessary details from the original text.[15]

$$Summarization = \min(AlignmentScore,\ CoverageScore) \tag{8.4}$$

SFIT LIBRARY 2025

Faithfulness The quality of the RAG model by evaluating whether the actual_output factually aligns with the contents of our retrieval_context is measured by the faithfulness metric using LLM-as-a-judge.[16]

$$Faithfulness = \frac{NumberofTruthfulClaims}{TotalNumberofClaims}$$

(8.5)

Toxicity The toxicity metric is another referenceless metric that uses LLM-as-a-judge to evaluate toxicness in the LLM outputs. [17]

$$Toxicity = \frac{NumberofToxicOpinions}{TotalNumberofOpinions}$$

(8.6)

Bais The bias metric uses LLM to determine whether the LLM output contains gender, racial, or political bias. [18]

$$Bias = \frac{NumberofBiasedOpinions}{TotalNumberofOpinions}$$

(8.7)

8.1.3 Discussion

The system effectively handles both textual and numerical data from the uploaded financial documents, retrieving numerical values accurately without any modification. Although it does not yet support advanced financial analysis, it can perform basic mathematical operations when required. Data extraction from tables is performed efficiently, ensuring that structured information is processed and presented correctly. Furthermore, the system provides clear references for each generated response, ensuring transparency and traceability—an aspect often missing in similar solutions.

SIT LIBRARY 2025

Chapter 9

Result and Discussion

9.1 Conclusion

The proposed Advanced Retrieval-Augmented Generation (RAG) system for financial information extraction and question answering successfully addresses the limitations of traditional LLM-based approaches. By incorporating structured document conversion, semantic chunking, query enhancement, embedding with HuggingFace models, and reranking using cross-encoders, the system significantly improves the relevance and accuracy of retrieved responses. The integration of LlamaIndex and Pinecone further enhances contextual awareness and information retrieval capabilities. Additionally, by minimizing hallucinations, data loss, and misinformation, the system provides users—particularly analysts and investors—with a reliable tool to interpret complex financial documents and make informed decisions.

9.2 Future Scope

While the system currently supports interaction with a single financial document per session, future enhancements could include:

- **Multi-document Handling:** Enabling simultaneous querying and context switching across multiple financial documents within a single chat session.
- **Vision-Language Model Integration:** Extending the system to extract insights from charts, graphs, and images embedded in financial documents.
- **Domain Adaptation:** Training domain-specific LLMs or fine-tuning models to improve performance on highly technical financial texts.
- **Temporal Analysis:** Incorporating time-series data processing to allow trend analysis and forecasting.

SFIT LIBRARY 2025

- **User Personalization:** Adapting responses based on user profiles or preferences for a more customized experience.
- **Multilingual Support:** Expanding capabilities to include regional languages for broader accessibility.

SFIT LIBRARY 2025

Chapter 10

Acknowledgement

We would like to extend our heartfelt gratitude to St. Francis Institute of Technology for providing us with the opportunity and resources to undertake and successfully complete our project titled “Advanced RAG for financial Information.” Our sincere thanks to our Director, Bro. Shantilal Kujur, Principal, Dr. Sincy George, and Head of Department, Dr. Prachi Raut, for their unwavering support, guidance, and for fostering an environment that encourages innovation and academic excellence.

We are deeply appreciative of our project guide, Ms. Alvina Lopes, for her expert guidance, constructive feedback, and continuous support throughout the course of this project. Her mentorship has been instrumental in shaping the direction and success of our work. We would also like to express our gratitude to Mr. Devesh Rajadhyax, Co-founder and CEO of Cere Labs, for providing us with the industry-sponsored project, which allowed us to apply our knowledge to real-world challenges.

SFIT LIBRARY 2025

Literature Cited

[1] Spurthi Setty, Harsh Thakkar, Alyssa Lee, Eden Chung, Natan Vidra, "Improving Retrieval for RAG based Question Answering Models on Financial Documents", arXiv preprint arXiv:2404.07221, Aug 2024.

[2] Dr. Selva Kumar S,Imadh Ajaz Banday, Vibha Venkatesh Shanbhag, Afifah Khan Mohammed Ajmal Khan, Manikantha Gada, "Overcoming LLM Challenges using RAG-Driven Precision in Coffee Leaf Disease Remediation", *2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS), Bengaluru, India, 2024*, pp. 1-6, doi: 10.1109/ICETCS61022.2024.10543859.

[3] Min Gao, Jize Xiong, Yanqi Zong, Shulin Li, "Science Exam Question Answering based on Retrieval-Augmented Generation", *2024 3rd International Conference on Computer Technologies (ICCTech)*, 979-8-3503-9541-9/24/31.002024$IEEE|DOI$: 10.1109/$ICCTech$61708.2024.00020.

[4] Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li, "Financial Report Chunking for Effective Retrieval Augmented Generation", arvix preprint arXiv:2402.05131v2, Feb 2024.

[5] Junde Wu, Jiayuan Zhu, Yunli Qi, "Medical Graph Rag: Towards Safe Medical Large Language Model Via Graph Retrieval Augmented Generation", *arXiv:2408.04187v1 [cs.CV] 8 Aug 2024*.

[6] MyScale, "Enhancing Advanced RAG Systems Using Reranking with LangChain", June 2024. [Online] Available: https://medium.com/@myscale/enhancing-advanced-rag-systems-using-reranking-with-langchain-523a0b840311 [Last accessed: Nov 10, 2024]

[7] Tredence Studio, "Multi-Modal Retrieval-Augmented Generation (RAG) Systems: An In-Depth Overview and Implementation", July 2024. [Online] Available: https://medium.com/@tredencestudio/multi-modal-retrieval-augmented-generation-rag-systems-an-in-depth-overview-and-implementation-f5a7531be8a9 [Last accessed: Nov 10, 2024]

[8] Tee Meng Kiat,"Mengtee", April 2024. [Online] Available: https://github.com/mengtee/AdvanceRAG [Last accessed: Nov 10, 2024]

[9] Tarun Singh, "Advanced RAG Techniques: Unlocking the Next Level", April 2024. [Online] Available: https://medium.com/@krtarunsingh/advanced-rag-techniques-unlocking-the-next-level-040c205b95bc [Last accessed: Nov 10, 2024]

SIT LIBRARY 2025

[10] LlamaIndex [Online] Available: https://docs.llamaindex.ai/en/latest/ [Last accessed: Nov 10, 2024]

[11] Langchain [Online] Available: https://python.langchain.com/docs/introduction/ [Last accessed: Nov 10, 2024]

[12] DeepEval, "Answer Relevancy," 2025. [Online]. Available: https://www.deepeval.com/docs/metrics-answer-relevancy.

[13] DeepEval, "Contextual Recall," 2025. [Online]. Available: https://www.deepeval.com/docs/metrics-contextual-recall.

[14] DeepEval, "Contextual Precision," 2025. [Online]. Available: https://www.deepeval.com/docs/metrics-contextual-precision.

[15] DeepEval, "Summarization," 2025. [Online]. Available: https://www.deepeval.com/docs/metrics-summarization

[16] DeepEval, "Faithfulness," 2025. [Online]. Available: https://www.deepeval.com/docs/metrics-faithfulness

[17] DeepEval, "Toxicity," 2025. [Online]. Available: https://www.deepeval.com/docs/metrics-toxicity

[18] DeepEval, "Bias," 2025. [Online]. Available: https://www.deepeval.com/docs/metrics-bias

SFIT LIBRARY 2025