LARGE LANGUAGE MODEL WITH FACIAL RECOGNITION

FOO WAI HONG

UNIVERSITI TUNKU ABDUL RAHMAN

LARGE LANGUAGE MODEL WITH FACIAL RECOGNITION

FOO WAI HONG

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Electrical and Electronics Engineering with Honours

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2026

DECLARATION STATEMENT

I hereby declare that this project report is my original work, with all sources duly acknowledged, and that it has not been submitted previously or concurrently for any degree or award at UTAR or elsewhere. I also declare the following *(check the boxes if applicable)*:

☐ Generative AI tools (e.g., ChatGPT, Gemini, Copilot, etc.) were used for language refinement, such as grammar correction and readability improvement.

☐ Vibe coding tools and/or frameworks (e.g., Replit Agent, Cursor Composer, Bolt, etc.) were used in project development.

☐ Other declaration(s), if any (*please consult your supervisor for guidance in completing this section*):

_____

_____

I take full responsibility for the originality, accuracy, and integrity of all content presented in this report. No Generative AI tools were used to manipulate project data and results in ways that could cause falsification or misrepresentation.

Name : FOO WAI HONG

ID No. : 1902085

Date : 3/5/2026

COPYRIGHT STATEMENT

This final year project report is submitted in partial fulfilment of the requirements for the Bachelor of Electrical and Electronics Engineering with Honours. This final year project report represents the work of the author, except where due acknowledgement has been made in the text. No part of this final year project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ABSTRACT

Business intelligence is often about understanding complicated relationships between financial data, business risks, companies, market factors, and operations. Financial information is often embedded in long documents, structured tables, and across multiple business entities, which makes it difficult for a single AI agent to analyse these relationships. In this project we propose a secure Multi-Agent RAG (Retrieval Augmented Generation) system for financial report analysis. Users can upload 10-K reports, register and log in with facial recognition and query financial questions through a chatbot interface. The backend stores the extracted text chunks, entities, relationships, and structured table rows from the uploaded reports into MongoDB, Milvus, and Neo4j. This project proposed a multi-agent retrieval pipeline that analyses each user query, routes it to appropriate retrieval agents, retrieves relevant text and table evidence, traverses graph relationships, supports financial calculations, rerank evidence and generates a final answer with citations. The system integrates vector retrieval, graph-based reasoning, table retrieval and agent-based coordination to help answer factual, comparative and analytical financial questions. The evaluation was done in two stages, retrieval evaluation, and Retrieval Augmented Generation Assesssment (RAGAS) evaluation. We evaluated the system on 150 examples of financial question answering, consisting of 60 factual, 48 comparative and 42 analytical questions. In the retrieval evaluation, the system achieved 100.00% pass rate, 99.33% Hit@5, and 100.00% Recall@10. For the standard RAGAS evaluation, it scored 97.73% faithfulness, 100.00% context recall, 87.43% answer relevancy, 70.09% answer correctness. The results reveal that the proposed system is capable of retrieving relevant financial evidence and generating responses that are mostly faithful and grounded in context. The results indicate that the proposed facial recognition authentication and Multi-Agent Graph RAG can enable secure, efficient, and explainable analysis of financial reports. However, we recommend further improvements to improve the correctness of answers, strengthen analytical reasoning, and expand the system to support more financial reports and user scenarios. This project highlights the potential of secure multi-agent Graph RAG systems for business intelligence and financial decision-making.

Keywords: Large Language Model; Facial Recognition; Multi-Agent System; Graph Retrieval-Augmented Generation; Financial Report Analysis; Question Answering; Natural Language Processing; Vector Database; Knowledge Graph; Artificial Intelligence

Subject Area: QA75.5-76.95 Electronic computers. Computer science

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

D	the document corpus or knowledge base
$d_1, d_2, \dots, d_N$	the individual documents or data points within the set
N	total number of stored documents
$D_k(q)$	set of the top k retrieved documents for given query q
$TopK$	the operation that selects the highest-ranked elements
$s(q, d_i)$	scoring function between query q and document d_i
k	the specific number of documents to be retrieved
$p(y\|q)$	probability of generating the final output sequence y given query q
$p_\eta(d\|q)$	the retrieval probability
$p_\theta(y\|q, d)$	the generation probability
$S_{hybrid}(q, d)$	the final combined similarity score between query q and document d
$\tilde{S}_{BM25}(q, d)$	keyword score (Sparse), calculated by BM25 algorithm
$\tilde{S}_{dense}$	semantic score (Dense), calculated by using cosine similarity or inner product of embeddings
λ	weighting factor (0 to 1)
q	input query
q	document chunk
e_q	query embedding
e_d	document embedding
$s(q, d)$	cosine similarity metric
$e_q \cdot e_d$	vector dot product
$\|e_q\| \, \|e_d\|$	normalization factor
$D_k(q)$	the resulting subset of the k most relevant document for query q
$TopK$	the operation that selects the highest-ranked elements
$s(q, d_i)$	scoring function between query q and document d_i
k	the specific number of documents to be retrieved

e_q	query embedding
e_{di}	embedding of the document chunk d_i
$s_{dense}(q, d_i)$	semantic similarity score between query q and document d_i
$e_q \cdot e_{di}$	dot product of the two embedding vectors
$\|e_q\| \, \|e_{di}\|$	product of the magnitudes used for normalization
$C_{\text{final}}(q)$	final set of n documents
TopN	selection of top n results
$C_k(q)$	initial top-k candidate set
$R(q, d_i)$	reranking score function
G	Knowledge Graph structure
V	set of vertices (nodes/entities) in the graph
(v_i, r, v_j)	specific triplet representing a start node, relationship and end node
v_i, v_j	individual vertices in V
$N_h(v_q)$	h-hop neighborhood of the query node v_q
v_q	query-related node
$v \in V$	set of nodes belonging to the total vertices of the graph
$dist(v_q, v)$	graph distance between nodes
h	maximum hop distance for the neighborhood search
z_i	output face embedding vector
x_i	input face image
$f_\theta(\cdot)$	face recognition model with parameter θ
$T(\cdot)$	alignment/transformation function applied to the detected face
$D(\cdot)$	face detection
$\mathbb{R}^d$	Embedding space of dimension d
$d_{\cos}$	cosine distance between a query face and a stored identity
q	embedding vector of the query face

e_u	embedding vector of the enrolled user u stored in the database
$q \cdot e_u$	dot product between query and stored vector
$\|q\|_2$	L2 norm (Euclidean magnitude) of query vector
$\|e_u\|_2$	L2 norm (Euclidean magnitude) of enrolled user vector
e_u	final user template embedding (stored in database)
z_i	individual face embedding
n	number of face samples collected
$\sum_{i=1}^{n} z_i$	sum of all embeddings
$\frac{1}{n}$	average operation
u^*	best matching user
$\arg min_u$	finds the closest match
q	live face embedding captured during login
e_u	stored average embedding for user u
τ	threshold for acceptance
FAR	false acceptance rate
FRR	false rejection rate
FP	impostor incorrectly accepted
TN	impostor correctly rejected
FN	genuine user incorrectly rejected
TP	genuine user incorrectly accepted
N	total number of evaluation questions
R_i^k	top-k retrieved chunks for question i
G_i	set of relevant gold evidence for question i
$1(\cdot)$	indicator function, equal to 1 if the condition is true
Recall@k	proportion of gold evidence retrieved within the top-k results
$\left\|R_i^k \cap G_i\right\|$	number of retrieved chunks that match the gold evidence
G_i	total number of gold evidence items for question i
N	total number of queries in the evaluation set
rank_i	rank position of the first relevant retrieved chunk for

	question i
$NDCG@k$	Normalised Discounted Cumulative Gain at top-k
$DCG_i@k$	discounted gain of the retrieved ranking
$IDCG_i@k$	ideal discounted gain for the best possible ranking
g_{ij}	relevance score of result j for question i
AC_i	answer correctness score for question i
λ	weighting factor between factual and semantic companents
F_i^{ans}	factual agreement scores between generated and reference answer
S_i^{ans}	semantic similarity score between generated and reference answer
AR_i	answer relevancy score for question i
m	number of generated evaluator questions
q_i	original use question
$\hat{q}_{ij}$	question inferred from the generated answer
$E(\cdot)$	embedding function
$Faithfulness_i$	faithfulness score for question i
S_i	set of statements extracted from the generated answer
V_i	subset of statements supported by the retrieved context
$ContextPrecision_i@K$	context precision for question i at top-k
$P_i@j$	precision at rank j
v_{ij}	relevance label of retrieved context j
K	number of retrieved contexts considered
$ContextRecall_i$	context recall score for question i
C_i^{ref}	required information units covered by the retrieved context
T_i^{ref}	information units required by the reference answer

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Financial 10-K reports are important sources of information for understanding a company's business performance, risks, financial position and future direction. However, these reports are usually long, dense and difficult to analyse manually, especially when users need to compare figures, trace explanations, or understand relationships between business events and financial outcomes.

Large Language Model (LLM) can help users ask questions in natural language, but a standalone model may generate answers which are not fully grounded in the original document. Retrieval-Augmented Generation (RAG) solves this problem by gathering relevant evidence before generating a response, allowing the response to be based on external knowledge rather than model memory alone (Lewis et al., 2020). Graph RAG extends this idea by using graph structures to represent entities and relationships, which is useful when the answer depends on connections between companies, financial metrics, risks and report sections (Edge et al., 2024).

This project develops a Multi-Agent Graph RAG system for financial report analysis with facial recognition as the authentication layer. The system combines document ingestion, vector retrieval, graph retrieval, calculation support and answer synthesis to help users ask factual, comparative and analytical questions. Facial recognition is included to control access to the system, using face embeddings for identity verification, a method commonly used in modern face recognition systems (Deng, Guo and Zafeiriou, 2018).

1.2 Importance of the Study

The study is important because financial reports are often difficult for non-expert users to read efficiently. A user may need to search across many pages, compare different sections, identify financial figures, and understand how one piece of information relates to another. This creates a need for a system that can support faster, clearer and more evidence-based report analysis.

Retrieval-Augmented Generation (RAG) is useful in this context because it allows the system to retrieve relevant report content before generating an answer (Lewis et al., 2020). GraphRAG adds another layer by storing relationships between entities, topics and financial concepts, which helps the system reason over connected information instead of treating every chunk as isolated text (Edge et al., 2024). This is especially useful for financial questions that involve causes, comparisons, trends or relationships.

The facial recognition component also adds practical value because financial documents may contain sensitive business information. Instead of allowing open access, the system requires users to register and log in through face authentication. This supports a more secure workflow while keeping the user experience simple.

1.3 Problem Statement

Many financial report analysis systems still depend heavily on keyword search or manual reading. These methods are limited because users must know the correct terms to search for and still need to interpret the results themselves. When the report is long, this process becomes slow and error-prone.

A general LLM can summarise or answer questions, but it may generate unsupported statements if it is not connected to the actual report content. This is a serious issue in financial analysis because users need answers that are traceable to the source material. Financial questions may also require calculations, table lookup, comparison across companies, or reasoning over relationships, which cannot be handled reliably by simple text generation alone.

Therefore, the problem addressed in this study is the need for a secure, evidence-grounded and multi-agent system that can answer financial report questions accurately. The system must retrieve relevant evidence, use graph relationships when needed, perform calculations where appropriate, and generate answers with supporting citations. This follows the idea that language models become more useful when they can reason and act with external tools rather than relying only on internal knowledge (Yao et al., 2022).

1.4 Aim and Objectives

The aim of this project is to design and develop a secure Multi-Agent Graph RAG system for financial 10-K report analysis. The system allows authenticated users to upload financial reports, ask natural-language questions, and receive answers that are grounded in retrieved document evidence. It combines vector retrieval, graph retrieval, calculation support, and response synthesis within a multi-agent workflow, while facial recognition is used to protect access to the application.

Project Objectives:

1. To develop a document ingestion pipeline that can process financial 10-K reports and prepare their content for retrieval and analysis.
2. To design a Multi-Agent Graph RAG workflow that supports factual, comparative and analytical financial questions.
3. To integrate vector, graph and metadata storage so that the system can retrieve relevant text, relationships, tables and document information.
4. To develop facial recognition for user registration and login, to guarantee that only authenticated users are permitted to use the system.
5. To evaluate the system using retrieval evaluation and RAGAS evaluation, where retrieval performance is measured first and answer quality is evaluated in the next evaluation layer.

1.5 Scope and Limitation of the Study

This study focuses on a web-based system for dealing with questions about financial reports. Users can authenticate through facial recognition, upload documents, view processed reports, and ask questions through a chatbot interface. The system focuses on factual, comparative and analytical questions related to financial report content.

The technical scope includes FastAPI for the backend, React with Vite for the frontend, MongoDB for metadata and application records, Milvus for vector retrieval, Neo4j for graph relationships, and DeepFace for facial recognition. The retrieval system uses multiple agents to route and process questions, while the final answer is determined by retrieving evidence and using structured reasoning. This design is aligned with RAG and graph-based retrieval approaches discussed in previous research (Lewis et al., 2020; Edge et al., 2024).

There are also limitations. The system does not replace professional financial judgement and should not be treated as financial advice. Its answer quality depends on the uploaded documents, extraction quality and retrieval performance. In addition, the current evaluation is not fully complete because the retrieval layer has been evaluated, but the RAGAS layer is still ongoing.

1.6 Contribution of the Study

This study contributes a chatbot system that combines Multi-Agent RAG, graph-based retrieval and facial recognition in one application. Instead of only providing a chatbot over documents, the system separates responsibilities across agents so that retrieval, graph search, calculation and answer synthesis can be handled more clearly. This makes the system easier to evaluate and improve.

Another contribution is the use of graph retrieval for financial report analysis. Financial reports often contain connected information, such as relationships between revenue, risks, business segments, markets and management discussion. By representing these connections in a graph, the system can support more structured retrieval than plain text search alone (Edge et al., 2024).

The study also contributes an evaluation workflow with two layers. The retrieval evaluation has achieved strong results, with all 150 retrieval test cases passing, hit@5 of 0.9933, recall@10 of 1.0000 and MRR of 0.8511. The RAGAS evaluation is still in progress and will later provide additional evidence on answer faithfulness, context relevance and response quality.

1.7 Outline of the Report

Chapter 1 introduces the project background, importance of the study, problem statement, aim and objectives, scope and limitations, contribution, and overall report structure. It explains why financial report analysis needs a secure and evidence-grounded question answering system.

Chapter 2 will review related work on Large Language Model (LLM), Retrieval-Augmented Generation (RAG), Graph RAG, multi-agent systems, financial document analysis and facial recognition. This chapter will also explain how existing research supports the design decisions used in this project.

Chapter 3 will describe the methodology and system design, including architecture, data flow, agent roles, storage components and authentication workflow.

Chapter 4 will present implementation and evaluation results, while Chapter 5 will conclude the project and discuss future improvements.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Chapter 2 reviews the literature needed to understand the design of a Multi-Agent Graph RAG (Retrieval-Augmented Generation) system for financial 10-K report analysis. The chapter begins with financial 10-K reports because they define the problem setting of this project. These reports contain narrative disclosures, financial tables, risk factors, management discussion, and notes, which makes them useful for analysis but difficult to search and interpret manually.

The chapter then discusses Large Language Model (LLM), RAG and GraphRAG. LLM can generate fluent answers, but they need grounding when the source material is long, specialised, and evidence-dependent. Retrieval-Augmented Generation addresses this by retrieving relevant document content before answer generation (Lewis et al., 2020). GraphRAG extends this idea by using graph structures to represent entities and relationships, which is useful when questions depend on links between companies, financial metrics, risks, years, and report sections (Edge et al., 2024).

The final part of the chapter reviews multi-agent workflows, evaluation methods, and facial recognition. Multi-agent design is relevant because financial report analysis can be separated into query routing, retrieval, graph reasoning, calculation, and answer synthesis, instead of being handled as one single step (Yao et al., 2022). Evaluation is also reviewed because a RAG system must be assessed based on retrieval quality, answer faithfulness, and context relevance, rather than fluency alone (Es et al., 2023). Facial recognition is discussed as the authentication layer of the system, where face embeddings can be used to verify user identity before allowing access to the financial analysis application (Deng, Guo and Zafeiriou, 2018).

2.2 Financial Reports and 10-K Document Analysis

2.2.1 Overview of Financial Reports

Financial reports are formal documents used by companies to communicate their financial performance, financial position, business activities and major risks. They usually combine numerical statements, such asbalance sheets, income statements and cash flow statements, with narrative explanations from management. This combination allows users to understand both the reported figures and the business reasons behind those figures.

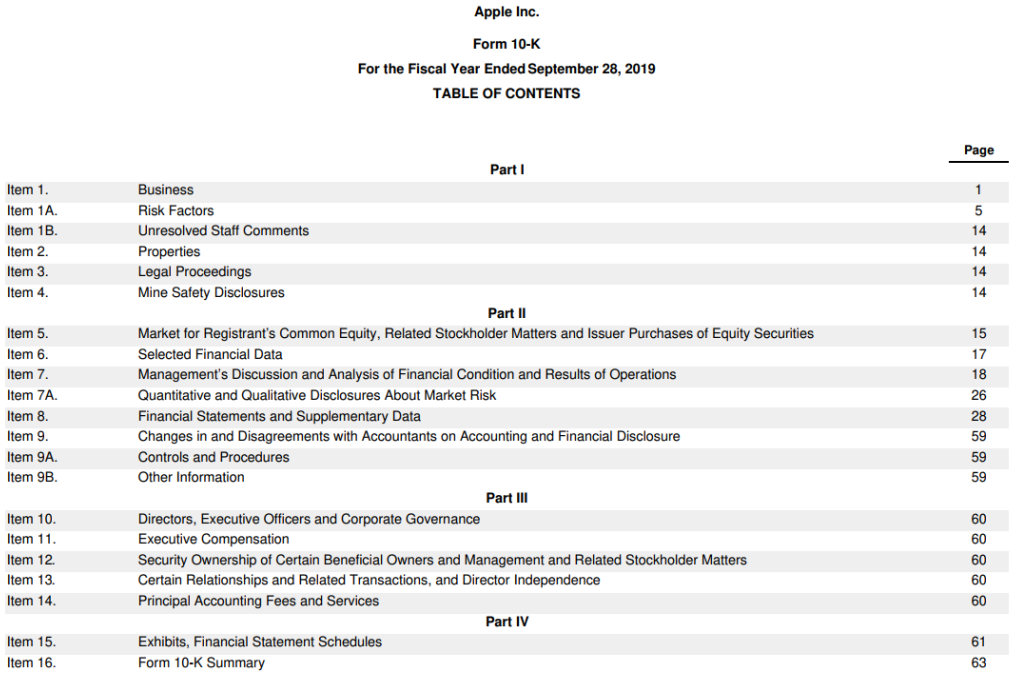
For financial question answering, these reports are useful because they contain both structured and unstructured information. However, this also makes them difficult to analyse automatically. A system must understand tables, accounting terms, written explanations, fiscal periods and units such as millions, billions or percentages. Prior work on financial question answering shows that real financial reports often require reasoning across both tables and text, especially when answers involve calculations or financial scales (Zhu et al., 2021).

2.2.2 Structure of Form 10-K Reports

Annual reports are submitted to the Securities and Exchange Commission of the United States using the Form 10-K. These reports are filed by publicly traded corporations. It includes a comprehensive overview of a company's operations, financial position, risks, management commentary, and audited financial statements (US Securities and Exchange Commission, 2025). The SEC delineates that essential components of a 10-K comprise Risk Factors, Financial Statements, Business and Management’s Discussion and Analysis (U.S. Securities and Exchange Commission, n.d.).

The standard structure of Form 10-K makes it suitable for systematic document analysis. Zhang et al. (2023) explain that Form 10-K reports are organised into four parts and 22 item sections, where each item focuses on a specific financial aspect of the company. This structure is useful for retrieval because section labels such as Risk Factors, MD&A and Financial Statements

can help a system locate the correct evidence instead of searching the entire document as plain text.



Apple Inc.

Form 10-K

For the Fiscal Year Ended September 28, 2019

TABLE OF CONTENTS

Figure 2.1: Table of Content of Form 10-K

2.2.3 PDF Parsing, Table Extraction and Document Chunking

PDF parsing is the first step in converting financial 10-K reports into content that a retrieval system can use. It should not be treated as simple text extraction because 10-K reports contain section headings, financial statements, footnotes, page headers, tables and captions. If the parser loses the reading order, section hierarchy or table boundaries, the retrieved evidence may lose its original meaning. Layout-aware document processing is therefore important because it preserves document elements such as titles, section headers, tables and footnotes before the content is used for retrieval (Pfitzmann et al., 2022; Auer et al., 2024).

Table extraction is also important because many financial facts are presented in rows and columns instead of full sentences. The system must preserve row labels, column labels, fiscal periods, units, negative values and footnotes so that each number can be interpreted correctly. For example, the same value can mean revenue, net income, assets or cash flow depending on the table label and reporting period. Financial question answering benchmarks such

as TAT-QA show that table-text reasoning is difficult because many questions require locating the correct table values and performing calculations or comparisons over them (Zhu et al., 2021).

The processed report is chunked into smaller retrieval units through the process of document splitting. For 10-K reports, chunking should preserve section context, page metadata, company information, fiscal year, headings and table context. A simple fixed-size split may separate a number from its label or a paragraph from its section heading, which can reduce retrieval accuracy. Structure-aware chunking is more suitable for financial RAG because it keeps related information together and gives retrieval agents cleaner evidence for answering user questions (Jimeno Yepes et al., 2024).

2.2.4 Challenge in Financial Report Analysis

Financial report analysis is challenging because Form 10-K reports are long, dense and variable in structure. Although the SEC provides a standard item structure, companies may use different wording, formatting and section presentation. Zhang et al. (2023) note that item titles can vary, some sections may be missing, and keywords may appear in headers, tables of contents or normal paragraphs, which makes simple keyword matching unreliable.

Another challenge is that financial meaning depends heavily on context. The same number can have different meanings depending on the metric, fiscal year, company, unit and section where it appears. Questions about revenue growth, operating margin or cash flow may require evidence from both tables and narrative explanations. TAT-QA shows that financial question answering often needs numerical reasoning over hybrid table-text content, including operations such as addition, subtraction, division, comparison and scale prediction (Zhu et al., 2021). Therefore, a reliable 10-K analysis system must preserve document structure, table context and source metadata before retrieval and answer generation.

2.3 Large Language Models

2.3.1 Overview of Large Language Models

Large Language Model (LLM) are neural language models trained on large collections of text to understand, generate and transform natural language. Most modern LLM are based on the Transformer architecture, which uses self-attention to capture relationships between words or tokens across a sequence (Vaswani et al., 2017). This design allows the model to process long and complex text more effectively than earlier recurrent models, making it suitable for analysing dense documents such as financial reports.

LLM also represent a shift from task-specific NLP systems to general-purpose pre-trained models. Instead of building a separate model for every task, an LLM can be prompted or adapted to perform many tasks, including summarisation, question answering and reasoning. Brown et al. (2020) showed that GPT-3 could perform several tasks in zero-shot and few-shot settings, where the model follows instructions or examples without task-specific retraining.

In this project, the LLM is not treated as the only source of knowledge. Its main role is to understand user questions, interpret retrieved financial evidence and generate readable answers. This is important because financial analysis requires exact figures, correct reporting periods and source traceability. Therefore, the LLM must work together with retrieval, graph relationships and evaluation components rather than relying only on its internal knowledge.

2.3.2 LLM for Financial Question Answering

Financial question answering is more difficult than general question answering because financial reports contain long text, tables, accounting terms, footnotes and numerical values. Questions may require factual lookup, comparison between years, or calculation using values found in different sections. FinQA shows that financial QA often needs multi-step numerical reasoning over financial reports, not just simple text extraction (Chen et al., 2021). TAT-QA also highlights the challenge of combining table and text evidence when answering financial questions (Zhu et al., 2021).

Domain-specific LLM have been explored to improve performance in financial NLP tasks. BloombergGPT, for example, was trained using a mixture of financial and general-domain data, and its evaluation showed that financial-domain training can improve performance on finance-related tasks (Wu et al., 2023). This supports the idea that financial language has its own vocabulary, structure and reasoning requirements.

For this project, LLM are useful because they can convert retrieved evidence into a natural answer that users can understand. However, financial QA cannot depend only on fluent text generation. The answer must be grounded in the correct report content, especially when dealing with values, percentages, risks or year-based comparisons. This is why the system uses a retrieval-grounded and graph-supported design.

2.3.3 Limitations of LLM-only System

An LLM-only system is risky for financial report analysis because it may produce answers that sound correct but are not supported by the source document. Hallucination is a known issue in natural language generation, where generated text may include unsupported or incorrect information (Ji et al., 2023). In finance, this problem is serious because a wrong amount, missing unit or incorrect fiscal year can change the meaning of the answer.

Another limitation is that LLM do not guarantee access to the latest or most relevant document evidence. Their knowledge depends on training data and prompt context, so they may answer from general memory instead of the specific filing being analysed. Lewis et al. (2020) argued that retrieval helps address this issue by connecting generation models to external knowledge sources, allowing answers to be based on retrieved evidence rather than model parameters alone.

LLM-only systems also struggle with source traceability and numerical reliability. Financial users often need to check where an answer came from, such as a report section, table or extracted passage. Without retrieval and citation support, the system cannot clearly show the evidence behind the answer. For

this reason, the proposed Multi-Agent Graph RAG system uses the LLM as one component within a larger workflow that includes retrieval, graph reasoning, source grounding and evaluation.

2.4 Retrieval-Augmented Generation (RAG)

2.4.1 Overview of RAG

Retrieval-Augmented Generation (RAG) is a technique for improving large language model replies by obtaining external evidence before coming up with an answer. Instead of relying only on the knowledge stored within the model, RAG enables the model to refer to relevant documents during inference time. This is useful for financial report analysis because answers often need to be supported by a specific filing section, table value, reporting year, or explanatory note. Lewis et al. (2020) delivered RAG as a framework that combines a retriever with a generator so that generated answers can be grounded in external sources. Given a document collection:

$$D = \{d_1, d_2, \dots, d_N\} \tag{2.1}$$

where

D = the document corpus or knowledge base

$d_1, d_2, \dots, d_N$ = the individual documents or data points within the set

N = total number of stored documents

and a use query q, the retriever selects the most relevant evidence from the collection:

$$D_k(q) = TopK_{d_{i\epsilon D}}(s(q, d_i), k) \tag{2.2}$$

where

$D_k(q)$ = set of the top k retrieved documents for given query q

$TopK$ = the operation that selects the highest-ranked elements

$s(q, d_i)$ = scoring function between query q and document d_i

k = the specific number of documents to be retrieved

In this project, a document unit may refer to a text chunk, table segments, filing section, or extracted financial evidence rather than only a full report.

The generator then generates an answer y based on the question and retrieved evidence:

$$p(y|q) = \sum_{d \epsilon D_k(q)} p_\eta(d|q) p_\theta(y|q,d) \quad (2.3)$$

where

$p(y|q)$ = probability of generating the final output sequence y given query q

$p_\eta(d|q)$ = the retrieval probability

$p_\theta(y|q,d)$ = the generation probability

This formulation matches the goal of the project because the system should answer form the retrieved financial evidence rather than from unsupported model memory.

2.4.2 Retrieval-Augmented Generation (RAG) Pipeline

A RAG pipeline usually includes document ingestion, chunking, indexing, retrieval, context construction, and answer generation. During ingestion, long reports are converted into smaller searchable units. These units are indexed so that the system can retrieve relevant content when a user submits a query. Modern RAG systems may also include query processing, reranking, context filtering, and evaluation stages to improve retrieval and answer quality (Gao, Y. et al., 2023).

Dense retrieval is commonly used in RAG because it represents both the query and document units as embeddings. Dense Passage Retrieval shows that query and passage embeddings can be compared using inner product similarity (Karpukhin et al., 2020):

$$S_{hybrid}(q,d) = \lambda \tilde{S}_{BM25}(q,d) + (1-\lambda)\tilde{S}_{dense}(q,d) \qquad (2.4)$$

where

$S_{hybrid}(q,d)$ = the final combined similarity score between query q and document d

$\tilde{S}_{BM25}(q,d)$ = keyword score (Sparse), calculated by BM25 algorithm

$\tilde{S}_{dense}$ = semantic score (Dense), calculated by using cosine similarity or inner product of embeddings

λ = weighting factor (0 to 1)

Here, λ controls the balance between lexical and dense retrieval. After the most relevant evidence is retrieved, it is added to the prompt so that the language model can generate a grounded response.

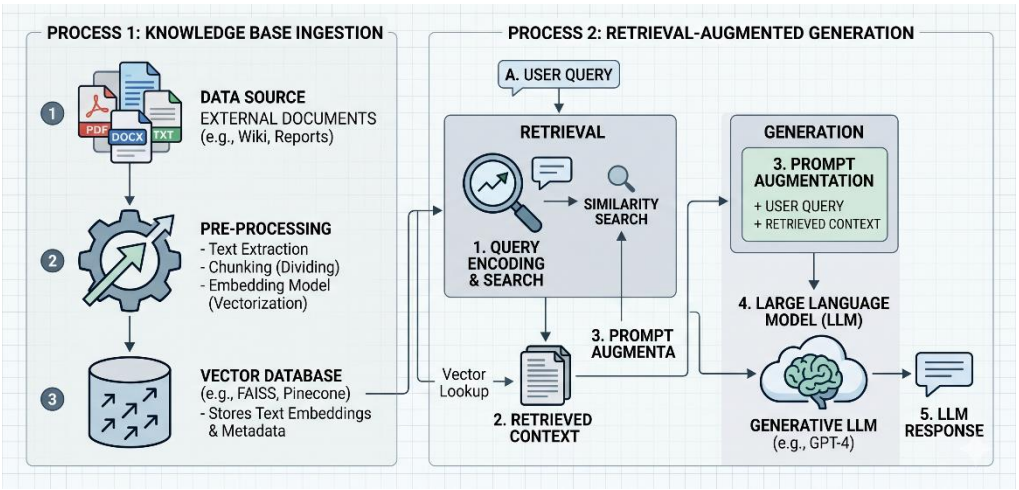


Figure 2.2: RAG Pipeline Workflow

2.5 Vector Retrieval and Embeddings

2.5.1 Text Embeddings

Text embeddings are numerical vector representations of natural language. In a RAG system, they allow user questions and document chunks to be compared in a shared semantic space. Instead of relying only on exact keyword overlap, the system can retrieve passages that are similar in meaning. This is useful for financial 10-K reports because users may ask about “revenue growth”, while the report may use related terms such as “net sales”, “total revenues”, or “year-over-year increase”.

Sentence-BERT demonstrates the way transformer models can be changed to provide significant embeddings of sentences that can be efficiently evaluated with similarity measures such as cosine similarity (Reimers and Gurevych, 2019). In this project, the same idea is applied to financial report chunks. At runtime, the user's query is embedded, and each chunk is converted into vector embedding and stored in the databases for retrieval.

Formally, an embedding model $f(\cdot)$ maps a query q and document chunk d into vectors:

$$e_q = f(q), \quad e_d = f(d) \tag{2.5}$$

where

q = input query

q = document chunk

e_q = query embedding

e_d = document embedding

The relevance between the query and chunk can then be estimated using cosine similarity:

$$s(q,d) = \frac{e_q \cdot e_d}{\|e_q\| \, \|e_d\|} \tag{2.6}$$

where

$s(q, d)$ = cosine similarity metric

$e_q \cdot e_d$ = vector dot product

$\|e_q\| \|e_d\|$ = normalization factor

This allows the system to retrieve semantically related content even when the wording is not exactly the same.

2.5.2 Vector Databases

The embeddings are stored in a vector database, which also allows for a quick search for similarity. After the financial reports are parsed and chunked, each chunk is embedded and inserted into the vector store together with metadata such as company name, filing year, section, page number, and table identifier. This metadata is important because financial answers often depend not only on semantic similarity, but also on the correct company, year, and reporting context.

Vector retrieval normally follows a two-step process. First, all document chunks are embedded and indexed. Second, when a query is submitted, the query is embedded and compared against the stored vectors. The top-k closest chunks are then retrieve by the system:

$$D_k(q) = TopK_{d_{i \epsilon D}}(s(q, d_i), k) \tag{2.7}$$

where

$D_k(q)$ = the resulting subset of the k most relevant documents for query q

$TopK$ = the operation that selects the highest-ranked elements

$s(q, d_i)$ = scoring function between query q and document d_i

k = the specific number of documents to be retrieved

FAISS is a widely used library for efficient similarity search over dense vectors and supports approximate nearest neighbour search for large-scale vector collections (Douze et al., 2024). In a financial RAG system, this type of

vector index helps reduce retrieval time when searching across many report chunks.

In the end, vector databases should not be treated as complete financial reasoning systems. They retrieve candidate evidence, but they do not guarantee that the selected chunks contain the correct financial value, year, or unit. Therefore, vector retrieval should be combined with metadata filtering, reranking, and later graph-based reasoning.

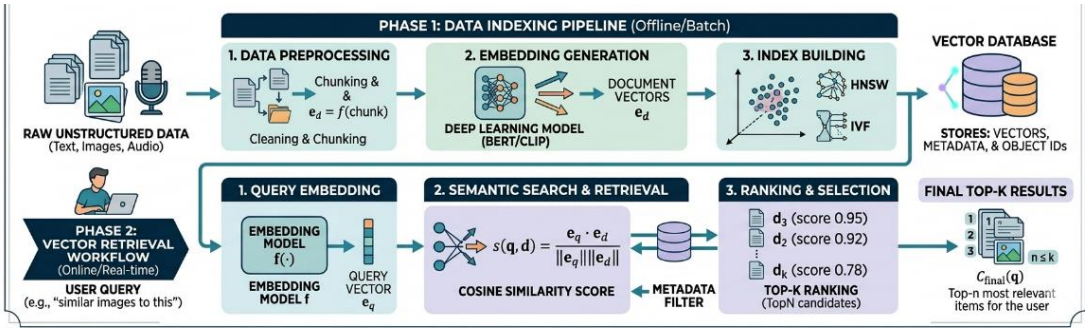


Figure 2.3: Vector Embedding and Retrieving Workflow

2.5.3 Ranking and Reranking

Ranking determines which retrieved chunks are passed to the answer generation stage. In vector retrieval, the query and each document chunk are represented as embeddings, then compared using a similarity score. A common dense ranking score is cosine similarity:

$$s_{dense}(q, d_i) = \frac{e_q \cdot e_{di}}{\|e_q\| \, \|e_{di}\|} \tag{2.8}$$

where

e_q = query embedding

e_{di} = embedding of the document chunk d_i

$s_{dense}(q, d_i)$ = semantic similarity score between query q and document d_i

$e_q \cdot e_{di}$ = dot product of the two embedding vectors

$\|e_q\| \, \|e_{di}\|$ = product of the magnitudes used for normalization

Dense retrieval is useful because it can identify semantically related evidence even when the user's wording differs from the wording in the 10-K report (Karpukhin et al., 2020). Efficient vector search libraries such as FAISS also make it practical to rank large numbers of embedded chunks quickly (Douze et al., 2024).

However, the highest similarity score does not always mean the chunk is the best evidence. In financial reports, a retrieved passage may be broadly related to the query but still refer to the wrong company, year, section, table, or financial metric. Therefore, reranking is used to refine the initial retrieval result. If $C_k(q)$ is the initial top-k candidate set, the final context can be selected as:

$$C_{\text{final}}(q) = \text{TopN}_{d_i \in C_k(q)}(R(q, d_i), n), \quad n \leq k \tag{2.9}$$

where

$C_{\text{final}}(q)$ = final set of n documents

TopN = selection of top n results

$C_k(q)$ = initial top-k candidate set

$R(q, d_i)$ = reranking score function

The reranker score $R(q, d_i)$ and number of chunks n finally passed to the language model. In this project, reranking helps reduce noisy context and prioritise chunks that directly support the financial answer, especially when the question depends on exact figures, reporting periods, or table-based evidence.

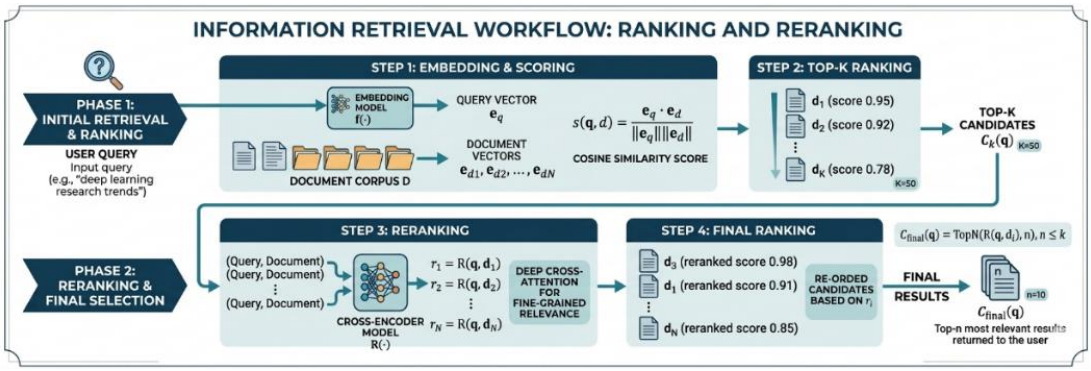


Figure 2.4: Workflow of Ranking and Reranking

2.6 Graph Retrieval-Augmented Generation

2.6.1 Definition of Graph RAG and Knowledge Graph

Graph RAG extends normal RAG by adding graph-structured knowledge into the retrieval process. While vector retrieval is useful for finding semantically similar text chunks, it may struggle when the answer depends on relationships across entities, sections, years, or financial concepts. Graph RAG solves this by representing information as connected nodes and edges, allowing the system to retrieve not only relevant text, but also the relationships between pieces of evidence.

A knowledge graph may also be thought of as an organised graph where nodes represent entities and edges represent relationships between them. Hogan et al. (2021) describe knowledge graphs as a way to represent real-world entities and their relations in a machine-readable form. In the context of financial report analysis, the graph may include entities such as companies, reporting years, financial metrics, business segments, risk factors, tables, and document sections. This can be expressed as:

$$G = (V, E), \quad E = \{(v_i, r, v_j) \mid v_i, v_j \in V, r \in R\} \tag{2.10}$$

where

G = Knowledge Graph structure

V = set of vertices (nodes/entities) in the graph

(v_i, r, v_j) = specific triplet representing a start node, relationship and end node

v_i, v_j = individual vertices in V

Edge et al. (2024) show that Graph RAG can support broader understanding of large document collections by extracting entities and relationships, then using the graph structure to guide retrieval and summarisation. This makes Graph RAG suitable for financial 10-K analysis because important evidence is often spread across different sections rather than contained in one paragraph.

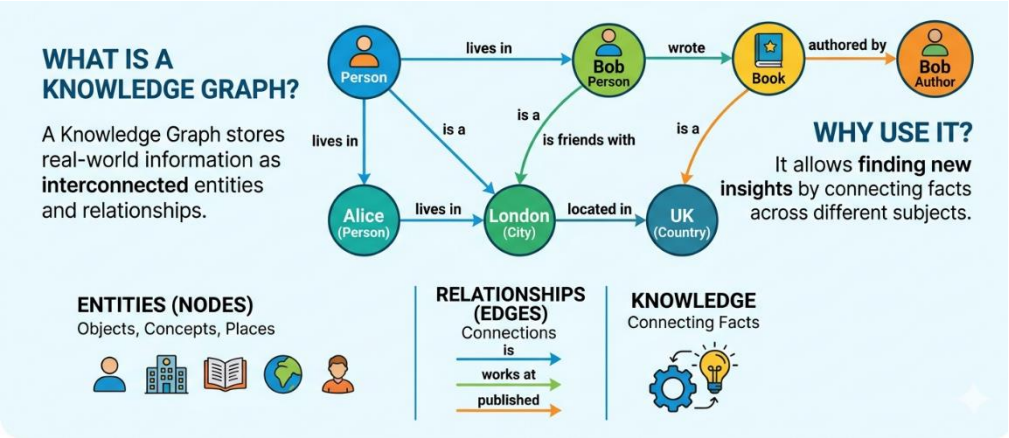


Figure 2.5: Overview of Knowledge Graph

2.6.2 Financial Entity and Relationship Modelling

Financial 10-K reports contain dense and highly connected information. A single answer may depend on a company name, fiscal year, accounting metric, business segment, currency unit, risk disclosure, and supporting table. Therefore, financial information should not only be stored as plain text chunks, but also modelled as entities and relationships. For example, a graph can connect a company to its reported revenue, link the revenue to a fiscal year, and connect the value back to the exact filing section or table.

This type of modelling is important because financial terms are often repeated across different contexts. The same term, such as “revenue”, may refer to total revenue, product revenue, service revenue, segment revenue, or year-over-year revenue. Without relationship modelling, a retrieval system may retrieve a relevant-looking passage but still select the wrong figure or reporting period. A graph structure helps reduce this risk by keeping the relationship between entities, values, dates, and sources explicit.

Shah, Ryali and Venkatesh (2024) show this direction in multi-document financial question answering by using semantic tagging and a knowledge graph RAG approach for 10-K reports. Their work supports the idea that knowledge graph triples can provide useful structured context for financial QA, especially when the question requires links between multiple entities or documents. For this project, the same principle supports the use of graph-based

retrieval together with vector retrieval, so that answers can be grounded in both semantic similarity and financial relationships.

2.6.3 Graph Retrieval for Multi-hop Financial Reasoning

Many financial questions require multi-hop retrieval. For example, a user may ask how a risk factor affects a company's business segment, or how a financial metric changed across years and what management explanation was given. These questions cannot always be answered by retrieving one isolated chunk. The system needs to follow a chain of related evidence, such as company, segment, metric, year, disclosure and finally source section.

Graph retrieval is useful for this because it can search around relevant entities and follow connected paths. A simple graph neighbourhood can be written as:

$$N_h(v_q) = \{v \in V \mid dist(v_q, v) \leq h\} \tag{2.11}$$

where

$N_h(v_q)$ = h-hop neighborhood of the query node v_q

v_q = query-related node

$v \in V$ = set of nodes belonging to the total vertices of the graph

$dist(v_q, v)$ = graph distance between nodes

h = maximum hop distance for the neighborhood search

In practice, this means the system can start from an entity found in the user query, expand to nearby financial concepts, retrieve connected evidence, and pass the most relevant graph context to the language model.

For a multi-agent Graph RAG system, this also creates a clear division of work. One agent can identify financial entities, another can perform graph traversal, another can retrieve text or table evidence, and a final agent can synthesise and validate the answer. However, Graph RAG should not be treated as a replacement for vector retrieval. Its main value is in relationship-heavy and

multi-hop questions, while direct factual questions may still be handled effectively through chunk or table retrieval.

2.7 Multi-Agent LLM System

2.7.1 Overview of LLM-Based Multi-Agent Systems

LLM-based multi-agent systems use multiple language-model agents to complete a task through role separation and communication. Instead of relying on one model to perform every step, the system divides the work into specialised components, such as query understanding, retrieval, reasoning, validation and answer generation. This design is useful when the task is too broad or complex for a single prompt-based workflow.

Guo et al. (2024) describe LLM-based multi-agent systems as systems where agents may have different profiles, communication patterns and interaction roles. In this project, the agents are not treated as human-like collaborators, but as system components with specific responsibilities. For example, one agent may retrieve text chunks, another may retrieve graph relationships, and another may check whether the generated answer is supported by evidence.

This approach is relevant to financial 10-K analysis because the task often involves long documents, tables, financial entities, relationships and source citations. A single-agent system may retrieve useful context but still miss the correct year, metric, company or reporting section. A multi-agent design provides a more controlled way to separate retrieval, reasoning and verification before producing the final answer.

2.7.2 Reasoning, Tool Use and Agent Coordination

LLM agents become more useful when they can reason and interact with external tools. ReAct shows that language models can interleave reasoning steps with actions, such as searching, observing and updating the next step based on retrieved information (Yao et al., 2023). ReAct is not a multi-agent framework by itself, but it provides an important reasoning-and-acting pattern that can support the behaviour of individual agents inside a larger system.

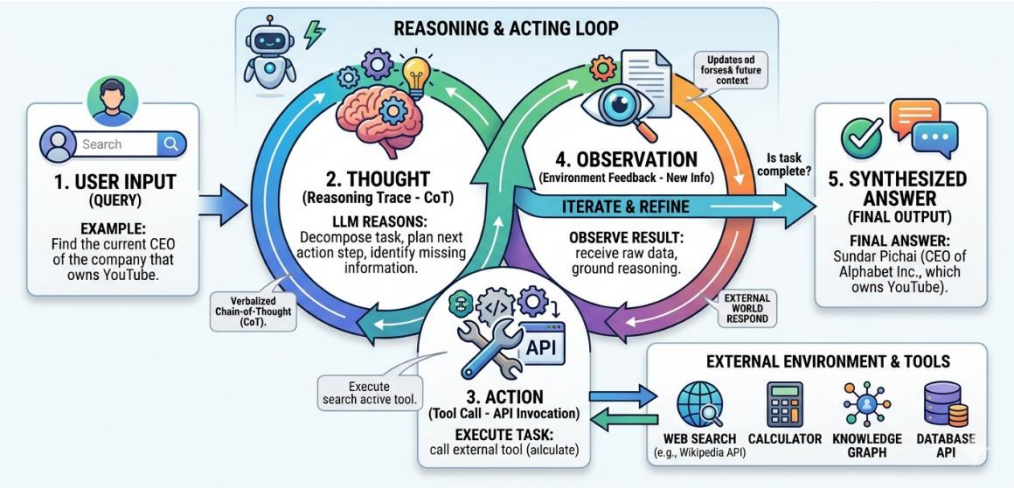


Figure 2.6: ReAct (Reasoning + Acting) Workflow

AutoGen extends this idea into multi-agent conversation, where agents can be configured with different roles, tools and interaction patterns (Wu et al., 2023). This is useful for building systems where one agent can ask another agent for retrieval, validation or code execution. In a Graph RAG pipeline, this allows the system to coordinate between text retrieval, graph traversal and final response generation.

However, coordination must be designed carefully. More agents do not automatically produce better answers. If agents pass noisy or incomplete information to each other, the final response may still contain unsupported claims. Therefore, a multi-agent workflow should define clear agent roles, controlled message formats and stopping conditions, especially in financial question answering where accuracy and traceability are important.

2.7.3 Multi-Agent RAG for Financial 10-K Question Answering

Multi-agent RAG separates the question-answering process into specialised stages instead of depending on a single retrieval and generation step. In financial 10-K analysis, this is useful because an answer may require narrative disclosures, table values, numerical checks, entity relationships and source citations. RAGentA demonstrates how multi-agent RAG can use separate agents for document filtering, attributed answer generation and answer revision to improve grounded question answering (Besrour et al., 2025). In this project, the same

idea supports a workflow where retrieval, graph reasoning, calculation checking and final response generation are handled by different agents.

The main benefit of this design is traceability. Each agent can produce intermediate evidence before the final answer is generated, making it easier to inspect which report sections, tables or graph relationships support the response. This is important in financial QA because small errors in year, metric, unit or company context can change the meaning of an answer. However, multi-agent RAG also increases runtime, cost and coordination complexity. Therefore, its effectiveness should be evaluated through retrieval relevance, citation accuracy, numerical faithfulness, answer quality and response time.

2.8 Facial Recognition Authentication Using DeepFace

2.8.1 Face Detection, Embeddings and Face Matching

Facial recognition is used in this project as an access control layer before users enter the Multi-Agent Graph RAG system. It does not improve retrieval or answer generation directly. Its role is to check whether the person using the system matches a registered face profile.

DeepFace is used as the recognition component because it supports face detection, alignment, embedding extraction and matching in one Python framework. In this project, the face image is processed into a numerical embedding using a selected recognition model such as ArcFace (Serengil and Ozpinar, 2020; Deng et al., 2019).

$$z_i = f_\theta\left(T\left(D(x_i)\right)\right), \quad z_i \in \mathbb{R}^d \tag{2.12}$$

where

z_i = output face embedding vector

x_i = input face image

$f_\theta(\cdot)$ = face recognition model with parameter θ

$T(\cdot)$ = alignment/transformation function applied to the detected face

$D(\cdot)$ = face detection

$\mathbb{R}^d$ = Embedding space of dimension d

FaceNet introduced this embedding-based approach by learning a space where distance reflects face similarity (Schroff, Kalenichenko and Philbin, 2015).

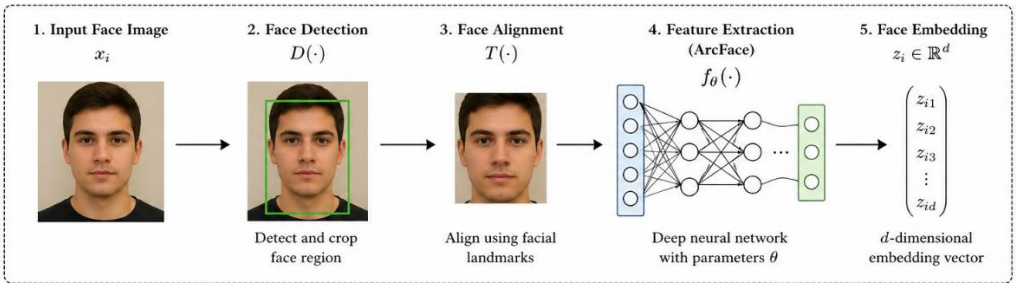


Figure 2.7: Face Recognition Pipeline

Face matching can then be performed using cosine distance:

$$d_{\cos}(q, e_u) = 1 - \frac{q \cdot e_u}{\|q\|_2 \|e_u\|_2} \tag{2.13}$$

where

$d_{\cos}$ = cosine distance between a query face and a stored identity

q = embedding vector of the query face

e_u = embedding vector of the enrolled user u stored in the database

$q \cdot e_u$ = dot product between query and stored vector

$\|q\|_2$ = L2 norm (Euclidean magnitude) of query vector

$\|e_u\|_2$ = L2 norm (Euclidean magnitude) of enrolled user vector

A lower distance means the login face is closer to the stored user template. This should be treated as a probability-based matching decision, not absolute proof of identity.

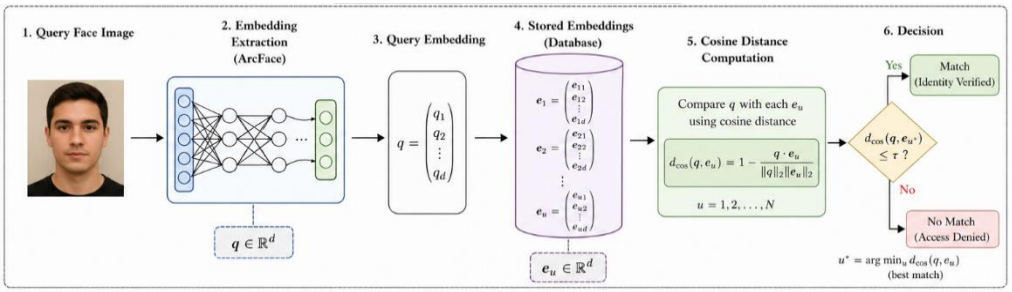


Figure 2.8: Face Matching using Cosine Distance

2.8.2 User Registration and Login Flow

During registration, the user captures several face samples. Each sample is converted into an embedding, and the stored user template can be represented as the average embedding:

$$e_u = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{2.14}$$

where

e_u = final user template embedding (stored in database)

z_i = individual face embedding

n = number of face samples collected

$\sum_{i=1}^{n} z_i$ = sum of all embeddings

$\frac{1}{n}$ = average operation

This reduces reliance on a single image and helps handle small changes in lighting, expression or camera angle. The system stores the embedding template with the user account instead of storing raw face images.

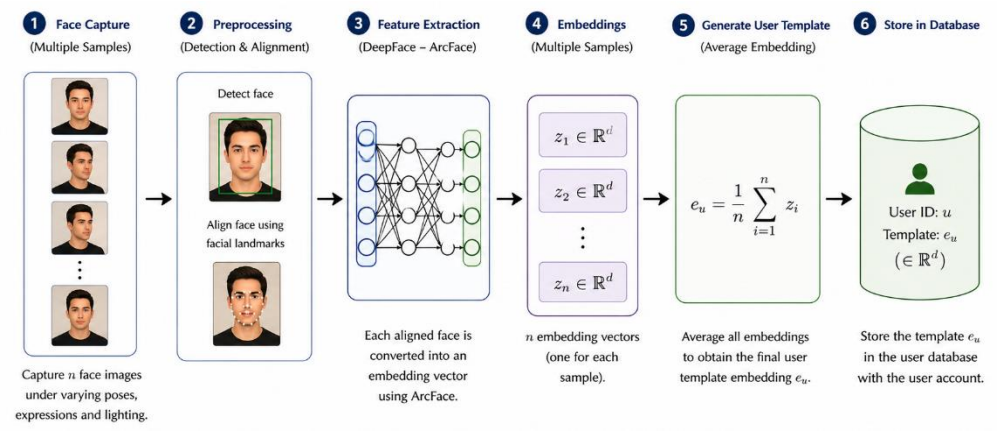


Figure 2.9: User Face Registration Workflow

During login, a new face embedding q is compared with stored templates. The system accepts the closest matching profile only if the distance is below the threshold τ,

$$u^* = \arg min_u \, d_{\cos}(q, e_u), \quad \text{Accept if } d_{\cos}(q, e_{u^*}) \leq \tau \qquad (2.15)$$

where

u^* = best matching user

$\arg min_u$ = finds the closest match

q = live face embedding captured during login

e_u = stored average embedding for user u

τ = threshold for acceptance

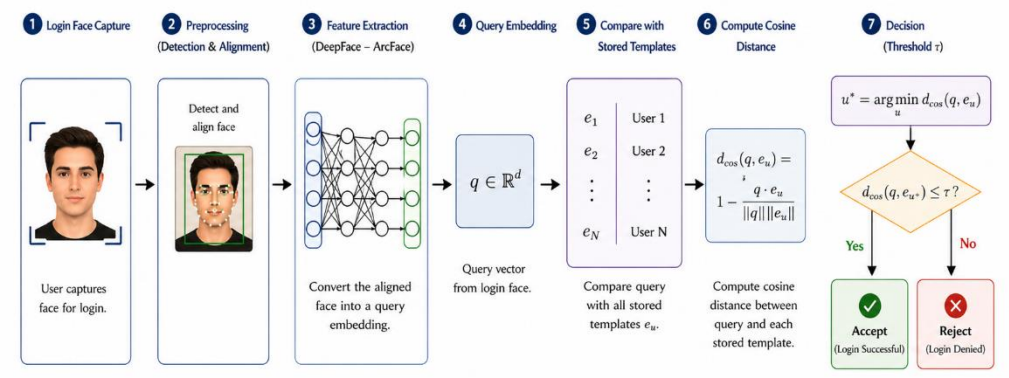


Figure 2.10: User Face Login Workflow

The threshold affects both security and usability. A strict threshold may reject genuine users, while a loose threshold may accept an impostor. This trade-off is usually described using false acceptance rate and false rejection rate:

$$FAR = \frac{FP}{FP + TN}, \quad FRR = \frac{FN}{FN + TP} \tag{2.16}$$

where

FAR = percentage of real users wrongly rejected

FRR = percentage of unauthorized users wrongly accepted

FP = impostor incorrectly accepted

TN = impostor correctly rejected

FN = genuine user incorrectly rejected

TP = genuine user incorrectly accepted

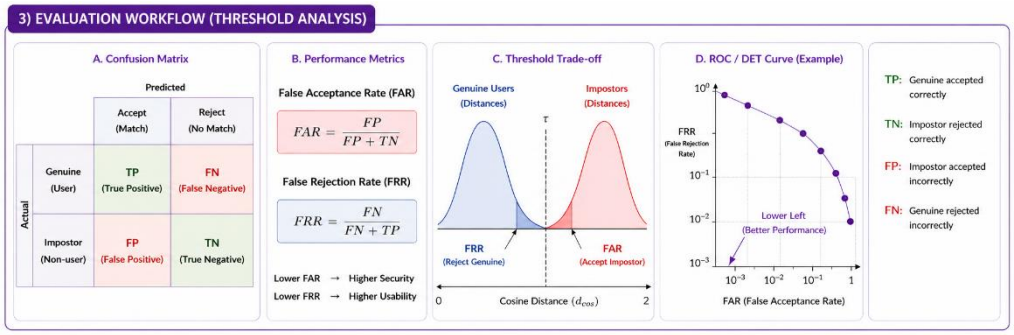


Figure 2.11: The Evaluation Workflow

2.9 Evaluation of RAG and Question Answering Systems

Evaluation of a Retrieval-Augmented Generation (RAG) question answering system should check both retrieval and generation. Retrieval evaluation measures whether the system can find the correct supporting evidence, while Retrieval-Augmented Generation Assessment (RAGAS) evaluation measures whether the generated answer is correct, relevant and grounded in the retrieved context. This separation is important because a poor answer may be caused by missing evidence, weak ranking, or unsupported generation (Es et al., 2023).

In this project, the retrieval of the system is evaluated using NDCG@k, Recall@k, MRR and Hit@k. However,the RAGAS evaluation will be using answer correctness, answer relevancy, faithfulness, context precision and context recall.

2.9.1 Retrieval Evaluation Metrics

Retrieval evaluation checks whether the system returns the correct source chunks within the top-ranked results. This is important for financial 10-K reports because the answer may depend on a specific table value, year, section, or statement. If the correct evidence is not retrieved, the language model may generate an answer based on incomplete context.

First of all, Hit@k determines if the top-k retrieved items include at least one relevant result. It indicates if the algorithm is successful in identifying any right answer among the first k outcomes.

$$\text{Hit@}k = \frac{1}{N}\sum_{i=1}^{N} 1\left(\left|R_i^k \cap G_i\right| > 0\right) \tag{2.17}$$

where

N = total number of evaluation questions

R_i^k = top-k retrieved chunks for question i

G_i = set of relevant gold evidence for question i

$1(\cdot)$ = indicator function, equa to 1 if the condition is true

Secondly, Recall@k determines the percentage of all relevant items that are found in the top-k results. It indicates how thoroughly the system records all relevant evidence.

$$\text{Recall@}k = \frac{\left|R_i^k \cap G_i\right|}{|G_i|} \tag{2.18}$$

where

Recall@k = propotion of gold evidence retrieved within the top-k results

$\left|R_i^k \cap G_i\right|$ = number of retrieved chunks that match the gold evidence

G_i = total number of gold evidenceitems for question i

Thirdly, Mean Reciprocal Rank (MRR) evaluates how early the first relevant result appears by taking the inverse of its rank. Higher values results that the correct answers appear closer to the top.

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i} \tag{2.19}$$

where

N = total number of queries in the evaluation set

rank_i = rank position of the first relevant retrieved chunk for question i

Lastly, NDCG@k evaluates the overall quality of the ranking by rewarding items that are highly relevant and appear higher in the list. A score between 0 and 1 is obtained by comparing the actual ranking to an ideal rating.

$$NDCG@k = \frac{1}{N}\sum_{i=1}^{N}\frac{DCG_i@k}{IDCG_i@k} \tag{2.20}$$

$$DCG_i@k = \sum_{j=1}^{k}\frac{2^{g_{ij}}-1}{\log_2(j+1)} \tag{2.21}$$

where

$NDCG@k$ = Normalised Discounted Cumulative Gain at top-k

$DCG_i@k$ = discounted gain of the retrieved ranking

$IDCG_i@k$ = ideal discounted gain for the best possible ranking

g_{ij} = relevance score of result j for question i

2.9.2 Anwer Generation Metrics

Answer generation evaluation checks whether the final answer is correct and relevant to the user question. In financial question answering, correctness is not only about fluent wording. The answer must also preserve the correct figure, unit, reporting year and comparison direction.

RAGAS uses answer_correctness to compare the generated answer with the reference answer. A simplified form can be written as:

$$AC_i = \lambda F_i^{\text{ans}} + (1 - \lambda) S_i^{\text{ans}} \tag{2.22}$$

where

AC_i = answer correctness score for question i

λ = weighting factor between factual and semantic companents

F_i^{ans} = factual agreement score between generated and reference answer

S_i^{ans} = semantic similarity score between generated and reference answer

RAGAS also uses answer_relevency to check whether answer directly addresses the question. The formula can be written as:

$$AR_i = \frac{1}{m} \sum_{j=1}^{m} \cos\left(E(q_i), E\left(\hat{q}_{ij}\right)\right) \tag{2.23}$$

where

AR_i = answer relevancy score for question i

m = number of generated evaluator questions

q_i = original use question

$\hat{q}_{ij}$ = question inferred from the generated answer

$E(\cdot)$ = embedding function

2.9.3 Faithfulness and Context Relevance

Faithfulness determines the extent to which the response is solidly supported by the retrieved context. This is different from answer correctness. An answer may match the reference answer but still be unfaithful if the retrieved sources do not justify the claim.

$$Faithfulness_i = \frac{|\mathrm{V_i}|}{|\mathrm{S_i}|} \tag{2.24}$$

where

$Faithfulness_i$ = faithfulness score for question i

S_i = set of statements extracted from the generated answer

V_i = subset of statements supported by the retrieved context

Context-based evaluation checks whether the retrieved context is useful and complete. In this project, this is measured using context_precision and context_recall.

$$ContextPrecision_i@K = \frac{\sum_{j=1}^{K} P_i@j \cdot v_{ij}}{\sum_{j=1}^{K} v_{ij}} \tag{2.25}$$

$$P_i@j = \frac{\sum_{t=1}^{j} v_{it}}{j} \tag{2.26}$$

where

$ContextPrecision_i@K$ = context precision for question i at top-k

$P_i@j$ = precision at rank j

v_{ij} = relevance lable of retrieved context j

K = number of retrieved contexts considered

$$ContextRecall_i = \frac{\left|C_i^{\text{ref}}\right|}{\left|T_i^{\text{ref}}\right|} \quad (2.27)$$

where

$ContextRecall_i$ = context recall score for question i

C_i^{ref} = required information units covered by the retrieved context

T_i^{ref} = information units required by the reference answer

2.9.4 RAGAS Evaluation Framework

Retrieval-Augmented Generation Assessment (RAGAS) is used in this project as a second evaluation layer for the retrieval-augmented generation pipeline. The retrieval evaluation measures whether the system can find and rank useful evidence through hit@k, recall@k, MRR and NDCG@k, but these scores do not show whether the final answer is correct or properly supported. RAGAS addresses this gap by evaluating the generated response together with the retrieved contexts, using faithfulness, context recall, answer correctness, context precision and answer relevancy. This makes the evaluation more diagnostic because it separates retrieval quality from answer quality instead of treating the whole pipeline as a single score (Es et al., 2023)

For financial 10-K question answering, this separation is important because a response may sound fluent while still using incomplete evidence, the wrong financial figure or unsupported reasoning. answer_correctness checks whether the response matches the expected answer, answer_relevancy checks whether it directly addresses the question, and faithfulness checks whether the claims are grounded in the retrieved context. Context precision and context recall then assess whether the retrieved evidence is relevant and sufficient for answer generation. Therefore, RAGAS is presented here as the selected framework for evaluating answer quality and grounding, while the actual RAGAS scores should only be discussed after the standard RAGAS run has been completed and verified.

2.9.5 Summary

The literature reviewed in this chapter shows that financial 10-K report analysis requires more than basic document search or standalone language model generation. Financial reports contain long narrative sections, tables, accounting terms and document-specific figures, which makes evidence extraction difficult when the system depends only on keyword matching. The review of large language models and RAG shows that LLMs can support natural language question answering, but their answers need to be grounded in retrieved report content. Vector embeddings, vector databases, ranking and reranking therefore form the retrieval foundation for matching user questions with relevant financial evidence.

Graph RAG and multi-agent LLM systems extend this foundation by addressing the relationship-based and task-specific nature of financial questions. A graph structure can represent links between companies, report sections, financial concepts and numerical evidence, while specialised agents can divide the workflow into query understanding, retrieval, reasoning and answer synthesis. This is useful for factual, comparative and analytical questions because different query types may require different evidence paths. The literature therefore supports the use of a Multi-Agent Graph RAG design rather than a single general retrieval pipeline.

Facial recognition authentication adds an access-control component to the system, with DeepFace supporting face detection, embedding generation and face matching for registration and login. At the same time, the evaluation literature shows that the system should be assessed through both retrieval quality and answer quality, using retrieval metrics together with RAGAS-based answer and context metrics. Overall, the reviewed work provides the basis for developing a secure Multi-Agent Graph RAG system that combines semantic retrieval, graph-based reasoning, agent specialisation and structured evaluation for financial 10-K question answering.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The chapter study concept for this project focuses on design, develop and evaluate the secure Multi-Agent Graph Retrieval-Augmented Generation (RAG) system for financial Form 10-K report analysis. Chapter 2 reviewed the background of financial reports, Large Language Model (LLM), retrieval-augmented generation, graph-based retrieval, multi-agent systems, facial authentication and evaluation methods. This chapter moves from that literature background to the actual project methodology. The purpose is to explain how the proposed system processes financial documents, builds a retrievable knowledge base, protects user access and generates evidence-grounded answers through a chatbot interface.

The system is designed for report-based financial question answering, not live market prediction or investment recommendation. Uploaded 10-K PDF reports are processed into structured records, text chunks, table-aware content, entities and relationships. These outputs are stored across MongoDB, Milvus and Neo4j so that the system can retrieve evidence through structured lookup, vector similarity search, graph traversal and calculation support. This design is important because financial questions often depend on exact values, fiscal years, units, table context and the meaning of the original source section. The query workflow therefore uses specialist retrieval agents to support factual, comparative and analytical questions, while the final answer is prepared with supporting sources so that users can trace the response back to the report evidence.

In the nutshell, this chapter also includes the secure user workflow because access control is part of the system design. Facial registration and login are used to protect document upload, chat access and user-owned conversation

records. The methodology therefore covers both the intelligence layer of the system and the application layer that users interact with.

3.2 System Components and Main Purpose

3.2.1 React/Vite Frontend

React is a popular JavaScript package used to develop user interfaces, particularly for web applications. Vite is a frontend build tool that provides a fast development server and production build workflow for modern web applications (Meta Open Source, n.d.; Vite contributors, n.d.).

React and Vite are used to build the user-facing interface. The frontend supports facial login, document upload, chatbot interaction, source display and conversation history. It sends user requests to the backend and presents the generated answer with supporting evidence.

3.2.2 FastAPI Backend

FastAPI is a Python framework for building web APIs using Python type hints. It supports API routing, request validation and automatic API documentation, which makes it suitable for backend service development (FastAPI, n.d.).

FastAPI acts as the main backend service layer. It receives requests from the frontend, validates user sessions, handles document and query routes, and connects the frontend to the retrieval runtime and storage services.

3.2.3 OpenAI Agents SDK Runtime

The OpenAI Agents SDK Runtime is a framework designed for the development and execution of autonomous AI agents capable of performing tasks by integrating language models with tools, memory, and workflows (OpenAI, n.d.). It serves as the execution environment whereby agents analyse user inputs, choose the appropriate tools or data sources, and formulate responses. This renders it advantageous for intricate systems such as your RAG pipeline, because the agent can orchestrate retrieval, reasoning, and response creation in a systematic and scalable manner.

The OpenAI Agents SDK coordinates the Multi-Agent Graph RAG workflow. It analyses user queries, selects suitable retrieval components, gathers evidence from different storage layers, and prepares the retrieved context for answer generation.

3.2.4 MongoDB

MongoDB is a NoSQL database that stores data in JSON-like formats rather than rigid tables (MongoDB, Inc., n.d.). This enables developers to manage unstructured or dynamic data effortlessly without predetermined schemas. It is exceptionally scalable, facilitates rapid read and write operations, and is frequently employed in contemporary applications such as online systems, APIs, and AI-driven platforms.

Within this system, MongoDB stores document metadata, processed chunks, extracted tables, configuration records, user records, and conversation records. It facilitates organised retrieval and management of application state.

3.2.5 Milvus Database

Milvus is an open-source vector database intended for the storage and retrieval of high-dimensional embeddings utilised in artificial intelligence applications. It enables quick similarity searches, making it appropriate for applications such as semantic query, recommendation algorithms, and RAG (Wang et al., 2021). Milvus facilitates large-scale data management, GPU acceleration, and a distributed architecture, enabling the efficient processing of millions or billions of vectors in contemporary AI systems.

Milvus saves embeddings generated from 10-K report entities and chunks in this system. Even when the user's language differs from the original report text, it enables the retrieval engine to locate semantically significant information.

3.2.6 Neo4j Database

Neo4j is a graph database that stores data as nodes, relationships and properties. This structure is suitable for representing connected entities and relationship-based queries (Neo4j, Inc., n.d.). The inherent graph structure facilitates quick navigation and querying of intricate relationships, proving especially advantageous for activities related to entity linking, recommendation systems, and semantic reasoning.

Neo4j act as the storage which stores financial entities and relationships extracted from 10-K reports. It supports graph traversal when a question depends on connections between companies, fiscal years, financial metrics, sections or reported values.

3.2.7 Deepface Authentication

DeepFace is an open-source facial recognition system that utilises deep learning models for face detection, verification, and recognition with high precision (Serengil and Ozpinar, 2020). It incorporates several advanced algorithms, including VGG-Face, FaceNet, and ArcFace, facilitating strong performance under varying lighting circumstances and facial differences. DeepFace is extensively utilised in applications including biometric authentication and user identity verification.

In this system, DeepFace supports facial registration and login. It helps verify user identity before allowing access to document upload, chatbot functions and owner-scoped conversation records.

3.2.8 RAGAS Evaluation Framework

RAGAS (Retrieval-Augmented Generation Assessment) is an evaluation paradigm for measuring the performance of RAGAS systems by examining faithfulness, answer relevance, and context precision, without needing ground-truth labelling (Shasul Es et al., 2023). It is particularly beneficial for enhancing the reliability and transparency of AI systems that depend on external knowledge sources, as it allows for a systematic assessment of the extent to which retrieved documents validate generated responses.

The evaluation runners are used to test retrieval quality and answer quality using prepared datasets. They are separate from the live chatbot workflow and are used to produce evaluation artifacts for Chapter 4.

3.3 Document Ingestion and Knowledge Base Construction

This section describes how the system constructs a searchable knowledge base from uploaded financial PDF documents. Source documents are parsed, cleaned, chunked, enriched, embedded, and saved across the native storage backends as part of the workflow that follows the ingestion side of the system pipeline.

When a user uploads a 10-K PDF via the backend document endpoint in the current system, the ingestion process begins. PDF files are accepted by the backend, which then temporarily stores the upload, computes a file hash for duplicate detection, generates a pending document record, and does the entire processing in the background. As a result, the application would continue to operate while the more involved parsing and storing phases are finished.

3.3.1 Upload and Background Ingestion Worfklow

3.3.1.1 PDF Upload and Validation

The ingestion process begins when a user uploads a financial report through the system interface. The current system accepts PDF files as the document input because 10-K reports are commonly distributed in this format and contain both narrative sections and financial tables. After upload, the backend validates the file type and stores the PDF temporarily for processing.

3.3.1.2 Duplicate Detection and Task Creation

Before the full ingestion process starts, the backend computes a file hash for the uploaded document. This hash is used to detect duplicate uploads and reduce repeated processing of the same report. If the file is accepted, the system creates a pending document record and schedules the remaining ingestion steps as a background task. This allows the upload workflow to remain responsive while PDF parsing, extraction, embedding, and storage run separately.

3.3.2 Document Parsing and Metadata Preparation

3.3.2.1 Table-Aware PDF Parsing

The uploaded PDF is processed using the DoclingProcessor, which converts the report from pdf format into structured Markdown while preserving document layout where possible. Table-aware parsing is enabled because many financial answers depend on table values, row labels, column headings, units, and fiscal years. This prevents the system from treating the 10-K report as plain text only.

3.3.2.2 Document Normalisation and Metadata Extraction

After parsing the pdf file, the DocumentAnalyzer organises the extracted content into structured elements such as headings, text blocks, and tables. Table content may retain rows, columns, headers, captions, and nearby context where available. The MetadataExtractor then identifies important document-level information such as company name, ticker, CIK, fiscal year, page count, and table count. These metadata fields help the system keep each chunk and answer linked to the correct source report.

3.3.3 Chunk Construction and Entity-Relationship Extraction

3.3.3.1 Structured Chunk Generation

After document analysis and metadata extraction, the system uses the Structured Chunker to convert the analysed 10-K content into retrieval-ready chunks. Instead of splitting the report only by page number or fixed text length, the chunker processes the document section by section and groups ordered elements such as paragraphs, headings, and tables according to the detected report structure. During this process, each chunk preserves useful context, including section type, section title, subsection title, chunk index, token count, and position within the processed section.

For table-related chunks, the system also keeps available table metadata such as table format, row and column information, headers, records, captions, and nearby context. The original chunk content may still contain Markdown-style table text for traceability, while the retrieval_text field can provide a cleaner representation when table linearisation is possible. This allows the system to keep source content linked to the original report while improving how table-heavy financial information is searched and embedded.

3.3.3.2 Financial Entity and Relationship Extraction

After structured chunks are generated, the system extracts important financial and business entities from the selected chunks. These entities include company names, financial metrics, fiscal years, business segments, products, locations, risk factors, regulations, and executives. The extraction process uses the section context preserved during chunking, so the system can interpret information differently depending on whether it appears in Business, Risk Factors, Management Discussion, or Financial Statements.

The system also identifies relationships between the extracted entities. For example, a company may be linked to a reported metric, a business segment, a risk factor, or a reporting period. These relationships are important because they allow the knowledge base to represent the 10-K report as connected information rather than isolated text passages. This supports later graph-based

retrieval, where the system can search for related entities and evidence across different sections of the same report.

3.4 Multi-Store Knowledge Base Construction

3.4.1 MongoDB Document and Metadata Storage

MongoDB stores the main structured records created during ingestion. These include document records, chunk records, extracted entities, relationships, processing jobs, and normalised table rows. In this system, MongoDB acts as the main record layer that keeps the processed document content and metadata available for later retrieval, citation, and document management.

3.4.2 Milvus Vector Indexing

The system generates embeddings for both document chunks and extracted entities. Chunk embeddings are created from the retrieval text when available, otherwise from the original chunk content. Entity embeddings are generated from the entity name, type, and description. These embeddings are stored in Milvus as chunk and entity vector indexes, which support semantic retrieval during user question answering.

3.4.3 Neo4j Knowledge Graph Storage

Neo4j stores the graph representation of the processed report. The graph contains document nodes, entity nodes, and typed relationships between entities. As a result, the retrieval pipeline can now search not just for terms that are semantically related, but also for relationships among metrics, segments, risks, and organisations in the financial sector.

3.4.4 Graph Maintenance and Cosistency Verification

3.4.4.1 Graph Maintenance

After the main storage process, the system performs graph maintenance to improve the quality of the knowledge graph. This stage helps remove weak or invalid relationships, handles duplicated entities where possible, and supports cross-chunk relationship discovery. As a result, the graph becomes more useful for relationship-based and multi-hop retrieval.

3.4.4.2 Cross-Store Consistency Checking

The final stage checks whether the processed document remains aligned across MongoDB, Milvus, and Neo4j. This is important because the system depends on multiple storage layers: MongoDB stores structured records, Milvus stores vector indexes, and Neo4j stores graph relationships. Once the checks are completed, the document status can be marked as completed and the processed report becomes available for the downstream Multi-Agent Graph RAG query workflow.

3.5 Multi-Agent Graph RAG Query Processing

This section explains how the system processes a user question after the financial 10-K knowledge base has been constructed. The query workflow starts from the chatbot request and ends with a grounded answer supported by retrieved sources. In the current system, the FastAPI backend and OpenAI Agents SDK runtime coordinate query analysis, specialist retrieval, evidence reranking, context construction, calculation support, and final answer synthesis.

3.5.1 Query Analysis and Scope Extraction

The query processing workflow begins with deterministic query analysis before ReAct-based routing is applied. In this stage, the system normalises the user query and extracts structured signals such as company names, fiscal years, financial metrics, relationship types, requested retrieval mode, and query intent. This converts a free-form financial question into a more stable query representation that can be used by the downstream retrieval pipeline.

This step also identifies the likely evidence sources required to answer the query, such as semantic document chunks, structured table rows, graph relationships, or calculated values. For example, a factual query may mainly require document chunks or table records, while an analytical query may require structured values followed by calculation. This step's output does not constitute the final solution. Instead, it specifies the retrieval scope to ensure that the right sources are retrieved by the ReAct router.

3.5.2 ReAct Evidence and Tool Routing

After query analysis, the system applies a ReAct-style routing approach to guide evidence collection. ReAct combines reasoning with tool actions, allowing a language model to decide which external tool should be used next based on the current task and observations (Yao et al., 2023). In this system, ReAct is used only as an evidence and tool router. It does not generate the final user-facing answer.

The ReAct router uses the extracted query scope to decide whether to call Milvus, MongoDB, Neo4j, the Calculator, or a combination of these tools. Milvus is used for semantic retrieval from document chunks, MongoDB is used for structured financial rows and metadata, Neo4j is used for graph-based entity and relationship evidence, and the Calculator is used for derived numerical values. The router continues collecting evidence until the required sources have been covered or sufficient evidence has been gathered.

3.5.3 Specialist Retrieval Tools and Evidence Bundle Construction

After the ReAct router identifies the required evidence sources, the system calls the relevant specialist tools. Milvus retrieves semantically related document chunks from embedded 10-K filings, while MongoDB retrieves structured financial rows, metadata, and table-based values. Neo4j is used for graph-based evidence, such as relationships between companies, filing sections, entities, and financial concepts.

The Calculator is used when the query requires derived numerical answers, such as percentage changes, ratios, or comparisons. However, calculations are only performed after the required supporting values have been retrieved, ensuring that numerical outputs are based on collected evidence rather than unsupported assumptions.

The retrieval process uses a fixed baseline configuration across factual, comparative, and analytical queries. The requested mode affects tool selection and routing, but not separate retrieval budgets.

Table 3.1: Retrival Baseline Configuration

Parameter	Value	Purpose
top_k_chunks	30	Semantic document chunks
top_k_entities	30	Entity candidates
top_k_relations	30	Graph Relationships
top_k_tables	30	Structured table rows
graph_depth	3	Graph traversal depth
rerank_top_n	30	Reranking candidate pool
final_top_n	10	Final evidence for synthesis

The retrieved chunks, structured rows, graph relationships, and calculated outputs are then normalised into an evidence bundle. This shared format allows different evidence types to be passed consistently into reranking and fusion before final answer synthesis.

3.5.4 Reranking, Fusion and Answer Synthesis

After the evidence bundle is constructed, the system applies reranking to prioritise the most useful evidence. Evidence from different sources is not treated equally by default, because a semantically similar text chunk may not contain the exact financial value, while a structured row may require narrative context to be interpreted correctly. The reranking stage therefore helps select evidence that is more relevant, specific, and useful for answering the analysed query.

The selected evidence is then fused into a consolidated context for answer generation. This fused context may include narrative passages, structured financial values, graph relationships, and calculated results. The purpose of this stage is to organise the evidence into a form that the synthesis component can use without needing to repeat retrieval decisions.

The final answer is produced by the Synthesis Agent, not by the ReAct router. The Synthesis Agent uses the fused evidence to generate a grounded response with supporting sources, while preserving important financial details

such as values, units, fiscal years, comparison direction, and citations. The final response also maintains fields such as answer, sources, query analysis, and answer payload where available, so that the frontend and evaluation workflow can inspect both the answer and the evidence behind it.

3.6 Secure Frontend and User Workflow

This section describes the system's user-facing procedure and how access to the Financial RAG application is secured by facial authentication. This part focuses on how users access the system, handle documents, use the chat interface, and maintain authenticated sessions, whereas Sections 3.4 and 3.5 explain document processing and question answering.

3.6.1 Frontend User Workflow

The primary layer of communication between the user and the Financial RAG process is provided by the frontend. Before granting access to the secured workspace, the program initially takes the user to the login page when facial authentication is enabled. The user can upload financial documents, review processed documents, and ask natural-language queries via the chat interface once a valid login session has been initiated.

Every query is sent via the chat interface to the backend retrieval and synthesis process outlined in Section 3.5. To enable the user to follow the response back to the evidence that was obtained, the produced response is shown together with supporting sources. Additionally, conversation logs are connected to the verified user, enabling users to revisit earlier exchanges while maintaining the privacy of each user's past.

3.6.2 Face Registration Workflow

The identification profile needed for a subsequent login is created throughout the face registration process. The user inputs a name and uses the frontend camera interface to take several face samples during registration. Before facial embeddings are created, these samples are transferred to the backend, where face detection is performed.

In order to simplify future authentication, the created face template is saved alongside the user profile. The system establishes a session for the newly registered user to access the protected application when registration is successful. This keeps the process of setting up an identity apart from the regular uploading of documents and responding of questions.

3.6.3 Face Login and Session Workflow

Before allowing access to protected pages, the facial login process confirms that the user already exists. The backend pulls an embedding from the provided image when the user takes a picture of their face from the login page. The chosen matching threshold is used to compare this embedding with saved facial profiles.

The backend recognises the user and creates a session cookie if the provided face matches a profile that has been registered. This session state is then used by the frontend to grant access to secure routes like document upload, document viewing, chat, and conversation history. The user stays in the authentication flow and is unable to enter the main workspace if authentication is unsuccessful or there is no active session.

3.7 Gantt Chart

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17
1.	1. Go through and familiar the library project counting using face detection system codes. 2. Deploy the LLM Facial Memory System and make sure it can run properly on the pc.	2025-07-06																	
2.	1. Learn how to start with machine learning. 2. Create a school library dataset for machine learning.	2025-07-19																	
3.	- Build the full project structure - Update the database for books and FAQs	2025-08-02																	
4.	1. Implement Agentic RAG to the system	2025-08-15																	
5.	1. Build and design an user interface (UI) for the system 2. Test and optimize the performance of the Agentic Rag	2025-08-29																	
6.	1. Initial chatbot testing and obtaining preliminary results 2. Perform test and optimize the accurary and response time of the chatbot	2025-09-12																	
7.	Progress report and presentation slide preparation	2025-09-22																	

Figure 3.1: Gantt Chart for FYP Part 1

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17
1.	Design the Graph RAG system with facial recognition for financial reports	2026-02-13																	
2.	-Design and optimize the ingestion and retrieval pipeline of the system	2026-02-27																	
3.	Perform retrieval metric evaluation	2026-03-13																	
4.	Integrate facial recognition for the authentication	2026-03-27																	
5.	Perform RAGAS performance evaluation	2026-04-10																	
6.	Refactor the codebases	2026-04-24																	
7.	Prepare FYP report and presentation slides	2026-05-03																	

Figure 3.2: Gantt Chart for FYP Part 2

Based of Figure 3.1 and Figure 3.2 above, all the objectives that had planned are being carried out successfully during the semester.

3.8 Summary

This chapter outlines the design and methodology of the Multi-Agent Graph RAG system including facial recognition for the analysis of financial 10-K reports. The system has seven fundamental components: a React frontend, a FastAPI backend, the OpenAI Agents SDK for multi-agent coordination, MongoDB for structured records and metadata, Milvus for vector search, Neo4j for graph relationships, and DeepFace for face authentication. The RAGAS framework was chosen for the evaluation of answer quality.

Moreover, the ingestion process converts 10-K PDFs into a multi-store knowledge base via table-aware parsing, structured chunking, and the extraction of entities and relationships. The resultant chunks, embeddings, and graph nodes are preserved in MongoDB, Milvus, and Neo4j.

In the nutshell, the retrieval pipeline employs a deterministic analyser to extract companies, metrics, and purpose from every question. A ReAct evidence router orchestrates four specialist agents to gather evidence, which is subsequently reranked, fused, and sent to the synthesis agent for response formulation. User access is regulated via facial registration and authentication, with session management safeguarding the frontend process from upload to chat. The Gantt chart verified that all intended objectives were executed within the project schedule.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

Chapter 3 outlined the construction of the system, detailing the ingestion pipeline that converts 10-K reports into a multi-store knowledge base, the face authentication procedure, and the ReAct-driven multi-agent retrieval pipeline that addresses user enquiries. This chapter evaluates the extent to which the constructed system fulfils the objectives defined in Chapter 1.

The chapter shows that each part of the system works, namely the front-end with face authentication, the document ingestion pipeline, the multi-store knowledge base and the end-to-end query processing workflow. Figure 4.1 illustrates the entire system workflow, from document upload to answer generation. The subsequent section of the chapter assesses the system's performance through retrieval and answer quality metrics, concluding with a discussion of the overall findings.

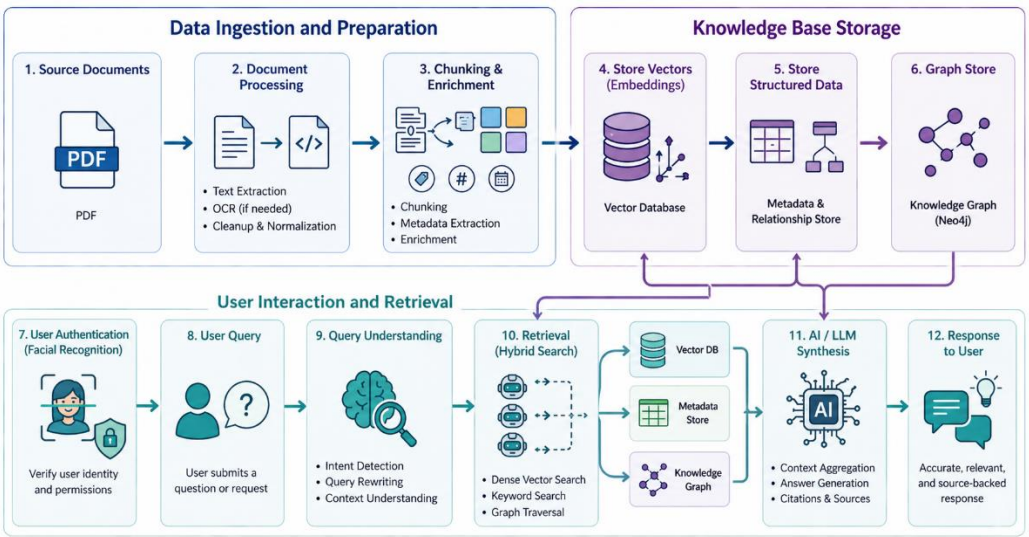


Figure 4.1: Complete System Architecture

4.2 Frontend Implementation

4.2.1 Facial Authentication Page

The authentication page is the initial interface displayed to a user when facial recognition is activated. It facilitates two processes: registration and authentication. Upon registration, the user inputs a name and captures a minimum of three facial photos via the camera. A camera preview component shows the live feed during capture, while the website presents status feedback, including a loading indicator during face processing and a success or error message at completion of the procedure. Figure 4.2 shows the registration page of a new user.

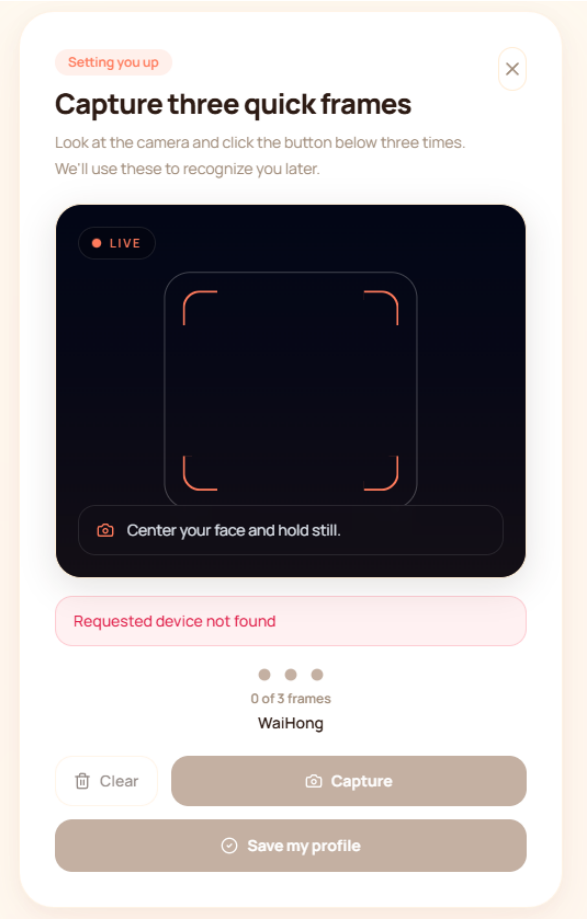


Figure 4.2: New User Registration Page

Upon login, the user is required to capture a singular facial image, which is then compared to previously stored embeddings for identity verification. If face authentication is deactivated in the frontend settings, this page is fully bypassed, and the user is directed straight to the chat interface. Figure 4.3 shows the login page after user had complete the registration.

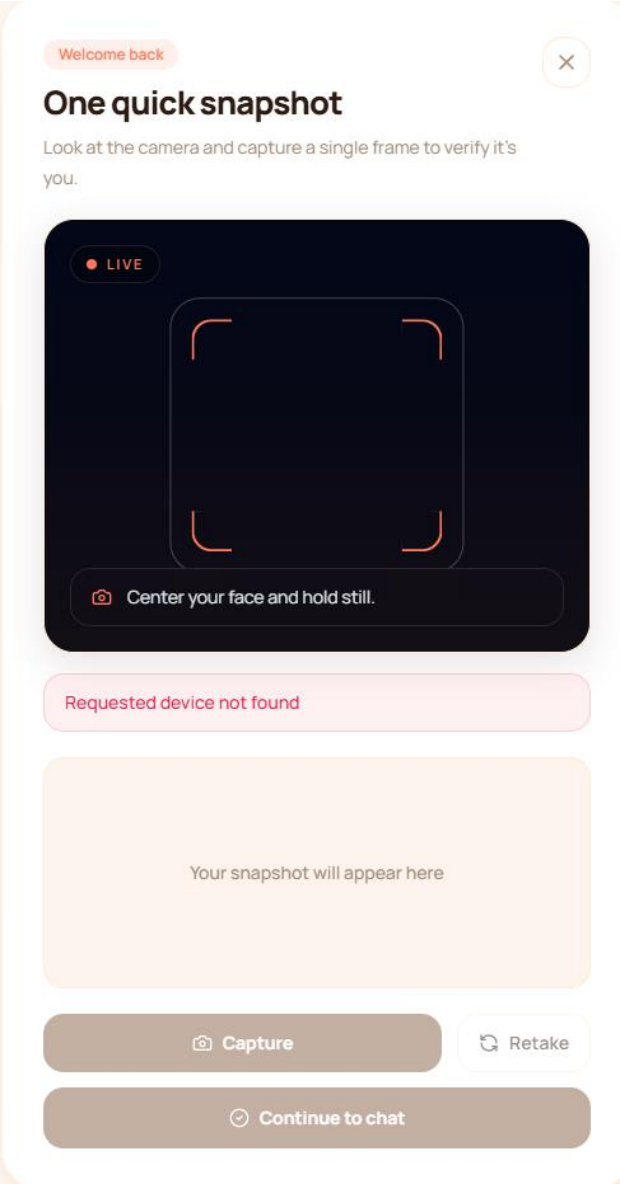


Figure 4.3: User Login Phase

4.2.2 Frontend Web UI

The chat interface consists of a left sidebar and a central content section. The sidebar includes the application name, a button to initiate a new chat, navigation links to more pages, and a scrollable list of prior conversations identified by title and timestamp. Choosing any conversation retrieves its complete message history into the primary area.

The primary section includes a text input for the user's inquiry and the answer presentation. The active retrieval stage during query processing is indicated by a streaming indicator, which may include the analysis of the question, the search of document chunks, the querying of financial tables, or the traversal of the knowledge graph. The conclusive response is presented in a message bubble accompanied by numerical citations that reference specific material inside the original 10-K reports. Users may access a citation panel to examine the retrieved source, a comparison panel to observe metric values across companies side-by-side, and a calculation panel to view derived values along with the source evidence from which they were generated.

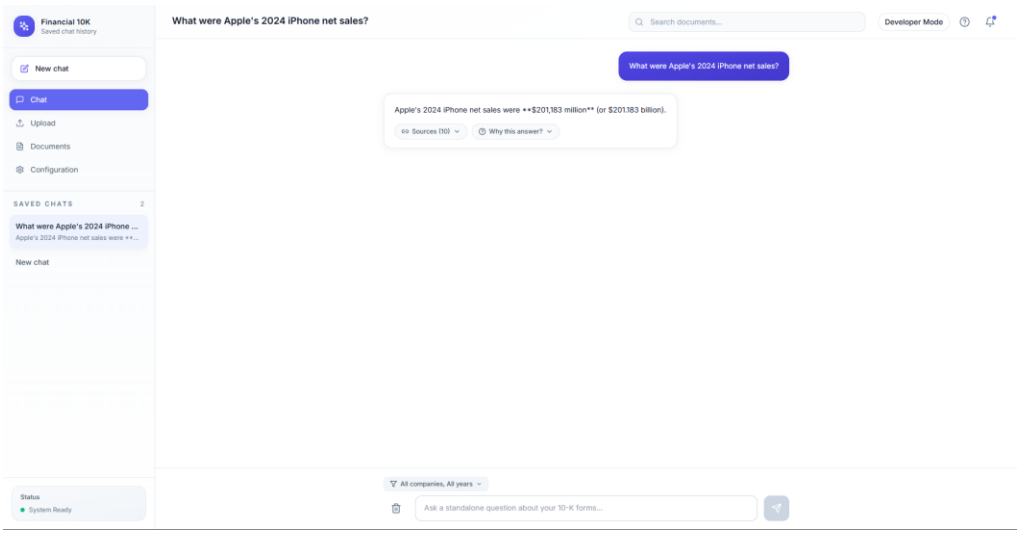


Figure 4.4: Frontend Chat Interface

The upload page enables users to submit 10-K reports to the system via drag-and-drop or file selection. Only PDF files are permitted. Upon upload, a progress indicator displays the processing status as the system parses, segments, extracts, and indexes the document in the background. The documents page displays a table containing all uploaded reports, including the company name, fiscal year, processing status, and upload date. The setup page presents the existing retrieval settings, including fixed baseline parameters such as number of chunks, entities, relationships, and tables retrieved per query. This page is non-editable and displays the runtime configuration utilised during assessment.

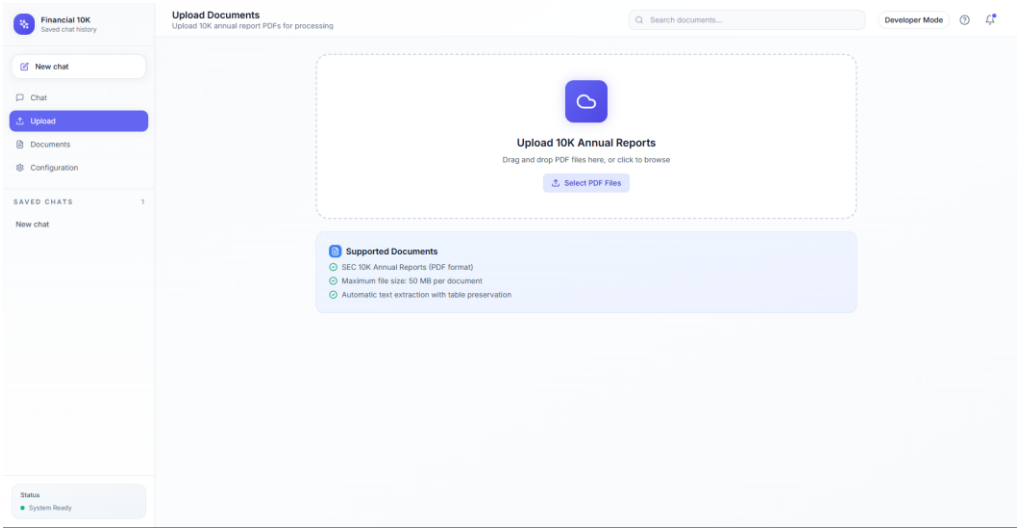


Figure 4.5: Frontend Upload Interface

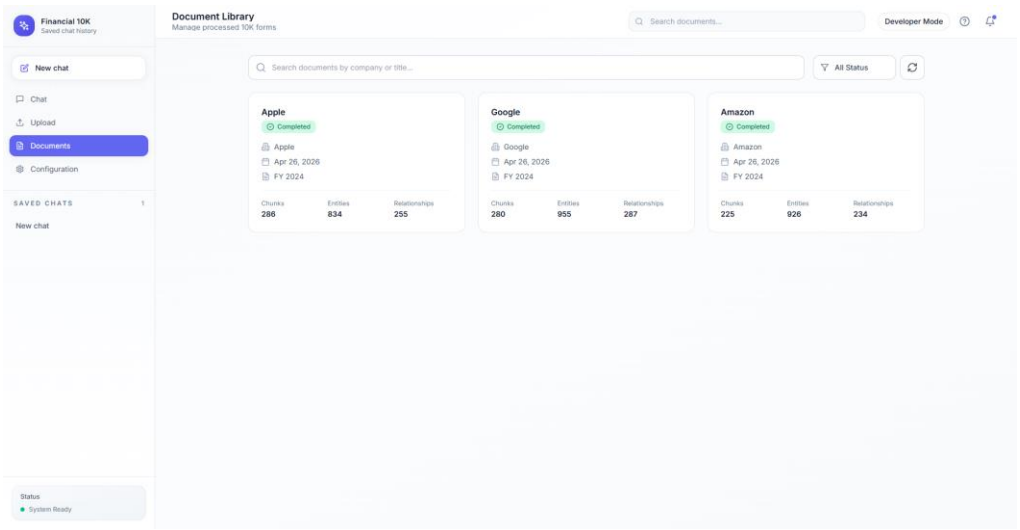


Figure 4.6: Frontend Documents Library Interface

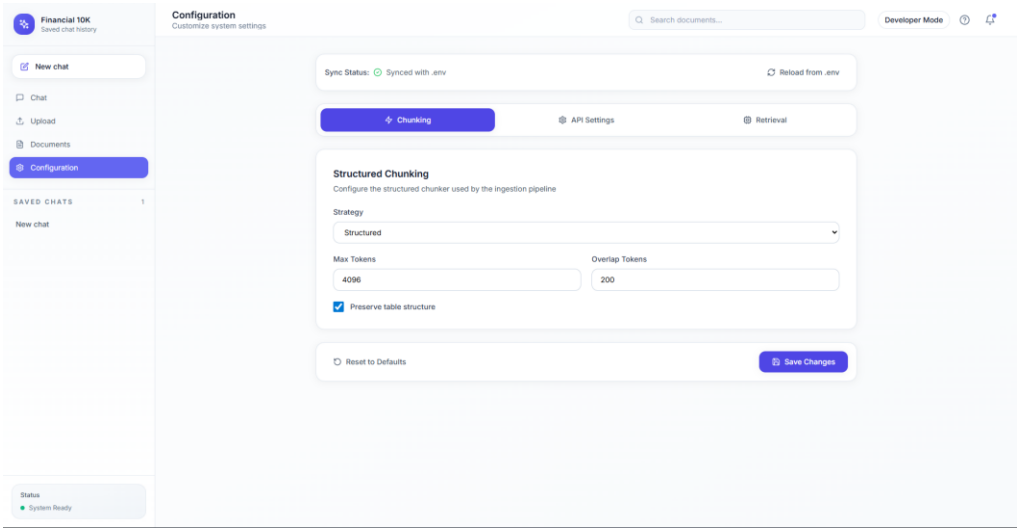


Figure 4.7: Frontend Configuration Interface

4.3 Facial Recognition verification

The facial recognition system was set up as the access control mechanism for the application. This section highlights the system's performance during the registration and login testing phases.

Upon registration, a user inputs a name and captures a minimum of three facial photographs using the webcam. Each image is sent to the backend, where the DeepFace library utilises the ArcFace model in conjunction with a RetinaFace detector to extract a 512-dimensional embedding vector. The embeddings from all collected samples are averaged into a singular representation and stored in the database with the user's name. The system rejects the registration if fewer than three photos are uploaded.

Upon login, the user captures a singular facial image. The identical embedding extraction procedure is executed, and the resultant vector is evaluated against all archived user embeddings utilising cosine similarity. If the maximum similarity score above a threshold of 0.35, the system authorises the login and creates a session with a twelve-hour expiration, recorded as a browser cookie. If no stored embedding yields a score exceeding the threshold, the login is denied.

The system accurately recognised registered users during various login attempts in testing. Unfamiliar faces were routinely dismissed. Sessions lasted through page refreshes, and users remained authentication until session expiration. Upon disabling facial authentication via the frontend setup, the system bypassed the authentication page and simply navigated to the chat interface, as designed.

4.4 Knowledge Base Construction Results

Three Financial 10-K annual reports from different companies were used to perform ingestion. The documents are Amazon 2024 10-K, Google 2024 10-K and Apple 2024 10-K.

4.4.1 Document Ingestion

Every PDF went through a number of processing phases. Docling preserved the table content, column titles, and section style while converting the report from PDF to structured Markdown. The Markdown was then categorised into structured pieces by the document analyser based on the following sections: Business, Risk Factors, Management Discussion and Analysis, and Financial Statements. At this point, metadata was extracted, including the firm name, ticker, CIK number, fiscal year, page count, and table count.

The structured chunker then grouped related paragraphs, headings, and tables into chunks by processing each report section by section. Each chunk contained its position, title, and section type. Row headers, column headers, and captions were also kept in table-heavy sections. After that, the entity extractor extracted financial and business entities, including companies, metrics, years, segments, products, risks, regulations, and executives, as well as the relationships between them, such as a company's reported revenue, a segment's contribution, or a metric value for a specific year.

4.4.2 Multi-Store Storage

Three databases were used to store the processed content. Document records, processed job records, extracted entities and relationships, structured chunks with section context, and normalised table rows are all stored in MongoDB. Milvus stores two sets of embeddings, which are chunk vectors from retrieval text and entity vectors from name, type and description. Both use 3,072-dimensional vectors in text-embedding-3-large. Figure 4.8 shows the real-time status dashboard of backend databases and ingested documents.

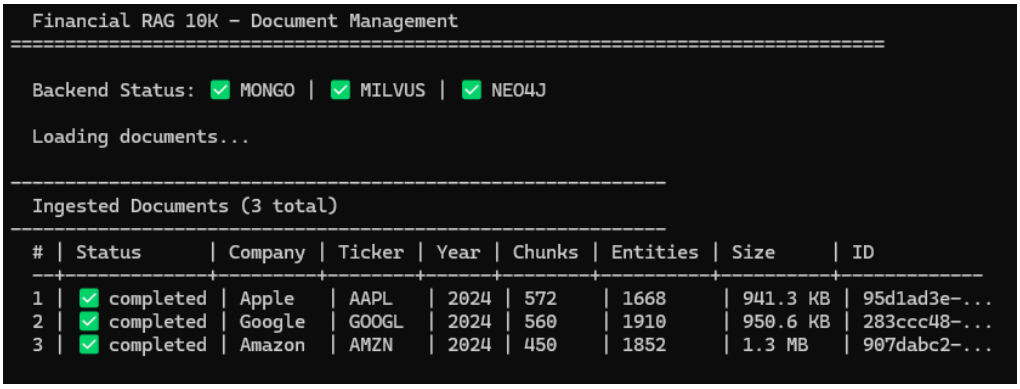


Figure 4.8: Backend Health and Document Ingestion Monitor

The knowledge graph is stored in Neo4j and consists of document nodes, entity nodes for businesses, metrics, years, segments, goods, risks, and executives, all linked by typed connections. During retrieval, the graph allows for a traversal depth of three hops. The knowledge graph as it appears in the Neo4j Browser is shown in Figure 4.9, where relationships are displayed as directed edges and nodes are colour-coded according to entity type. Before each document was declared as ready for retrieval, graph maintenance procedures eliminated weak or redundant relationships, fixed cross-chunk entity references, and confirmed consistency across the three stores.

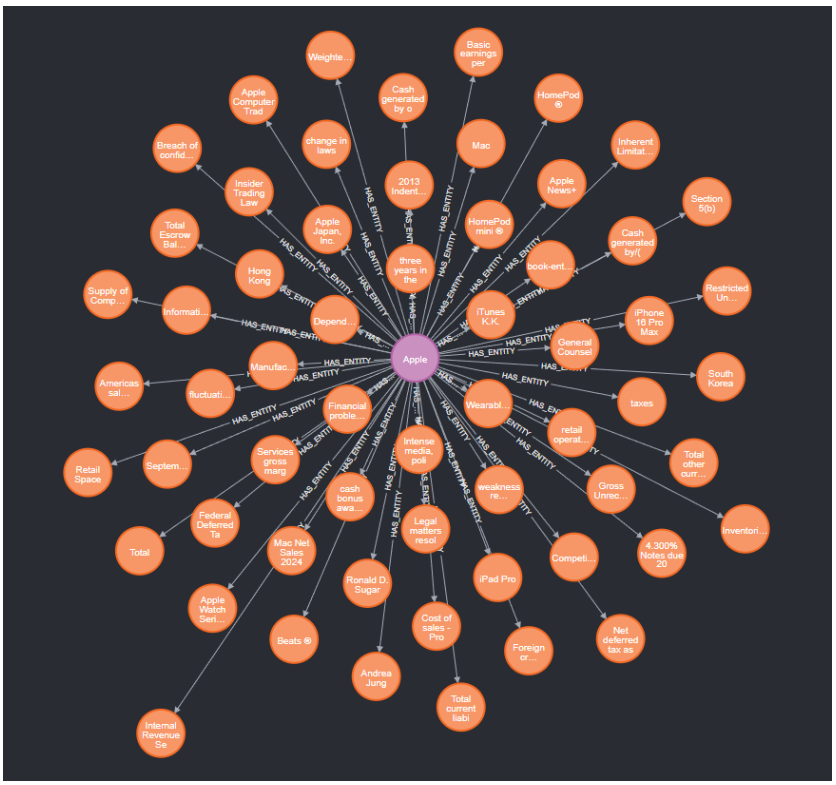


Figure 4.9: Knowledge Graph Display at Neo4j

4.5 Query Processing Demonstration

To verify that the retrieval pipeline works end-to-end, three example queries were run through the chat interface, one per retrieval mode. Each query produced a frontend answer with citations and a backend processing log showing the internal steps. The frontend and backend outputs are shown together so that both what the user sees and how the system arrived at it can be inspected.

4.5.1 Factual Query

A factual query asks for a specific value from a single company's 10-K report. The example query was "What was Apple's total revenue in fiscal year 2024?"

The backend log shows the system routed the query to two retrieval specialist agents. The first searched the financial tables and returned 18 relevant rows. The second searched document chunks by semantic similarity and returned 30 text passages from the Apple 2024 10-K, giving 48 evidence items total.

The reranker scored the 48 items and kept the 10 most relevant. The top ranked item was a table row containing the exact revenue figure of $391,035 million. The remaining nine provided supporting context from the report.

The synthesis agent used these sources to produce the answer: Apple reported total revenue of $391,035 million in fiscal year 2024, with a citation linking back to the table row. Processing took 9.6 seconds. Figure 4.10 shows the frontend answer and Figure 4.11 shows the backend log.

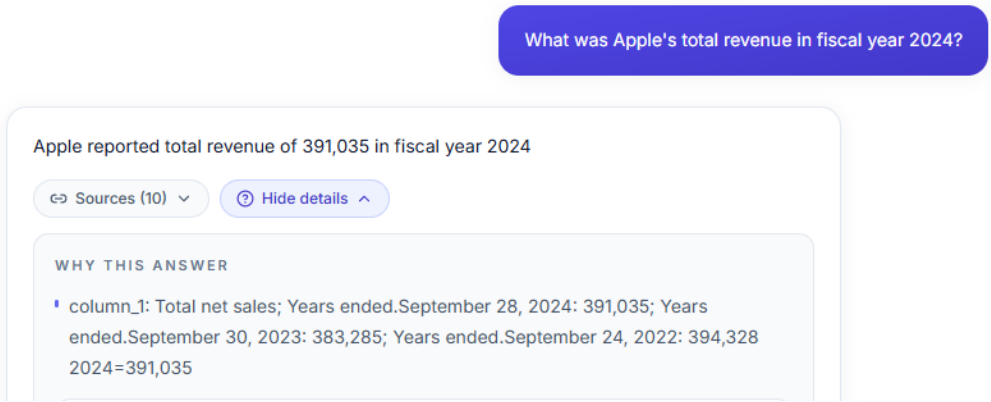


Figure 4.10: Factual Query Answer with Citation

```
# ── FACTUAL ──────────────────────────────────────────
01:56:43 | INFO   | ═══════════ QUERY ═══════════
01:56:43 | INFO   | What was Apple's total revenue in fiscal year 2024?
01:56:47 | INFO   | mongodb | financial tables | 18 results
01:56:49 | INFO   | milvus | document chunks | 30 results
01:56:49 | INFO   | Evidence collected | 48 items
01:56:52 | INFO   | Reranking | 48 → 10 candidates
01:56:52 | INFO   | Generating answer | 10 sources
01:56:52 | INFO   | Answer | Apple reported total revenue of 391,035...
01:56:52 | INFO   | ═══════════ 9.6s | 10 sources ═══════════
```

Figure 4.11: Factual Query Retrival Logs

4.5.2 Comparative Query

Unlike a factual query that targets a single value, a comparative query must retrieve and align figures from multiple companies. The query used was "Compare Apple and Google's net income for fiscal year 2024."

The system identified both companies and called the same two specialist agents. Because Apple and Google had to be searched separately, the MongoDB agent retrieved 64 datas while the Milvus agent retrieved 60 datas of both 10-K reports from the database, giving 124 items total.

The reranker had a different job here. Instead of ranking by relevance to one company, it had to keep evidence from both sides in the final selection. The 10 chosen items covered each company's net income along with supporting context.

The answer placed both figures side by side: Apple at $93,736 million, Google at $100,118 million, each with its own citation. Processing took 15.3 seconds, longer because retrieval ran across two documents. Figure 4.12 shows the frontend and Figure 4.13 the backend log.

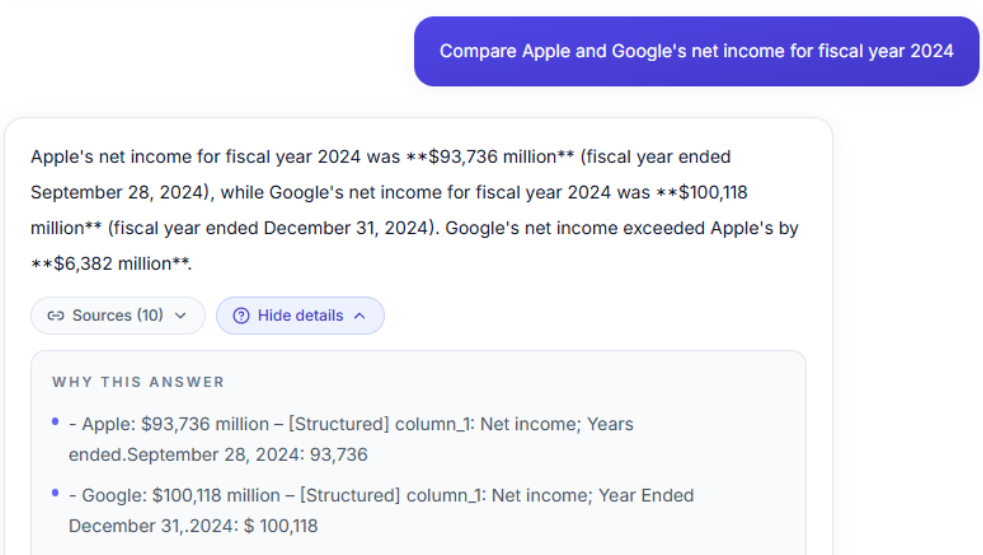


Figure 4.12: Comparative Query Answer with Citation

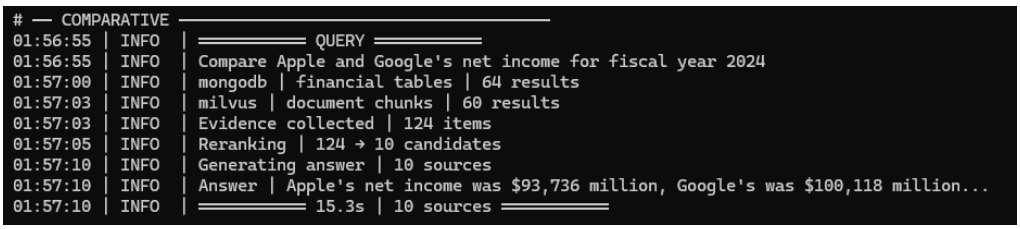


Figure 4.13: Comparative Query Retrival Logs

4.5.3 Analytical Query

Where factual and comparative queries retrieve specific values, analytical queries ask the system to explore connections and also perform calculations by trigerring the calculator agent. The query used was “ Calculate Amazon liabilities-to-assets ratio for 2024”.

The system called three specialist agents in parallel. The MongoDB agent retrieved 59 rows containing Amazon's total liabilities and total assets from the balance sheet. Milvus agent retrieved 30 text passages for context. The Calculator agent then computed the ratio from those two values, returning a single derived result. Together the tools produced 90 evidence items.

Then, the reranker selected 10 items. Due to the calculator output was derived directly from retrieved table rows, it was treated as more authoritative than the raw values alone and placed at the top rank.

Finally, the answer reported the computed ratio alongside the source values used in the calculation, with citations linking to the specific balance sheet rows. Processing took 18.8 seconds. Figure 4.14 shows the frontend and Figure 4.15 the backend log.

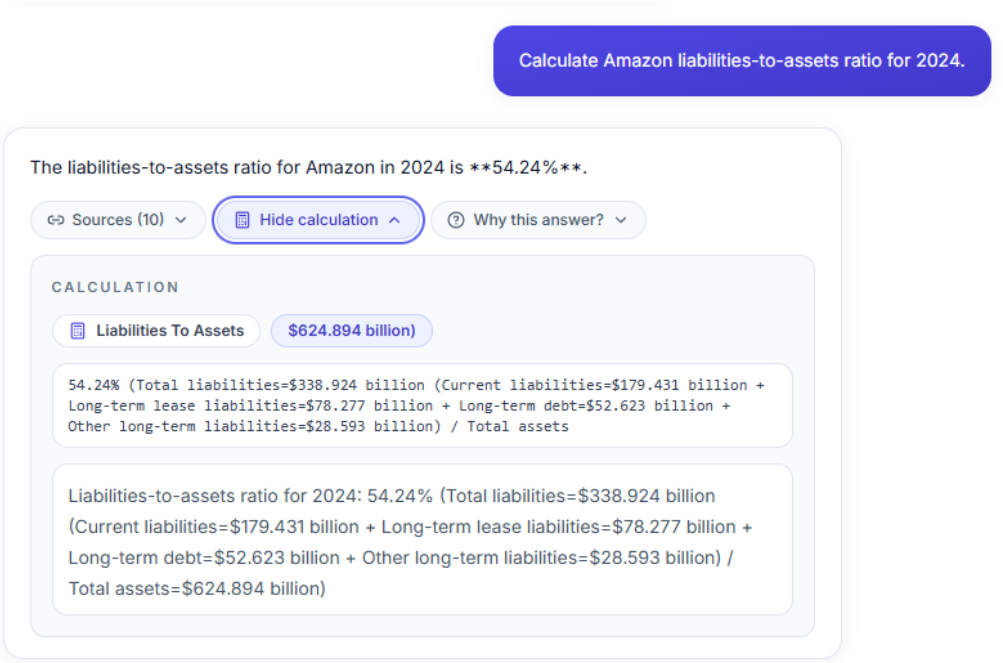


Figure 4.14: Analytical Query Answer with Calculation

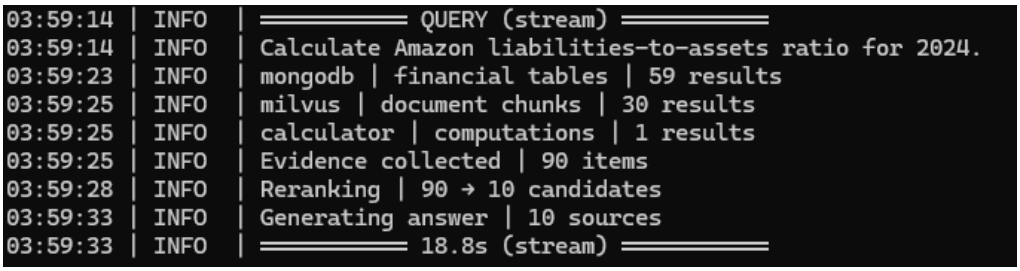


Figure 4.15: Analytical Query Retrival Logs

4.6 Evaluation Setup

The query walkthroughs in Section 4.5 demonstrated that the system executes all query types as designed. Performance was assessed by evaluating 150 pairs of financial questions and answers. The conditions are outlined in this section.

The dataset covers 150 annotated instances from the 2024 10-K reports of Amazon, Apple, and Google, consisting of 60 factual, 48 comparative, and 42 analytical queries. Each example associates a query with a corresponding reference answer and annotated evidence targets that identify the accurate document segments, table rows, or graph nodes.

All 150 samples were processed using the same pipeline with a set baseline: 30 document chunks, 30 entities, 30 relationships, and 30 table records per query, graph traversal depth of three, reranking of the top 30 candidates, and a final selection of 10 evidence pieces. The evaluation used DeepSeek v4 Flash for retrieval and answer synthesis, utilising OpenAI's text-embedding-3-large by dimensional vectors of 3072. The retrieval pipeline operated with a concurrency of five. RAGAS employed six concurrent workers and a batch size of 20.

The evaluation has two phases. Section 4.7 assesses retrieval quality, which is the frequency with which the system identifies the appropriate evidence. Section 4.8 evaluates response quality utilising RAGAS (Es et al., 2023) across five distinct parameters. Both make use of the same dataset, baseline, and environment.

4.7 Retrieval Evaluation Results

150 questions from the dataset, including factual, comparison, and analytical query types, were used for the retrieval evaluation. The retrieval evaluation was passed on all 150 questions, resulting in a 100% pass rate. This outcome demonstrates that the retrieval pipeline was able to provide relevant supporting data for each of the dataset's evaluated questions.

The complete retrieval performance is summarised in Table 4.4. The system obtained Hit@5 of 99.33% and Hit@10 of 100.00%, indicating that relevant evidence was consistently found in the top 10 results and nearly frequently retrieved within the first 5 results. Recall@5 was 99.00%, and Recall@10 was 100.00%, further confirming that the final retrieved context set contained the requisite reference evidence. With MRR of 0.8511, NDCG@5 of 0.8263, and NDCG@10 of 0.8301, the ranking metrics were likewise excellent. These numbers show that the algorithm not only found the right evidence, but it also typically placed it close to the top of the list of results.

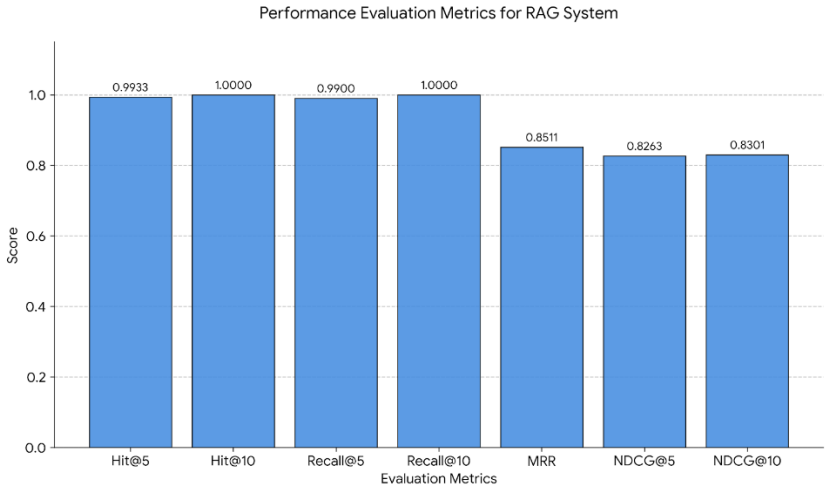


Figure 4.16: Retrieval Evaluation Metrics

4.7.1 Per-Mode Breakdown

The per-mode results demonstrate that although ranking quality varied with query complexity, retrieval performance remained remarkably stable across all query categories. With 60 out of 60 questions passing, Hit@5 of 1.00, MRR of 0.9667, and NDCG@5 of 0.9754, factual queries achieved the best results. This is to be expected since factual enquiries typically rely on concrete proof, like a certain revenue amount, an attribute of the business, or a statement from a single 10-K report.

Additionally, comparative queries achieved a Hit@5 of 1.00 and 48 out of 48 passed questions. Relevant comparative evidence was typically ranked fairly early, as indicated by the MRR of 0.9427. The algorithm discovered the necessary evidence, but the NDCG@5 score of 0.7666 was lower than the factual score, indicating that the ordering of several pieces of comparative evidence was less consistent. This makes sense for cross-company queries since the system frequently needs to retrieve and prioritise evidence from multiple companies in order to formulate an answer.

The most challenging retrieval category was analytical queries. All 42 analytical questions passed the overall results, although Hit@5 lowered to 0.9762, MRR dropped to 0.5813, NDCG@5 was 0.6814. The most relevant evidence was not always at the top of the ordered list, but these findings

demonstrate that the algorithm could nonetheless locate the required evidence. The retrieval task is more complicated than finding a single factual value since analytical queries frequently call for evidence from tables, narrative segments, graph relationships, and numerical computations. Table 4.1 shows the per-mode evaluation metric results.

Table 4.1: Per-Mode Evaluation Metric Reults

Mode	Count	Hit@5	MRR	NDCG@5
Factual	60	1.00	0.9667	0.9754
Comparative	48	1.00	0.9427	0.7666
Analytical	42	0.9762	0.5813	0.6814

4.8 RAGAS Answer Evaluation Results

The final 150-question dataset's standard Retrieval-Augmented Generation Assessment (RAGAS) answer evaluation is highlighted in this section. With 150 successful examples, zero failing examples, no missing metric data, and no zero-metric rows, the evaluation was successful for all 150 examples. The run used `text-embedding-3-large` as the embedding model with dimensional vectors of 3072 and DeepSeek v4 Flash for LLM-based evaluation. A otal of 6 RAGAS max workers, a batch size of 20, and an answer relevancy strictness value of 1 were used in the RAGAS configuration.

4.8.1 Overall Results

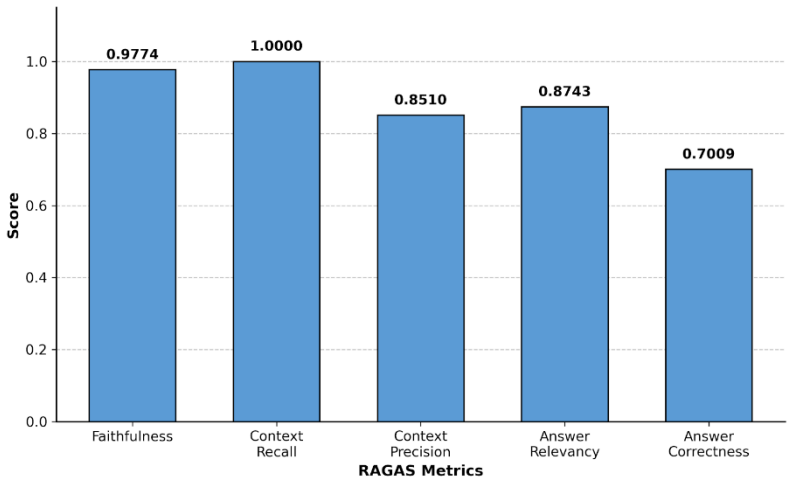


Figure 4.17: RAGAS Evaluation Results

The total results indicate that the system is able to ground generated answers in the collected data to a very high degree. The faithfulness was 97.74%, meaning that the answers were mostly supported by the circumstances and unsupported generation was rarely in this evaluation. At the context level, recall was 100.00% suggesting that the retrieved contexts always provided the information needed by the reference answers. This is an important conclusion, as it shows that the retrieval stage provided enough evidence for the answer generating stage for the whole evaluation set.

However, the answer correctness score of 70.09% reveals that retrieval of strong evidence did not always result in entirely right final answers. The distinction is crucial for a fair reading of the system. The model mostly provided the required evidence and remained based in that evidence, although several answers were still inaccurate during the synthesis. The fundamental quality restriction is therefore not the lack of source material, but the transformation of gathered data into precise, full and quantitatively correct answers.

4.8.2 Per-Metric Discussion

Faithfulness achieved the highest response rate at 97.74%. This indicates that the answer generation process infrequently produced unsupported statements beyond the retrieved context. This is a considerable advantage for a financial report question-answering system, as unsupported statements may result in incorrect understandings of company performance, risk elements, or financial metrics.

Futhermore, the context recall reached 100.00%, indicating that the retrieval process continuously provided contexts that matched the reference responses. This outcome aligns with the retrieval evaluation, wherein the system achieved full recall at the highest retrieval depth. The two metrics must be analysed independently: Recall@10 determines the presence of target evidence among the top retrieved results, whereas RAGAS context recall evaluates how well the reference responses are supported by the context set.

Moreover, the context precision was 85.10%, indicating that while the appropriate evidence was obtained, the context set included some irrelevant information. This is expected because the retrieval configuration deliberately maintains an extensive evidence budget prior to the synthesis of the final response. The trade-off is that while more comprehensive retrieval increases coverage, it may also add additional context that the response generator must filter.

The answer relevancy achieved 87.43%, indicating that the majority of generated responses remained relevant to the user inquiry. This score indicates that the system comprehensively understood the query intent and generated replies aligned with the desired financial information. The remaining gap shows that certain responses may have contained redundant information, inadequate structure, or diminished alignment with analytical and comparative enquiries.

In the conclusion, the accuracy of answers was the lowest metric at 70.09%. This outcome suggests that although the responses were relevant and

grounded, they were not consistently accurate in comparison to the reference answer. Potential sources of mistake include numerical precision, comparison ordering, and analytical completeness. These concerns are particularly significant in financial question responding, where even numerical differences or poor comparisons might diminish the utility of an otherwise supported response.

4.9 Summary

In this chapter, the functionality of each system component and subsequently evaluated the quality of retrieval and answers for 150 financial queries had been outlined. The frontend, facial recognition, and data ingestion pipeline operated as designed. Three 10-K reports were processed and saved in MongoDB, Milvus, and Neo4j. Four query walkthroughs covering factual, comparative, analytical, and calculator-driven queries verified that the pipeline operates seamlessly from start to finish.

Meanwhile, the retrieval evaluation demonstrated robust outcomes. All 150 queries were successful, achieving a Hit@5 of 0.9933, Hit@10 and Recall@10 of 1.00, and an MRR of 0.8511. Factual enquiries received the highest scores. Comparative enquiries exhibited elevated hit rates and lower ranking quality. Analytical queries had the lowest quality score due to the uneven distribution of evidence across tables, segments, and graphical relationships.The RAGAS evaluation evaluated response quality using five metrics. Faithfulness achieved 0.9774, while context recall achieved 1.00, substantiating that the replies are evidence-based. The worst indicator was answer correctness, at 0.7009.

In the end, we can see that the pattern is obvious. The system consistently obtains evidence and produces accurate responses. The constraint lies not in discovery but in synthesis. Enhancing numerical precision and analytical reasoning is the primary course of action. The system demonstrates that a safe Multi-Agent Graph RAG methodology is feasible for financial report question answering.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project successfully accomplished its goal of designing and constructing a secure Multi-Agent Graph RAG system for the analysis of financial Form 10-K files. The integration of DeepFace face recognition created a strong authentication layer that effectively safeguarded document access and user interactions. The document ingestion pipeline consistently converted unstructured PDF reports into structured chunks, entities, and relationships, systematically stored them in MongoDB, Milvus, and Neo4j. This multi-store architecture enabled ReAct-driven retrieval agents to effectively traverse semantic text, structured tables, and complex graph relations, resulting in a substantial enhancement over conventional keyword-based search techniques.

Meanwhile, the system's retrieval capabilities were thoroughly evaluated by using 150 factual, comparative, and analytical queries. The retrieval evaluation exhibited outstanding performance, attaining a 100.00% pass rate, a 99.33% Hit@5, and a 100.00% Recall@10 across all test cases. Factual queries were executed flawlessly, whereas comparative and analytical enquiries likewise achieved high success rates, but with marginally reduced ranking consistency due to the complexity of multi-hop retrieval. The evaluations validate that the multi-agent retrieval pipeline is exceptionally proficient at identifying the relevant financial tables, narrative segments, and data relationships necessary to resolve customer enquiries.

Notwithstanding the robust retrieval performance, the RAGAS evaluation underscored a particular limitation in the final response synthesis phase. The produced responses exhibited a high faithfulness to the source content (97.74%) and perfect context recall (100.00%), although the total accuracy score stabilised at 70.09%. This suggests that the main constraint lies not in the availability of correct evidence, but in the system's abilities for precise

numerical reasoning and multi-step synthesis. Finally, this investigation illustrates that a secure Multi-Agent Graph RAG methodology is highly viable for financial report analysis, provided that future iterations resolve the mathematical and reasoning constraints of the generative model.

5.2 Recommendations for Future Work

Future work should focus on improving the numerical reasoning capabilities of the synthesis agent, as Answer Correctness exhibited the lowest performance at 70.09%. Implementing specialised financial calculation frameworks or refining the synthesis model on tabular data will mitigate generation errors and enhance the accuracy of complex analytical solutions.

Then, the retrieval pipeline must be optimized to deal with analytical queries, which demonstrated the lowest ranking quality (NDCG@5 of 0.6814). Enhancing graph traversal techniques and instituting stricter reranking criteria will promote the ranking of complex financial relationships and elevate Context Precision (85.10%) by eliminating extraneous data prior to the synthesis phase.

Finally, the system need to expand its document scope and enhance its security framework. By expanding the intake pipeline to accommodate earnings transcripts and 10-K reports, the system will be validated for more extensive comparison research. Furthermore, enhancing the DeepFace implementation with liveness detection and Multi-Factor Authentication (MFA) would guarantee that the access control layer adheres to professional banking requirements.

REFERENCES

Deng, J., Guo, J. and Zafeiriou, S. (2018) 'ArcFace: Additive Angular Margin Loss for Deep Face Recognition', arXiv. Available at: https://arxiv.org/abs/1801.07698.

Edge, D. et al. (2024) 'From Local to Global: A Graph RAG Approach to Query-Focused Summarization', arXiv. Available at: https://arxiv.org/abs/2404.16130.

Lewis, P. et al. (2020) 'Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks', Advances in Neural Information Processing Systems, 33, pp. 9459-9474. Available at: https://arxiv.org/abs/2005.11401.

Yao, S. et al. (2022) 'ReAct: Synergizing Reasoning and Acting in Language Models', arXiv. Available at: https://arxiv.org/abs/2210.03629.

Es, S., James, J., Espinosa-Anke, L. and Schockaert, S. (2023) 'RAGAS: Automated Evaluation of Retrieval Augmented Generation', arXiv. Available at: https://arxiv.org/abs/2309.15217.

Auer, C. et al. (2024) 'Docling Technical Report', arXiv. Available at: https://arxiv.org/abs/2408.09869.

Jimeno Yepes, A., You, Y., Milczek, J., Laverde, S. and Li, L. (2024) 'Financial Report Chunking for Effective Retrieval Augmented Generation', arXiv. Available at: https://arxiv.org/abs/2402.05131.

Pfitzmann, B. et al. (2022) 'DocLayNet: A Large Human-Annotated Dataset for Document-Layout Segmentation', arXiv. Available at: https://arxiv.org/abs/2206.01062.

U.S. Securities and Exchange Commission (2025) Form 10-K. Available at: https://www.sec.gov/files/form10-k.pdf.

U.S. Securities and Exchange Commission (n.d.) 'How to Read a 10-K/10-Q', Investor.gov. Available at: https://www.investor.gov/introduction-investing/general-resources/news-alerts/alerts-bulletins/how-read.

Zhang, Y., Xia, M., Li, M., Mao, H., Lu, Y., Lan, Y., Ye, J. and Dai, R. (2023) 'Form 10-K Itemization', arXiv. Available at: https://arxiv.org/abs/2303.04688.

Zhu, F. et al. (2021) 'TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance', arXiv. Available at: https://arxiv.org/abs/2105.07624.

Brown, T.B. et al. (2020) ‘Language Models are Few-Shot Learners’. Available at: https://arxiv.org/abs/2005.14165.

Chen, Z. et al. (2021) ‘FinQA: A Dataset of Numerical Reasoning over Financial Data’. Available at: https://arxiv.org/abs/2109.00122.

Ji, Z. et al. (2023) ‘Survey of Hallucination in Natural Language Generation’, ACM Computing Surveys. Available at: https://arxiv.org/abs/2202.03629 (Accessed: 28 April 2026).

Vaswani, A. et al. (2017) ‘Attention Is All You Need’. Available at: https://arxiv.org/abs/1706.03762.

Wu, S. et al. (2023) ‘BloombergGPT: A Large Language Model for Finance’. Available at: https://arxiv.org/abs/2303.17564 (Accessed: 28 April 2026).

Gao, Y. et al. (2023) ‘Retrieval-Augmented Generation for Large Language Models: A Survey’. Available at: https://arxiv.org/abs/2312.10997.

Islam, P. et al. (2023) ‘FinanceBench: A New Benchmark for Financial Question Answering’. Available at: https://arxiv.org/abs/2311.11944.

Karpukhin, V. et al. (2020) ‘Dense Passage Retrieval for Open-Domain Question Answering’. Available at: https://arxiv.org/abs/2004.04906.

Rashkin, H. et al. (2023) ‘Measuring Attribution in Natural Language Generation Models’. Available at: https://arxiv.org/abs/2112.12870.

Douze, M. et al. (2024) ‘The FAISS Library’. Available at: https://arxiv.org/abs/2401.08281.

Reimers, N. and Gurevych, I. (2019) ‘Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks’. Available at: https://arxiv.org/abs/1908.10084.

Hogan, A. et al. (2021) ‘Knowledge Graphs’, ACM Computing Surveys. Available at: https://arxiv.org/abs/2003.02320

Shah, S., Ryali, S. and Venkatesh, R. (2024) ‘Multi-Document Financial Question Answering using LLMs’. Available at: https://arxiv.org/abs/2411.07264

Besrour, I. et al. (2025) ‘RAGentA: Multi-Agent Retrieval-Augmented Generation for Attributed Question Answering’. Available at: https://arxiv.org/abs/2506.16988

Guo, T. et al. (2024) ‘Large Language Model based Multi-Agents: A Survey of Progress and Challenges’. Available at: https://arxiv.org/abs/2402.01680

Wu, Q. et al. (2023) ‘AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation’. Available at: https://arxiv.org/abs/2308.08155

Deng, J. et al. (2019) ‘ArcFace: Additive angular margin loss for deep face recognition’. Available at: https://arxiv.org/abs/1801.07698

Schroff, F., Kalenichenko, D. and Philbin, J. (2015) ‘FaceNet: A unified embedding for face recognition and clustering’. Available at: https://arxiv.org/abs/1503.03832

Serengil, S.I. and Ozpinar, A. (2020) ‘LightFace: A hybrid deep face recognition framework’, ASYU, pp. 23-27. doi: 10.1109/ASYU50717.2020.9259802

Järvelin, K. and Kekäläinen, J. (2002) ‘Cumulated gain-based evaluation of IR techniques’, ACM Transactions on Information Systems, 20(4), pp. 422-446.

Ragas (n.d.) ‘RAG Evaluation Metrics’. Available at: https://docs.ragas.io/en/stable/concepts/metrics/

Yu, H. et al. (2024) ‘Evaluation of Retrieval-Augmented Generation: A Survey’. Available at: https://arxiv.org/abs/2405.07437

FastAPI (n.d.) FastAPI. GitHub repository. Available at: https://github.com/fastapi/fastapi.

Meta Open Source (n.d.) React. GitHub repository. Available at: https://github.com/facebook/react.

MongoDB, Inc. (n.d.) MongoDB Database. GitHub repository. Available at: https://github.com/mongodb/mongo.

Neo4j, Inc. (n.d.) Neo4j. GitHub repository. Available at: https://github.com/neo4j/neo4j.

OpenAI (n.d.) OpenAI Agents SDK for Python. GitHub repository. Available at: https://github.com/openai/openai-agents-python.

Shahul Es, S. et al. (2023) ‘RAGAS: Automated Evaluation of Retrieval Augmented Generation’, arXiv, arXiv:2309.15217. Available at: https://doi.org/10.48550/arXiv.2309.15217.

Vite contributors (n.d.) Vite. GitHub repository. Available at: https://github.com/vitejs/vite.

Wang, J. et al. (2021) ‘Milvus: A purpose-built vector data management system’, Proceedings of SIGMOD ’21, pp. 2614-2627. Available at: https://doi.org/10.1145/3448016.3457550.

APPENDICES