

**FUNDAMENTAL STOCK ANALYSIS WITH LLMS AND QUALITATIVE DATA -  
DEVELOPMENT OF ONTOLOGY-GROUNDED GRAPH-BASED RAG WITH  
TEXT-TO-CYPHER RETRIEVAL FOR MALAYSIAN LISTED COMPANIES**

BY

KAM CHEE QIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

## **COPYRIGHT STATEMENT**

© 2025 Kam Chee Qin. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment 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.

## **ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to my supervisors, Prof. Ts. Dr. Liew Sounng Yue, for offering me the invaluable opportunity to work on a project related to large language models, and Mr. Chng Chee Henn, for his continuous guidance and support throughout the course of this project. This experience marks an important first step in my journey towards a career in the field of artificial intelligence. A million thanks to you.

To a very special person in my life, Lee Hui Qing, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

## ABSTRACT

Fundamental analysis is essential for retail investors pursuing long-term investment, as a company's profitability ultimately drives its intrinsic value. At its core, fundamental analysis relies on deriving implicit insights—such as operational resilience, governance quality, or future growth potential—from explicit data, including financial disclosures and corporate announcements. Retail investors, however, often lack the expertise, resources, and analytical experience required to perform such analysis effectively. To address this challenge, this study proposes a corporate insight derivation module powered by Large Language Models (LLMs) that systematically transforms explicit corporate disclosures into actionable implicit insights. The module employs a novel ontology-grounded, graph-based Retrieval-Augmented Generation (RAG) pipeline with text-to-Cypher retrieval. It comprises three sub-modules: (i) an Automated Ontology Construction Module, which formalises domain-specific entities and their relationships; (ii) a Graph Construction Module, which integrates heterogeneous corporate data into a coherent knowledge graph capable of multi-hop reasoning; and (iii) a Text-to-Cypher Retrieval Module, enabling natural language queries to access the knowledge graph efficiently. The system leverages disclosures from five ACE Market-listed technology companies in Bursa Malaysia as a proof-of-concept. Evaluation results demonstrate that the proposed pipeline successfully derives implicit insights, with the Entity Deduplication process achieving a maximum deduplication rate of 73.0% and an overall rate of 66.5%, producing a compact and coherent knowledge graph. Despite limitations in ontology scalability, dynamic adaptability, and prompt robustness, the pipeline establishes a strong foundation for further refinement. The proposed module holds potential as a practical tool for retail investors, supporting more informed and rational decision-making by bridging the gap between explicit corporate data and implicit investment insights.

**Area of Study:** Application Development, Large Language Models

**Keywords:** Graph-Based RAG, Ontology-Grounded RAG, Hybrid RAG, Multi-Agent RAG, Automation of Ontology Construction, Temporal Knowledge Graph, Text-to-Cypher Retrieval,

## TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>COPYRIGHT STATEMENT</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENTS</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>LIST OF TABLES</b>	<b>xii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiii</b>
<b>Chapter 1     Introduction</b>	<b>1</b>
<b>1.1     Background Information</b>	<b>1</b>
<b>1.2     Problem Statement and Motivation</b>	<b>2</b>
<b>1.3     Objectives</b>	<b>3</b>
1.3.1    Main Objective	3
1.3.2    Sub-Objectives	3
<b>1.4     Project Scope and Direction</b>	<b>4</b>
<b>1.5     Contributions</b>	<b>5</b>
<b>1.6     Report Organization</b>	<b>5</b>
<b>Chapter 2     Literature Review</b>	<b>7</b>
<b>2.1     LLMs and RAG in Insight Derivation</b>	<b>7</b>
<b>2.2     Review of Graph-Based RAG</b>	<b>8</b>
2.2.1    GraphRAG	8
2.2.2    LightRAG	11
2.2.3    OG-RAG	13
2.2.4    Graphiti	14
<b>2.3     Review of Existing Work on Automating Ontology Construction</b>	<b>15</b>
<b>2.4     Summary and Identified Gap</b>	<b>16</b>

<b>Chapter 3</b>	<b>System Methodology/Approach</b>	<b>19</b>
<b>3.1</b>	<b>Overview of Solution</b>	<b>19</b>
3.2	System Architecture	20
3.2.1	Overall System Architecture	20
3.2.2	Graph RAG Agent Architecture	25
<b>3.3</b>	<b>Use Case Diagram and Use Case Descriptions</b>	<b>28</b>
3.3.1	Use Case Diagram	28
3.3.2	Use Case Descriptions	29
3.4	Activity Diagram	41
<b>Chapter 4</b>	<b>System Design</b>	<b>48</b>
<b>4.1</b>	<b>Overall System Design</b>	<b>48</b>
<b>4.2</b>	<b>Ontology Construction Module Design</b>	<b>50</b>
4.2.1	Ontology Extension	50
4.2.2	Ontology Refinement Loop	52
<b>4.3</b>	<b>Graph Construction Module Design</b>	<b>53</b>
4.3.1	Entity-Relationship Extraction	54
4.3.2	Entity Deduplication	55
4.3.3	Relationship Deduplication	57
4.3.4	Insertion of Entities and Relationships into the Knowledge Graph	57
<b>4.4</b>	<b>Graph Retrieval Module Design</b>	<b>58</b>
4.4.1	Response Formulation by the Chat Agent	58
4.4.2	Graph Retrieval Generation by the Graph RAG Agent	60
<b>Chapter 5</b>	<b>System Implementation</b>	<b>62</b>
<b>5.1</b>	<b>Hardware Setup</b>	<b>62</b>
<b>5.2</b>	<b>Software Setup</b>	<b>62</b>
<b>5.3</b>	<b>Setting and Configuration</b>	<b>63</b>
<b>5.4</b>	<b>System Operation</b>	<b>66</b>
5.4.1	Ontology Construction	67
5.4.2	Graph Construction	70
5.4.3	Graph Retrieval	76

<b>5.5</b>	<b>Implementation Issues and Challenges</b>	<b>80</b>
<b>Chapter 6</b>	<b>System Evaluation and Discussion</b>	<b>83</b>
<b>6.1</b>	<b>System Testing and Performance Metrics</b>	<b>83</b>
6.1.1	Testing and Performance Metrics for the Ontology Construction Module	83
6.1.2	Testing and Performance Metrics for the Graph Construction Module	84
6.1.3	Testing and Performance Metrics for the Graph Retrieval Module	84
<b>6.2</b>	<b>Testing Setup and Results</b>	<b>86</b>
6.2.1	Testing Setup and Results for Ontology Construction Agent	86
6.2.2	Testing Setup and Results for Graph Construction Agent	90
6.2.3	Testing Setup and Results for Graph Retrieval Agent	91
<b>6.3</b>	<b>Project Challenges</b>	<b>97</b>
<b>6.4</b>	<b>Objectives Evaluation</b>	<b>98</b>
<b>Chapter 7</b>	<b>Conclusion and Recommendations</b>	<b>100</b>
<b>7.1</b>	<b>Conclusion</b>	<b>100</b>
<b>7.2</b>	<b>Recommendation</b>	<b>101</b>
<b>REFERENCES</b>		<b>103</b>
<b>APPENDIX A</b>		<b>A1</b>
<b>POSTER</b>		<b>B1</b>



# LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 1.1	Company Listings and Their Respective Markets	5
Figure 2.1	Overall Architecture of Graph RAG	9
Figure 2.2	Graph RAG Global Search Dataflow	10
Figure 2.3	Graph RAG Local Search Dataflow	10
Figure 2.4	Graph RAG Dynamic Global Search	11
Figure 2.5	Light RAG Overall Architecture	12
Figure 2.6	Light RAG Graph Construction Prompt	13
Figure 2.7	OG-RAG Overall Architecture	14
Figure 3.1	Overall System Architecture Diagram	24
Figure 3.2	Graph RAG Agent Architecture Diagram	27
Figure 3.3	Use Case Diagram	29
Figure 3.4	UC001 Extend Ontology – Simplified Activity Diagram	41
Figure 3.5	UC002 Initiate Ontology Enhancement Loop - Simplified Activity Diagram	42
Figure 3.5	UC003 Extract Entities and Relationships - Simplified Activity Diagram	43
Figure 3.6	UC004 Deduplicate Extracted Entities - Simplified Activity Diagram	44
Figure 3.7	UC005 Deduplicate Extracted Relationships - Simplified Activity Diagram	45
Figure 3.8	UC006 Upsert Entities and Relationships into Knowledge Graph - Simplified Activity Diagram	46
Figure 3.9	UC007 Initiate Chat Session - Simplified Activity Diagram	47
Figure 4.1	System Block Diagram of Proposed Module	48
Figure 4.2	Example of Abbreviation List in a Prospectus	51
Figure 4.3	Threshold Mechanism for Ontology Refinement Loop	53
Figure 5.1	Python Libraries Required	63
Figure 5.2	Cloning the Repository using SourceTree	64
Figure 5.3	Creating a New Environment in Anaconda Prompt	64
Figure 5.4	Activating the Created Environment in Anaconda Prompt	64
Figure 5.5	Navigating to the Cloned Repository in Anaconda Prompt	65
Figure 5.6	Installing the Python Libraries in Anaconda Prompt	65
Figure 5.7	Configuring the Necessary API Keys in .env	66
Figure 5.8	Selecting a Jupyter Notebook	66

Figure 5.9 Initializing Variables for Database Connection	67
Figure 5.10 Initializing Ontology Construction System	68
Figure 5.11 Extending Ontology	68
Figure 5.12 Ontology Extension Entry in MongoDB	69
Figure 5.13 Enhancing Latest Ontology via Loop	69
Figure 5.14 Ontology Evaluation Report Entry in MongoDB	70
Figure 5.15 Ontology Enhancement Entry in MongoDB	70
Figure 5.16 Initializing Graph Construction System	71
Figure 5.17 Extracting Entities and Relationships from Source Text	72
Figure 5.18 Extracted Entity in MongoDB	72
Figure 5.19 Extracted Relationship in MongoDB	72
Figure 5.20 Deduplicating Entities	73
Figure 5.21 Entity Cache in MongoDB	73
Figure 5.22 Entity Cache in Pinecone	74
Figure 5.23 Entity Upsert Task Entry in MongoDB	74
Figure 5.24 Deduplicating Relationship	74
Figure 5.25 A Relationship Labelled “TO_BE_DELETED” in MongoDB	75
Figure 5.26 Entities in Pinecone	75
Figure 5.27 Entities and Relationships in Neo4J	76
Figure 5.28 Initializing Graph Retrieval System	77
Figure 5.29 Setting Up Gradio Chatbot Locally	78
Figure 5.30 The Chat Agent Confirming Entity to be Queried with the User	79
Figure 5.31 The Request Decomposition Agent Decomposing the Query	79
Figure 5.32 Cypher Query Generated by the Text2Cypher Agent	80
Figure 5.33 The Chat Agent Formulating Final Response	80
Figure 6.1 Ontology Purpose	86
Figure 6.2 Competency Question Generation Prompt for Ontology Evaluation	87
Figure 6.3 Competency Questions Generated for Ontology Evaluation	87
Figure 6.4 Ontology Competency Evaluation Prompt	88
Figure 6.5 Ontology Competency Evaluation Result - Part One	89
Figure 6.6 Ontology Competency Evaluation Result - Part Two	89
Figure 6.7 Ontology Competency Evaluation Result - Part Three	90

Figure 6.8 get_entity_count() and get_relationship_count() in the Graph Construction System	90
Figure 6.9 Entity and Relationship Deduplication Result	91
Figure 6.10 Competency Question Generation Prompt for Knowledge Graph Evaluation	92
Figure 6.11 Competency Questions Generated Prompt for Knowledge Graph Evaluation	92
Figure 6.12 Knowledge Graph Competency Evaluation Prompt	93
Figure 6.13 Response of the Graph Retrieval System to Competency Question One	93
Figure 6.14 Evaluation Report of the Graph Retrieval System Response to Competency Question One	94
Figure 6.15 Response of the Graph Retrieval System to Competency Question Two	94
Figure 6.16 Evaluation Report of the Graph Retrieval System Response to Competency Question Two	95
Figure 6.17 Response of the Graph Retrieval System to Competency Question Three	95
Figure 6.18 Evaluation Report of the Graph Retrieval System Response to Competency Question Three	96

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 3.1 UC001	Extend Ontology Use Case Description	31
Table 3.2 UC002	Initiate Ontology Enhancement Loop Use Case Description	32
Table 3.3 UC003	Extract Entities and Relationships Use Case Description	34
Table 3.4 UC004	Deduplicate Entities Use Case Description	36
Table 3.5 UC005	Deduplicate Extracted Relationships Use Case Description	39
Table 3.6 UC006	Upsert Entities and Relationships into Knowledge Graph Use Case Description	40
Table 3.7 UC007	Initiate Chat Session Use Case Description	41
Table 5.1	Hardware Specification (Laptop)	62
Table 5.2	Software specifications	63
Table 6.1	Five-point Likert scale definitions for response quality evaluation	85

## LIST OF ABBREVIATIONS

<i>IDE</i>	Integrated Development Environment
<i>LLM</i>	Large Language Model
<i>LRU</i>	Least Recently Used
<i>OWL</i>	Ontology Web Language
<i>RAG</i>	Retrieval Augmented Generation

### Chapter 1 Introduction

#### 1.1 Background Information

The stock market is a trading network that connects investors who want buy and sell shares representing fractional ownership in listed companies [1]. It serves two primary functions in the contemporary financial landscape. Firstly, it enables companies to raise capital by issuing shares to investors. Secondly, it provides individuals with the opportunity to participate in a company's profitability after purchasing its shares, typically by receiving dividends or selling shares when their market value appreciates.

For a retail investor to generate returns from the stock market in the long run, a carefully planned strategy is required. Such strategies are typically derived from one or more of the following analytical approaches: fundamental analysis, technical analysis, and sentiment analysis. Fundamental analysis seeks to determine the intrinsic value of a company. It assumes that information in the market is not perfectly disseminated, leading to scenarios in which a company's true value is not precisely reflected in its share price [2]. Consequently, investors can make profit by purchasing shares when a company is undervalued and selling them when they are overvalued. Technical analysis, on the other hand, assumes that the economic factors are reflected in the share price [2]. It focuses on identifying patterns in past market data, particularly share prices and trading volumes. By recognising these patterns, investors can determine favourable entry and exit points for trades. In contrast, sentiment analysis focuses on the collective emotional outlook of market participants by analysing data from sources such as news headlines and social media posts [3]. It is grounded on the belief that investor psychology significantly influences market prices. Hence, profits can be realised by following existing sentiment trends until a shift is detected.

Each of the analytical methods mentioned above has its own strengths and weaknesses. However, over the long term, a company's share price is ultimately driven by its profitability [1]. A profitable company tends to attract higher investor demand, which in turn drives up its share price. Therefore, fundamental analysis is generally regarded as more reliable than technical and sentiment analysis for retail investors in the long run, as it focuses on assessing a company's growth prospects and financial health.

### 1.2 Problem Statement and Motivation

Fundamental analysis can be broadly categorised into quantitative analysis and qualitative analysis, depending on the type of data utilised [4]. Quantitative analysis relies on measurable financial factors, such as profit margins and revenue growth derived from financial statements, to evaluate a firm's financial stability. In contrast, qualitative analysis emphasises non-financial aspects, including strategic direction, management quality, competitive advantage, and prevailing industry trends. Both approaches are complementary because a firm with robust financial health may underperform in the long term if it fails to adapt to emerging industry shifts, whereas a firm with visionary leadership and innovative strategies may struggle to sustain operations without sound financial fundamentals.

Although the financial and non-financial data required for fundamental analysis are widely accessible through company disclosures and industry reports, the essence of the process lies in deriving implicit insights from explicit information. For example, a disclosure indicating dependence on a single foreign supplier may appear routine, yet it implicitly signals vulnerabilities to supply chain disruptions arising from geopolitical tensions or regulatory shifts. Likewise, a company may appear unaffected by a new policy, yet its downstream clients could benefit substantially from emerging trends and regulatory incentives, indirectly boosting the company's revenue potential as their supplier. Inferring these implicit insights requires investors to process large volumes of heterogeneous information, differentiate material signals from noise, and interpret long-term implications—all of which are especially challenging for retail investors who often lack domain expertise, resources, and analytical experience. Consequently, retail investors face significant barriers in applying fundamental analysis effectively, despite having access to the same explicit data as institutional investors.

Prior to the advent of Large Language Models (LLMs), scalable and cost-effective computational methods for deriving implicit insights from unstructured corporate data were largely unavailable. The inherently unstructured and context-dependent nature of such data limited the effectiveness of traditional processing techniques. Recent advances in LLMs, however, demonstrate strong potential to address this challenge. With capabilities in contextual understanding, multimodal processing, reasoning over unstructured information, and handling open-ended queries, LLMs offer a promising means to enhance the ability of retail investors to infer implicit insights from explicit disclosures. Motivated by this opportunity, this study proposes a corporate insight derivation module powered by LLMs to support retail investors in conducting fundamental analysis more effectively.

### 1.3 Objectives

#### 1.3.1 Main Objective

- **Development of a Corporate Insight Derivation Module Powered by LLMs for Fundamental Analysis**

The objective of this study is to develop a corporate insight derivation module that leverages LLMs to support retail investors in performing fundamental analysis. Retail investors frequently encounter limitations in domain expertise, resources, and analytical capability. The proposed module addresses these challenges by deriving implicit insights from publicly available data, thereby enabling more informed and rational investment decisions.

#### 1.3.2 Sub-Objectives

- **Development of an Automated Ontology Construction Module**

The first sub-objective of this project is to develop an automated ontology construction module. An ontology, as defined in [6], is a formal representation of domain-specific entities and their interrelationships. Within the proposed system, the ontology constitutes a critical component, serving as the schema to populate the knowledge graph, which functions as the core data structure underpinning the corporate insight derivation module. Furthermore, the ontology facilitates the integration of heterogeneous data sources into a unified representation. Hence, a submodule will be implemented to automate the ontology construction process.

- **Development of a Knowledge Graph Construction Module**

The second sub-objective is to develop a module that populates a knowledge graph, defined as a network of real-world entities interconnected by their relationships [7]. The constructed ontology will provide the schema, while company disclosures, such as prospectuses and reports, will serve as the primary data sources. The knowledge graph forms the backbone of insight derivation, as implicit knowledge often emerges from chains of dependencies spanning multiple entities. The inherent structure of graph is well suited to capturing such interconnections and supports multi-hop queries that systematically link otherwise isolated facts. These linked facts can then be passed to LLMs for higher-level reasoning, thereby enabling the derivation of implicit information from explicit data.



- **Development of a Text-to-Cypher Retrieval Module for Knowledge Graph Querying**

The third sub-objective is to develop a text-to-Cypher retrieval module to facilitate querying of the knowledge graph. Cypher, the query language supported by the Neo4j graph database, will be employed to access the stored knowledge graph. This project adopts a text-to-Cypher approach, in which natural language queries are passed to the LLM and subsequently translated into Cypher statements. This module is essential, as it not only enables the LLM to retrieve relevant information from the knowledge graph efficiently, but also provides flexibility. Unlike traditional graph algorithms that are limited to predefined tasks, Cypher supports a wide range of expressive and composable queries, allowing complex traversal patterns, multi-hop dependencies, and conditional logic to be captured systematically. Consequently, the module bridges natural language interaction with the expressive flexibility and deterministic precision of structured graph querying.

### 1.4 Project Scope and Direction

For the scope of this project, the proposed corporate insight derivation module concentrates on qualitative data, with particular emphasis on strategic and operational dimensions. The strategic aspects include partnerships, supply chain dependencies, sources of competitive advantage, and related factors, while the operational aspects encompass executive roles, board and committee structures, business segments, core activities, and other organisational elements.

The module is powered by state-of-the-art LLMs from OpenAI and is implemented using Retrieval-Augmented Generation (RAG), a technique that enables LLMs to access external data sources potentially relevant to the query when generating responses. The module utilises data from 5 companies listed the ACE Market of Bursa Malaysia within the technology sector, as of September 2025, as illustrated in Figure 1.1. Companies on the LEAP Market are excluded from consideration, owing to the relatively limited transparency of their disclosures.

To demonstrate proof-of-concept and ensure accessibility, a user-friendly web-based chatbot interface is implemented. This interface enables direct interaction with the system and provides a practical means of evaluating its capabilities in supporting fundamental analysis.

Company Name	Market
AutoCount Dotcom Berhad	ACE Market
VETECE Holdings Berhad	ACE Market
ICT Zone Asia Bhd	ACE Market
Edeltec Holdings Berhad	ACE Market
SFP Tech Holdings Berhad	ACE Market

*Figure 1.1 Company Listings and Their Respective Markets*

## 1.5 Contributions

The key contributions of this project include:

1. **An ontology-grounded, auditable, scalable, and time-aware graph-based RAG pipeline** capable of deriving implicit insights from explicit data. The pipeline is grounded in a domain-specific ontology and remains auditable, as the data provided to the LLM can be traced back to specific nodes, relationships, or subgraphs in the knowledge graph. It is scalable through integration of heterogeneous data sources and support for incremental updates, and it is time-aware by explicitly encoding temporal information within the knowledge graph. This pipeline underpins the proposed corporate insight derivation module to assist retail investors in conducting fundamental analysis.
2. **An automated ontology construction module** that reduces reliance on extensive domain expert involvement by generating high-quality, domain-specific ontologies tailored to user needs. These ontologies are critical for integrating heterogeneous data sources, populating the knowledge graph required for implicit insight derivation, and enabling schema-grounded retrieval mechanisms such as text-to-Cypher.

## 1.6 Report Organization

This report is structured into chapters as follows:

**Chapter 1** introduces the project, providing the necessary background, problem statement and motivation, objectives, scope, and contributions.

**Chapter 2** presents a comprehensive literature review of contemporary graph-based Retrieval-Augmented Generation (RAG) approaches relevant to the proposed corporate

## CHAPTER 1

insight derivation module, highlighting their strengths, limitations, and the identified research gap.

**Chapter 3** provides an overview of the proposed solution, detailing the overall system architecture, component-level architecture, use case diagrams, use case descriptions, and corresponding activity diagrams.

**Chapter 4** focuses on system design, beginning with the system block diagram, followed by a discussion of the design principles of the sub-modules and their processes, illustrating how they collectively achieve the objectives of the corporate insight derivation module.

**Chapter 5** describes the implementation process, including hardware and software setup, configuration, system operation, and the challenges encountered during implementation.

**Chapter 6** presents the system evaluation, discusses project challenges, and concludes with an assessment of the objectives against the evaluation results.

**Chapter 7** concludes the report by summarising key findings, achievements, and recommendations for future work, emphasising the potential of the proposed module to assist retail investors in performing fundamental analysis.

## Chapter 2 Literature Review

### 2.1 LLMs and RAG in Insight Derivation

The core of the proposed corporate insight derivation module is the ability to derive implicit insights from explicit data. Large Language Models (LLMs) offer strong potential for this task, as they can understand context, process multimodal inputs, reason over unstructured data, and respond to open-ended queries. However, the reliability of LLM-generated responses remains a significant concern. First, LLMs are trained on static datasets and therefore cannot access information produced after their training cut-off. Second, they are prone to hallucinations, generating content that is plausible but factually incorrect. Third, their outputs are difficult to verify, as underlying sources are not explicitly referenced.

To mitigate these limitations, Retrieval-Augmented Generation (RAG) has been developed. RAG enables LLMs to access external data sources potentially relevant to a query during response generation, thereby improving reliability by addressing the issues of outdated knowledge, hallucination, and lack of traceability. Integrating RAG with LLMs makes it possible to build domain-specific applications. For example, [8] introduced a GPT-4-powered virtual analyst for fundamental stock analysis, designed to support both novice and experienced investors in practising fundamental analysis at relatively low cost.

In general, RAG consists of two phases: the indexing phase, which determines how external data sources are stored, and the retrieval phase, which defines the mechanism for querying these stored data. In conventional RAG, source documents are segmented into text chunks, which are subsequently transformed into vector embeddings and stored in a vector database during the indexing phase. In the retrieval phase, upon receiving a query, the system retrieves the top- $k$  text chunks most semantically similar to the query, or matching keywords depending on the implementation, and inserts them into the LLM's context window to generate a response. This conventional approach is referred to as vector RAG in [9], and it is effective for queries answerable using information confined to a limited set of records. However, vector RAG is less effective for sensemaking queries, that is, queries requiring comprehensive analysis across an entire dataset. Tasks involving the derivation of implicit insights from explicit data fall into this category.

To address the limitations of vector RAG in handling sensemaking queries, alternative approaches have been explored. In particular, graph-based RAG extends the conventional pipeline by leveraging graph-structured representations. By explicitly modelling entities and

their relationships, graph-based RAG enables retrieval and reasoning that go beyond surface-level text similarity, making it more suitable for sensemaking tasks, including the derivation of implicit insights from explicit data. Several variants of graph-based RAG are reviewed in Section 2.2. Section 2.3 then discusses existing work on automating ontology construction, which addresses the lack of schema observed in several graph-based RAG approaches presented in Section 2.2.

## 2.2 Review of Graph-Based RAG

### 2.2.1 GraphRAG

Recognising the limitations of vector RAG in addressing sensemaking queries, that is, queries that involve comprehensive analysis across an entire dataset, GraphRAG was introduced in [9]. GraphRAG leverages the inherent ability of graphs to form clusters, including nested clusters, to organise information hierarchically. Its pipeline employs an LLM to extract entities and relationships from the text corpus, constructs a knowledge graph from the extracted information, partitions the graph into a hierarchy of communities, and applies a bottom-up summarisation procedure along the hierarchy. Summaries of lower-level communities are progressively aggregated to form higher-level summaries, resulting in a hierarchy in which finer-grained summaries remain closer to the original text. Users can then exploit summaries at different levels of granularity to generate responses to sensemaking queries.

GraphRAG introduced a retrieval method known as global search—later termed static global search—which leverages community summaries at an appropriate level of granularity for response generation. This approach enables GraphRAG to outperform vector RAG in sensemaking queries, such as “What are the three most relevant themes in the dataset?”, even when applied to large corpora. The advantage arises because community summaries condense dispersed information into structured, higher-level insights that the model can synthesise more effectively. Beyond global search, GraphRAG also supports two additional retrieval mechanisms—local search and DRIFT search—tailored to different scenarios [10]. Local search addresses entity-centred queries, while DRIFT search enriches responses by combining local information with community summaries. Subsequently, dynamic global search was introduced as an improvement over static global search [11]. Unlike its predecessor, which incorporated all community summaries at a fixed level, dynamic global search filters out irrelevant communities and enables traversal into deeper sub-communities. This reduces the

number of summaries required in the response generation phase, thereby lowering cost and improving relevance. In addition, GraphRAG’s summarisation mechanism inherently mitigates redundancy by consolidating duplicated information into higher-level summaries.

Despite these advantages, GraphRAG poses several practical challenges. Its hierarchical summarisation process incurs significant computational costs during both indexing and retrieval. Specifically, the system must first extract entities and relationships, followed by multi-level summarisation in which lower-level summaries are recursively passed into an LLM to generate higher-level summaries until the hierarchy is complete. While this design captures information across multiple granularities, it substantially increases LLM usage during graph indexing. Later versions of GraphRAG introduce incremental updates, allowing users to configure thresholds that determine whether re-clustering and re-summarisation are necessary. However, major changes at lower levels of the hierarchy still trigger large-scale re-summarisation. Consequently, GraphRAG is less suitable for environments requiring frequent knowledge graph updates, such as rapidly evolving corporate domains. Moreover, although dynamic global search reduces cost relative to static global search, traversal in large, deep graphs remains complex and relatively slow, since the system must explore the graph to determine which summaries to include during response generation.

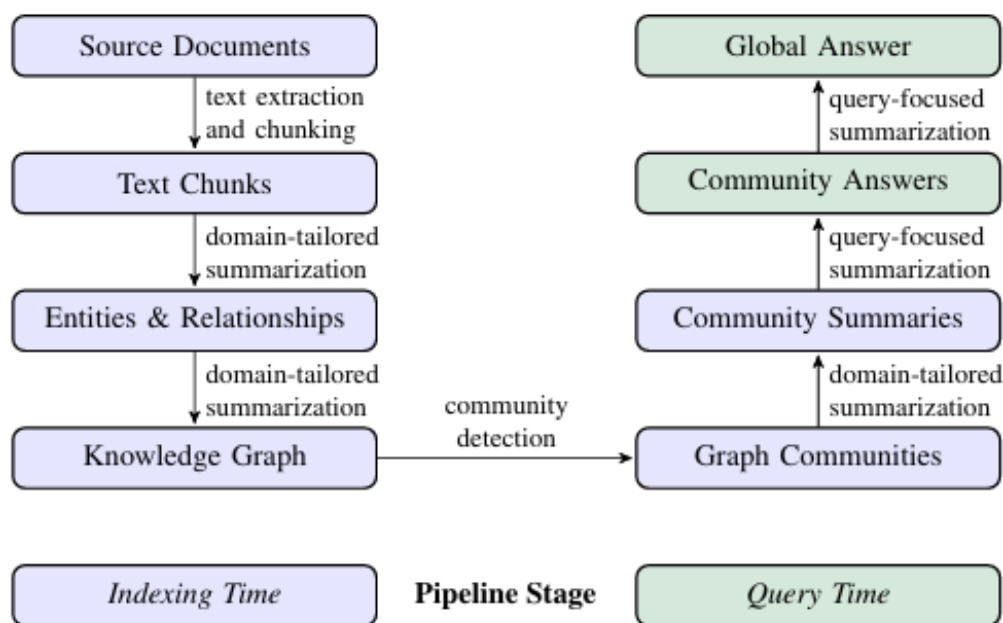


Figure 2.1 Overall Architecture of Graph RAG

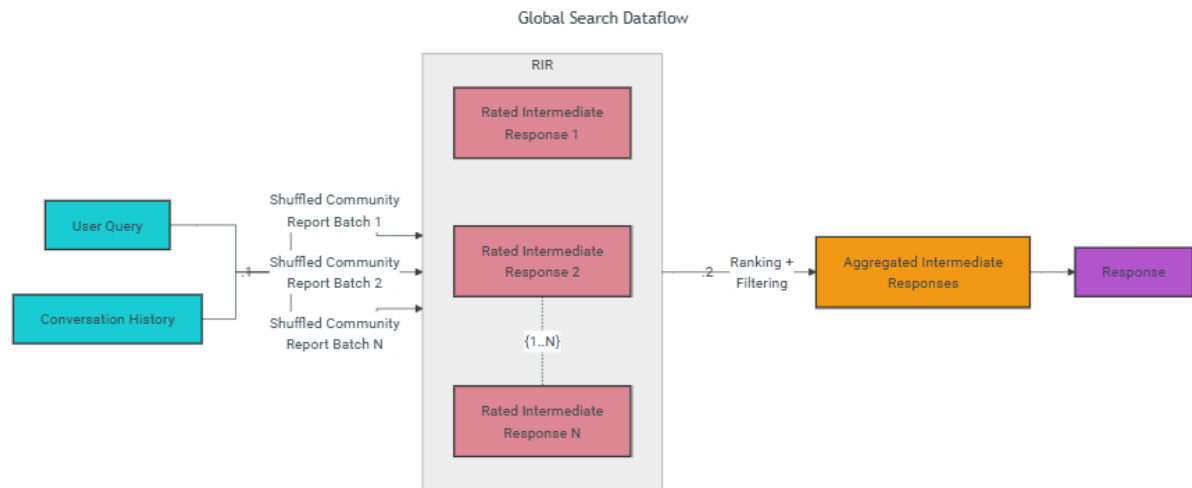


Figure 2.2 Graph RAG Global Search Dataflow

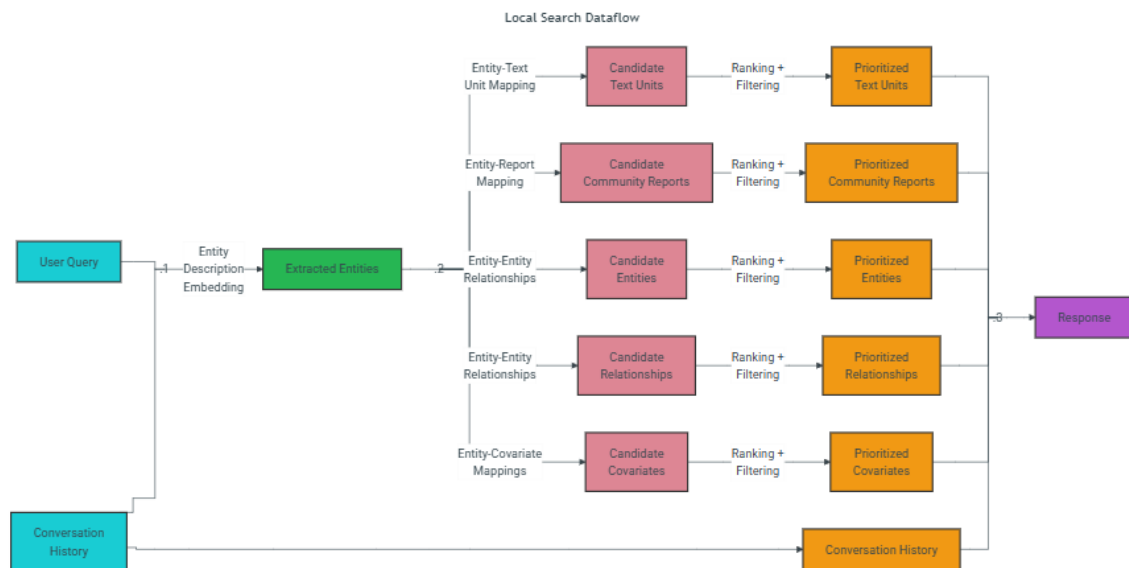
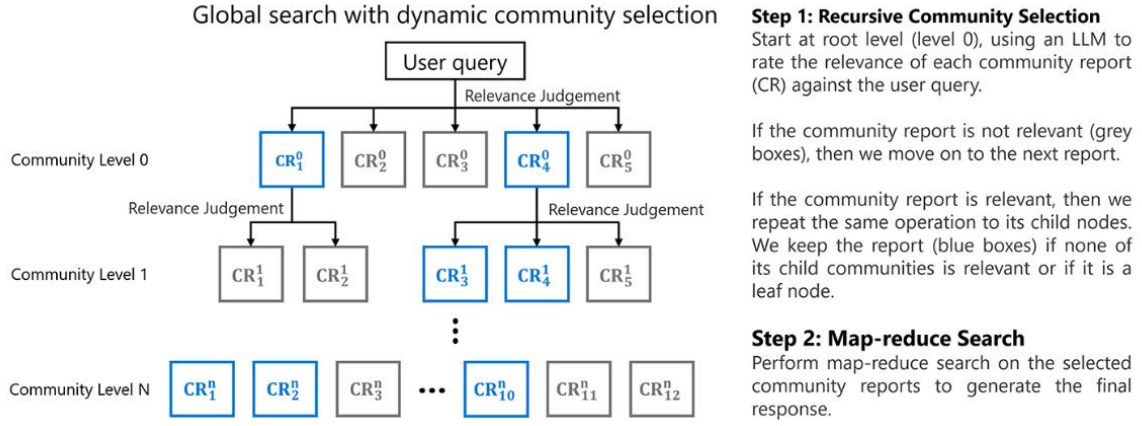


Figure 2.3 Graph RAG Local Search Dataflow



*Figure 2.4 Graph RAG Dynamic Global Search*

### 2.2.2 LightRAG

LightRAG was introduced to capture comprehensive contexts of interdependent facts across text corpora while maintaining fast retrieval and supporting incremental updates [12]. In the indexing phase, it begins by extracting entities and relationships from chunked source texts. For each extracted entity and relationship, a key–value pair is generated, where the key is vectorised and serves as the retrieval entry, and the value is a text paragraph summarised by an LLM, relevant to the entity or relationship from the source text. For entities, the index key corresponds to their names, whereas for relationships, the index key consists of words or phrases indicating a theme derived from connected entities. A deduplication process is then applied to merge identical entities and relationships. During retrieval, the system supports two modes: low-level retrieval, which addresses queries concerning a specific entity and its associated attributes or relationships; and high-level retrieval, which answers broader topics or overarching themes by aggregating information across multiple related entities and relationships. Given a user query, the system employs an LLM to extract local and global query keywords—potentially representing entity and relationship index keys respectively—and retrieves entities or relationships along with their attributes from a vector database, depending on the chosen retrieval mode. The attributes of the retrieved items are subsequently used to generate the user’s response.

LightRAG adopts a simplified form of graph indexing, omitting the clustering and community summarisation steps present in GraphRAG. Consequently, it requires only  $\frac{\text{document total tokens}}{\text{chunk size}}$  calls to the LLM during the indexing phase, significantly reducing costs. To address the issue of potentially duplicated information from heterogeneous data sources, it applies a deduplication process to merge identical entities and relationships. This process



enables the creation of a more compact knowledge graph, thereby reducing storage requirements and avoiding the propagation of redundant information to the LLM during response generation. Furthermore, LightRAG employs a vector-search approach for retrieval rather than a graph traversal method, which accelerates retrieval. Its lightweight design also enables efficient incremental updates, as the insertion of new data triggers indexing only on the additional content without necessitating re-indexing of the existing graph. The newly created subgraph can thus be seamlessly integrated into the existing structure without incurring additional computational overhead—unlike GraphRAG, which may require partial re-clustering and regeneration of community summaries upon data insertion.

Although LightRAG utilises a knowledge graph for indexing, it does not fully exploit the graph structure for retrieval. Instead, it relies on keyword matching to identify relevant entities and relationships. While cost-effective, this approach introduces several limitations. First, performance may degrade if the LLM fails to extract suitable matching keywords from the query. Second, additional engineering efforts may be necessary to sustain retrieval speed as the graph scales to millions of entities and relationships. Third, the retrieval mechanism treats entities and relationships as isolated text chunks, rather than as interconnected logical structures, since they are accessed through vector similarity search rather than systematic graph traversal. This limits performance on queries requiring reasoning across multi-hop paths. Although LightRAG allows users to configure the number of neighbouring entities to be retrieved when accessing an element from the graph, this approach is inefficient, as relevant information may not be confined to a fixed number of hops and some neighbouring entities may be extraneous. Furthermore, while LightRAG enables users to specify the entities to be captured from source texts, as shown in Figure 2.x, the relationships are not consistently extracted due to the absence of a structured and well-defined graph schema, resulting in the inclusion of irrelevant data that may not meet user requirements.

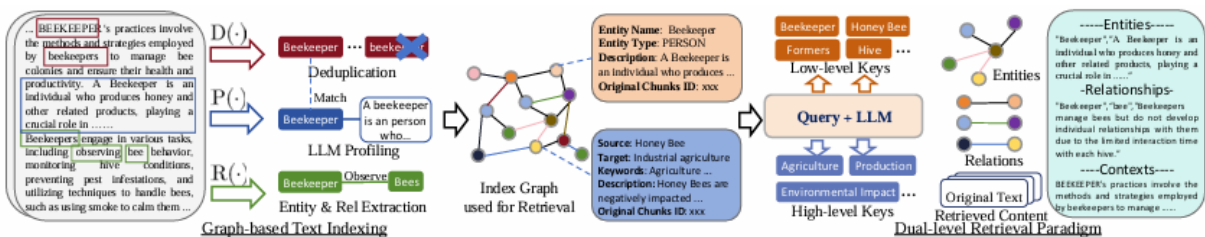


Figure 2.5 Light RAG Overall Architecture

**-Goal-**  
 Given a text document that is potentially relevant to this activity and a list of entity types, identify all entities of those types from the text and all relationships among the identified entities.

**-Steps-**  
**1. Identify all entities.** For each identified entity, extract the following information:  
 - **entity\_name:** Name of the entity, capitalized  
 - **entity\_type:** One of the following types: [organization, person, geo, event]  
 - **entity\_description:** Comprehensive description of the entity's attributes and activities  
 Format each entity as ("entity"<|><entity\_name><|><entity\_type><|><entity\_description>)

**2. From the entities identified in step 1, identify all pairs of (source\_entity, target\_entity) that are "clearly related" to each other.**  
**For each pair of related entities, extract the following information:**  
 - **source\_entity:** name of the source entity, as identified in step 1  
 - **target\_entity:** name of the target entity, as identified in step 1  
 - **relationship\_description:** explanation as to why you think the source entity and the target entity are related to each other  
 - **relationship\_strength:** a numeric score indicating strength of the relationship between the source entity and target entity  
 - **relationship\_keywords:** one or more high-level key words that summarize the overarching nature of the relationship, focusing on concepts or themes rather than specific details  
 Format each relationship as ("relationship"<|><source\_entity><|><target\_entity><|><relationship\_description><|><relationship\_keywords><|><relationship\_strength>)

**3. Identify high-level key words that summarize the main concepts, themes, or topics of the entire text. These should capture the overarching ideas present in the document.**  
 Format the content-level key words as ("content\_keywords"<|><high\_level\_keywords>)

**4. Return output in English as a single list of all the entities and relationships identified in steps 1 and 2. Use \*\*\*\* as the list delimiter.**

**5. When finished, output <[COMPLETE]>**

**-Real Data-**  
**Entity\_types:** {entity\_types}  
**Text:** {input\_text}

**Output:**

**Graph Construct Prompt**

Figure 2.6 Light RAG Graph Construction Prompt

### 2.2.3 OG-RAG

To ensure that a large language model (LLM) has access to domain-specific information while providing responses that support decision-making, OG-RAG employs an ontology—a structured representation of domain-specific entities and their relationships [6]—to populate its knowledge graph. Beyond ontology grounding, OG-RAG adopts a hypergraph representation, wherein a hyperedge connects multiple related facts rather than just two, as in conventional graphs. This design enables OG-RAG to condense information from heterogeneous data sources. The process begins by employing an LLM to map documents onto the ontology, producing factual blocks. Each block is then flattened into a hyperedge comprising hypernodes, where each hypernode is a key–value pair that represents a fact. During query processing, OG-RAG identifies relevant hypernodes based on similarity to both the key and the value, and subsequently applies a greedy algorithm to select the minimal set of hyperedges covering these nodes. This approach yields a compact, semantically rich context that enhances LLM performance.

Unlike domain-agnostic embedding-based systems, OG-RAG leverages ontology-grounded information to generate responses. This enables the capture of complex relationships and nuanced facts that are essential for precise retrieval in specialised domains. Moreover, OG-RAG provides LLMs with domain-specific knowledge without requiring fine-tuning, which is a more costly alternative. The ontology further ensures that domain knowledge is interpreted unambiguously across different LLMs. Additionally, OG-RAG adapts gracefully to evolving

information landscapes, as the underlying ontology can be extended over time. From a computational perspective, retrieval in OG-RAG is performed using vector similarity against hypernode keys and values, making it faster and more cost-effective than graph-based RAG approaches that rely on graph traversal, such as GraphRAG. Furthermore, the responses produced by OG-RAG are verifiable, since each hypernode key contains traceability information that links the context back to the original data source.

Nevertheless, OG-RAG is highly dependent on a well-designed ontology to achieve optimal performance. As each domain structures its knowledge and terminology differently, practitioners must invest considerable effort in constructing or selecting an appropriate ontology, which may extend deployment timelines. Furthermore, modifications at higher-level classes within a deep ontology can necessitate substantial recomputation due to the flattening process, as each hypernode key retains information inherited from its ancestor nodes. This hierarchical retention also increases storage requirements, thereby elevating storage costs. In addition, OG-RAG lacks a deduplication mechanism to address redundant information that may arise when integrating heterogeneous sources. Finally, although hypergraphs offer a more expressive means of capturing multifaceted relationships, they are less intuitive than property graphs, which model data as entities and relationships where both can carry attributes, and where relationships are restricted to pairs of entities. This relative lack of intuitiveness may hinder practical deployment.

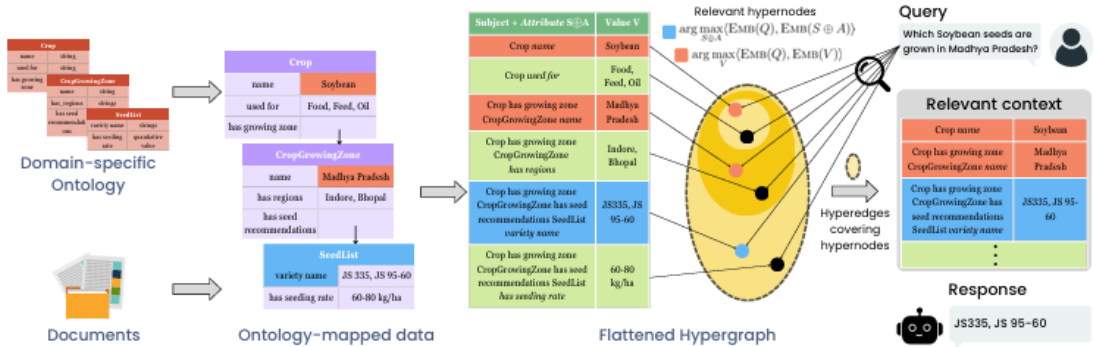


Figure 2.7 OG-RAG Overall Architecture

## 2.2.4 Graphiti

Graphiti, the knowledge graph engine underpinning Zep—a memory layer service for AI agents—constructs a non-static, temporally aware knowledge graph [13]. It addresses temporal challenges through a bi-temporal modelling approach, whereby two distinct timelines are tracked for each piece of data: the time of event occurrence and the time of data transaction.

When newly acquired temporal facts contradict existing ones, the system employs an LLM to resolve the temporal characteristics of the edges, prioritising newer information when determining whether to invalidate earlier edges. Consequently, Graphiti maintains a time-aware knowledge graph that preserves historical records while seamlessly incorporating new updates.

Graphiti enables reasoning over continuously evolving information through its bi-temporal modelling and LLM-based temporal resolution approach. This capability provides a distinct advantage over RAGs that are limited to operating on static text corpora—an inherent drawback in real-world, dynamic environments. Moreover, Graphiti supports both localised queries targeting specific entities and more complex global queries that require broader contextual understanding. This is facilitated by its three-layer graph architecture, comprising an episode subgraph (bottom layer), a semantic entity subgraph (middle layer), and a community subgraph (top layer).

Despite its strengths, Graphiti has several limitations. First, it assumes that temporal information—whether in relative form (e.g., “next Monday”) or absolute form (e.g., 21 June 2020)—is explicitly available within the source data. However, there are cases where temporal context is needed but not explicitly stated, such as “before acquisition,” which Graphiti cannot gracefully handle. Second, the three-layer graph structure, while enhancing the range and depth of queries, introduces additional maintenance complexity. This added overhead can lead to increased operational costs over time, especially in large-scale deployments.

### 2.3 Review of Existing Work on Automating Ontology Construction

An ontology, defined as a formal representation of domain-specific entities and their interrelationships, can serve as a schema for populating a knowledge graph in graph-based RAG. It enables LLMs to integrate information from heterogeneous data sources in a consistent manner, ensures unambiguous interpretation of domain knowledge, provides access to domain expertise without the need for fine-tuning, and supports retrieval mechanisms that rely on a schema, such as text-to-Cypher. Despite these advantages, integrating an ontology into a graph-based RAG poses challenges. Domain-specific ontologies may not always be available or may fail to meet user requirements, while designing one from scratch demands significant effort and involvement from domain experts.

In this context, [14] and [15] explored semi-automated approaches to ontology construction, generating ontologies from a set of human-verified or human labelled competency questions. This method ensures that the resulting ontology is more closely aligned

with user requirements, as competency questions explicitly reflect the queries practitioners expect the system to address. Moreover, the use of competency questions facilitates ontology evaluation, since the same questions can be used to test whether the modelled entities and relationships adequately support the intended reasoning tasks. Both [14] and [15] employed the Ontology Web Language (OWL)—a language for modelling concepts and their interrelationships [16]—to generate their ontologies. The use of OWL provides a mature syntax and ensures interoperability with tools and systems that support the standard.

Nevertheless, ontology generation from competency questions introduces certain limitations. While it helps ensure alignment with user needs, it risks overfitting, as the LLM may attempt to model entities and relationships narrowly around the specific questions rather than representing the broader domain knowledge. This can reduce flexibility as competency questions evolve and limit generalisability. Furthermore, this approach merely shifts the burden from constructing the ontology to formulating competency questions, without fully eliminating manual effort, as human verification is still required when preparing them. In addition, although OWL is expressive and machine-interoperable, it poses a practical barrier for adoption, as practitioners must acquire expertise in OWL to maintain and extend the ontology, which may hinder the wider application of ontology in graph-based RAG systems.

### 2.4 Summary and Identified Gap

The essence of fundamental analysis lies in deriving implicit insights from explicit data. However, retail investors face significant challenges in conducting fundamental analysis due to limited domain expertise, resources, and analytical experience. To address this, a corporate insight derivation module is proposed to support retail investors in performing fundamental analysis. Developing such a module requires a graph-based RAG, as graph-structured data captures implicit insights emerging from chains of dependencies across multiple entities and supports multi-hop queries that systematically connect otherwise isolated facts. These linked facts then provide additional context for the LLM to reason over, enabling more reliable responses by mitigating issues such as outdated knowledge, hallucination, and lack of traceability. Consequently, combining graph structures with LLMs—embodied in graph-based RAG—offers a scalable and cost-effective means of deriving implicit insights from explicit data, a challenge that traditional computational methods struggle to address.

Despite this promise, contemporary graph-based RAGs remain limited in their readiness for real-world deployment. GraphRAG’s hierarchical summarization design, while capable of producing context-rich responses and integrating heterogeneous data, does not adapt well to

incremental updates, limiting scalability in dynamic environments. LightRAG, though more cost-efficient by employing a property graph instead of hierarchical summarization, does not effectively exploit graph structures for retrieval and lacks schema grounding, leading to ad-hoc and inconsistent extraction. OG-RAG, which leverages ontology-grounded hypergraphs, requires a carefully crafted ontology tailored to user needs, and its reliance on hypergraphs is less intuitive than property graphs; furthermore, the need to pre-flatten hypernodes before passing them into an LLM reduces flexibility for substantial updates. Graphiti, with its bi-temporal modeling, introduces time-awareness but struggles when temporal context is not explicitly stated, and its three-layer graph design increases maintenance costs.

In contrast, property graphs—which model data as entities and relationships, both capable of carrying attributes, and where relationships are restricted to pairs of entities—provide a simpler, more cost-efficient structure. They support incremental updates seamlessly, since new data can be appended without restructuring, and allow temporal or additional contextual information to be encoded as attributes. Moreover, property graphs are compatible with diverse retrieval methods, from traditional graph algorithms to declarative query languages such as text-to-Cypher. However, deduplication is necessary to ensure compactness, which may incur additional costs depending on the mechanism used. A schema—typically an ontology—is also required to avoid ad-hoc and inconsistent extraction, particularly in scenarios where LLM responses must be grounded in domain knowledge or when retrieval relies on formal query languages.

The challenge, however, lies in constructing a well-crafted, domain-specific ontology. Contemporary approaches rely on competency questions, which risk producing ontologies that are overfitted, inflexible, and poorly scalable as requirements evolve. This approach also shifts manual effort from ontology design to competency question formulation and verification, maintaining a high level of human involvement. Furthermore, most ontologies are expressed in OWL, which, while highly expressive and machine-interoperable, poses significant barriers for practitioners without expertise in formal semantic web technologies.

Against this backdrop, this project proposes a novel graph-based RAG pipeline that is ontology-grounded (anchored in a domain-specific ontology), auditable (with insights traceable to specific nodes, relationships, or subgraphs), scalable (capable of integrating heterogeneous sources with incremental updates), and time-aware (explicitly encoding temporal information within the knowledge graph). To address the lack of suitable domain-specific ontologies, the pipeline further introduces an ontology construction automation

## CHAPTER 2

module, designed to generate tailored, maintainable ontologies that align with evolving user needs.

## Chapter 3 System Methodology/Approach

### 3.1 Overview of Solution

The proposed solution to the challenges retail investors face in performing fundamental analysis—namely the lack of domain expertise, resources, and analytical experience—is a corporate insight derivation module. This module is powered by a novel ontology-grounded, graph-based Retrieval-Augmented Generation (RAG) pipeline with text-to-Cypher retrieval, which serves as the backbone for insight generation. To address the key limitations of contemporary graph-based RAG, a novel graph-based RAG pipeline is introduced to power the proposed corporate insight derivation module. The architecture of this pipeline is described in Section 3.2, with several key design decisions summarised below:

#### 1. Property Graph for Modelling

A property graph is employed to model all captured information. No additional hierarchical summarisation is performed, as this would introduce extra cost and hinder adaptability to incremental updates. The property graph's ability to store attributes in both entities and relationships further enables the encoding of temporal information and other contextual details.

#### 2. Cache-Based Entity and Relationship Deduplication

As the property graph is used to capture information from heterogeneous data sources, deduplication of entities and relationships is necessary. Without deduplication, the graph would contain substantial amounts of redundant information, since identical facts may appear across multiple sources. This would increase storage requirements and result in unlinked facts, thereby hindering effective insight derivation. To address this, a cache-based deduplication mechanism is employed to strike a balance between maintaining a compact graph and avoiding exhaustive deduplication, which would incur higher computational cost. In this approach, the graph tolerates a configurable degree of duplication, determined by the cache size.

#### 3. Ontology Construction Automation Module

To ensure systematic and non-ad hoc population of the knowledge graph, as well as consistent interpretation by Large Language Models (LLMs) during graph population, an ontology is used to ground the knowledge graph. To address the challenge of deriving a high-quality ontology that aligns with user requirements, an automated ontology construction module is introduced. The ontology is expressed in natural



language rather than in the Web Ontology Language (OWL) to lower the adoption barrier for end users.

### 4. **Cypher for Graph Retrieval**

Cypher is adopted as the query language to leverage the graph structure for uncovering implicit insights from explicit data. Such insights typically emerge from chains of dependencies that would otherwise appear as isolated facts. Compared with traditional graph traversal algorithms, Cypher provides richer and more flexible retrieval capabilities.

### 5. **Vector RAG to Complement Graph RAG**

A Vector RAG component is introduced to complement Graph RAG, as not all information can be effectively captured within the graph due to the limited expressiveness of the ontology. In addition, answering simple and direct queries through Graph RAG alone is often inefficient in terms of both cost and performance. Vector RAG therefore enables Graph RAG to focus on modelling information that supports the derivation of implicit insights from explicit data.

## 3.2 **System Architecture**

### 3.2.1 **Overall System Architecture**

The proposed ontology-grounded graph-based RAG system with a text-to-Cypher retrieval module, which underpins the corporate insights derivation model, is divided into three primary modules, as illustrated in the system architecture diagram in Figure 3.1. A detailed description of each module is provided below. For clarity, the term agent in this context refers to a software component powered by a LLM that is responsible for a specific task or a set of tasks.

#### 1. **Ontology Construction Module**

The ontology construction module accepts company disclosures, such as prospectuses, along with the user-defined ontology purpose as input, and generates a natural language ontology as output. It comprises three agents: the Ontology Construction Agent, the Ontology Evaluation Agent, and the Ontology Enhancement Agent.

- The **Ontology Construction Agent** extends the existing ontology or creates a new ontology using the provided documents while adhering to the defined ontology purpose.

- The **Ontology Evaluation Agent** evaluates the extended ontology at both high-level (overall structure) and low-level (entity and relationship attributes), ensuring alignment with the ontology purpose.
- The **Ontology Enhancement Agent** refines the ontology at both high-level and low-level based on the evaluation report produced by the Ontology Evaluation Agent, again in line with the ontology purpose.

The workflow proceeds as follows: once the ontology is extended by the Ontology Construction Agent, it is passed to the Ontology Evaluation Agent. If enhancements are required, the ontology is forwarded to the Ontology Enhancement Agent; otherwise, it is stored in the database. The Ontology Enhancement Agent determines whether enhancement should be performed based on a pre-configured enhancement threshold. This threshold is dynamic: it increases with the number of iterations and the unresolved issues identified by the Ontology Evaluation Agent. The increasing threshold ensures that subsequent enhancements become progressively stricter to avoid an infinite enhancement loop.

## 2. Graph Construction Module

The graph construction module accepts company disclosures and an ontology as input, instantiates entities and relationships from both the ontology and the source text, and subsequently deduplicates the extracted entities and relationships. This module consists of three agents: the Entity–Relationship Extraction Agent, the Entity Deduplication Agent, and the Relationship Deduplication Agent.

- The **Entity–Relationship Extraction Agent** identifies entities and relationships defined in the ontology from the company disclosures.
- The **Entity Deduplication Agent** compares each extracted entity against existing entities in the knowledge graph to determine whether it should be merged with an existing entity or inserted as a new one. Prior to evaluation, a similarity search is employed to identify the most likely candidate entity for comparison. The associated relationships and attributes of the two candidates are then passed to the Entity Deduplication Agent—implemented using an LLM—which makes the final merging decision.
- The **Relationship Deduplication Agent** evaluates pairs of relationships of the same type between two entities to determine whether merging is required. The decision is based on the descriptions of the respective relationships.

The workflow proceeds as follows: entity deduplication is performed after entity–relationship extraction. Relationship deduplication, however, can only be performed after entity deduplication, as merging an entity into another requires its associated relationships to be reassigned to the new entity. Performing relationship deduplication prior to entity deduplication would therefore result in inconsistencies.

### 3. Graph Retrieval Module

The graph retrieval module accepts a user request as input and generates a response based on its nature. At a high level, this module comprises three agents: the Chat Agent, the Vector RAG Agent, and the Graph RAG Agent. Both the Vector RAG Agent and the Graph RAG Agent are in fact collections of sub-agents, each responsible for specific tasks.

- The **Chat Agent** serves as the primary interface with the user. It determines whether a request should be rejected, selects the appropriate agent to address the request, and synthesises a final response based on the retrieved information.
- The **Vector RAG Agent** processes the query formulated by the Chat Agent. It performs semantic search against text chunks derived from company disclosures within the project scope and generates a response based on the retrieved text. This agent operates through an internal set of collaborating agents. Its implementation has been carried out by another student taking the final-year project; thus, detailed discussion of its design is omitted here.
- The **Graph RAG Agent** is capable of deriving implicit insights from explicit data. Internally, it incorporates a collection of sub-agents to accomplish this task. It accepts the query formulated by the Chat Agent, decomposes the query into sub-queries if applicable, and rephrases each sub-query (or the original query if decomposition is unnecessary). It then performs text-to-Cypher retrieval based on the rephrased query, evaluates the retrieved results and Cypher queries, and determines whether re-retrieval is required. Finally, it generates a response to the original query passed by the Chat Agent. Further details of the Graph RAG Agent are provided in Section 3.2.2.

The workflow is as follows: the user interacts with the corporate insights derivation model through this graph retrieval module. In practice, the user communicates directly with the Chat Agent, which autonomously invokes the appropriate agents to retrieve relevant information if the request is valid, or rejects the request otherwise. After

invoking an agent, the Chat Agent may choose to re-invoke the same agent with a modified or identical query, or invoke a different agent, depending on the adequacy of the retrieved information. The Chat Agent autonomously determines when to generate the final response, subject to a pre-defined maximum number of agent calls.

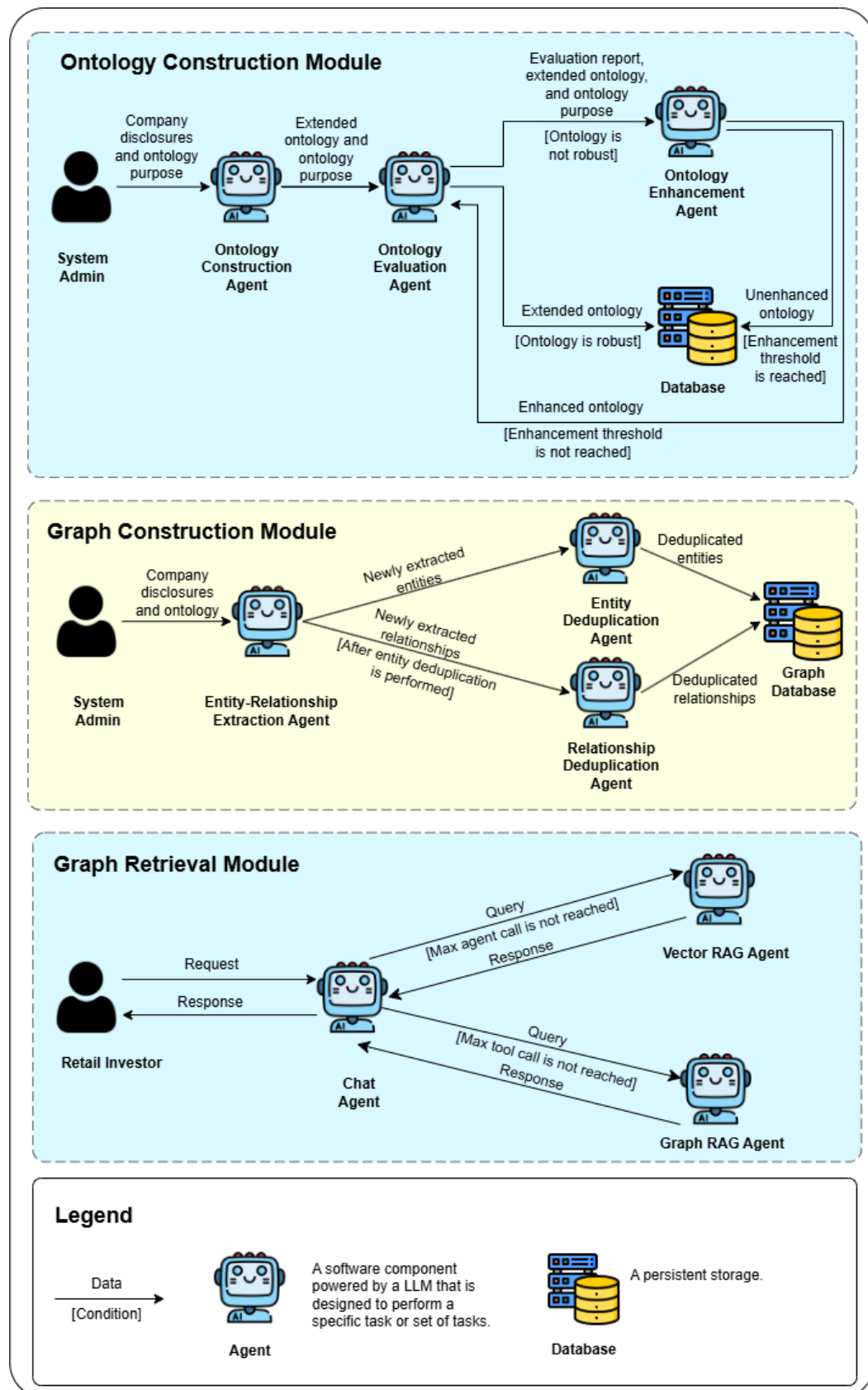


Figure 3.1 Overall System Architecture Diagram

### 3.2.2 Graph RAG Agent Architecture

The architecture of the Graph RAG Agent, which is one of the agents invoked by the Chat Agent in the graph retrieval module of the proposed ontology-grounded graph-based RAG system, is illustrated in Figure 3.2. This agent comprises a set of sub-agents working collaboratively: the Request Decomposition Agent, the Query Agent, the Text2Cypher Agent, and the Retrieval Result Compilation Agent.

- **Request Decomposition Agent:** This agent decomposes a query received from the Chat Agent into multiple sub-queries if the original query can be separated into independent queries. Decomposition enables parallel execution and more focused retrieval, thereby improving both efficiency and accuracy. If no decomposition is possible, the query is forwarded without modification.
- **Query Agent:** With access to the ontology, this agent reformulates the queries provided by the Request Decomposition Agent into ontology-aware natural language queries. This design ensures that the Text2Cypher Agent focuses solely on Cypher conversion, adhering to the principle of single responsibility. The Query Agent also evaluates the Cypher queries generated by the Text2Cypher Agent alongside the compiled retrieval results produced by the Retrieval Result Compilation Agent to determine whether re-retrieval is required. Re-retrieval may be necessary when the Text2Cypher Agent generates an incorrect Cypher query or when the rephrased query itself is inadequate. The number of allowable re-retrieval attempts is preconfigured. Additionally, the Query Agent filters irrelevant or noisy information from the compiled retrieval results before returning the refined output to the Request Decomposition Agent.
- **Text2Cypher Agent:** This agent is responsible for converting the ontology-aware natural language queries produced by the Query Agent into Cypher queries, ensuring strict adherence to the given ontology.
- **Retrieval Result Compilation Agent:** This agent compiles the raw retrieval results generated by executing the Cypher queries. Since such results may be noisy or unstructured, directly passing them to the Query Agent would increase computational cost and degrade performance. To mitigate this, a more cost-efficient LLM is configured for the Retrieval Result Compilation Agent, allowing it to focus exclusively on result compilation before handing over the output for further evaluation.

The overall workflow is as follows: the Chat Agent submits a query to the Request Decomposition Agent, which decomposes it into multiple sub-queries (up to a preconfigured

maximum) to improve concurrency and retrieval focus. The Query Agent then reformulates each query into an ontology-aware natural language query, which the Text2Cypher Agent converts into Cypher queries. These queries are executed against the knowledge graph, and the resulting outputs are compiled by the Retrieval Result Compilation Agent. The compiled results, together with the corresponding Cypher queries, are evaluated by the Query Agent to determine whether re-retrieval is necessary. Once the final retrieval results are obtained, the Query Agent filters irrelevant information and forwards the processed output to the Request Decomposition Agent, which aggregates the results and returns them to the Chat Agent.

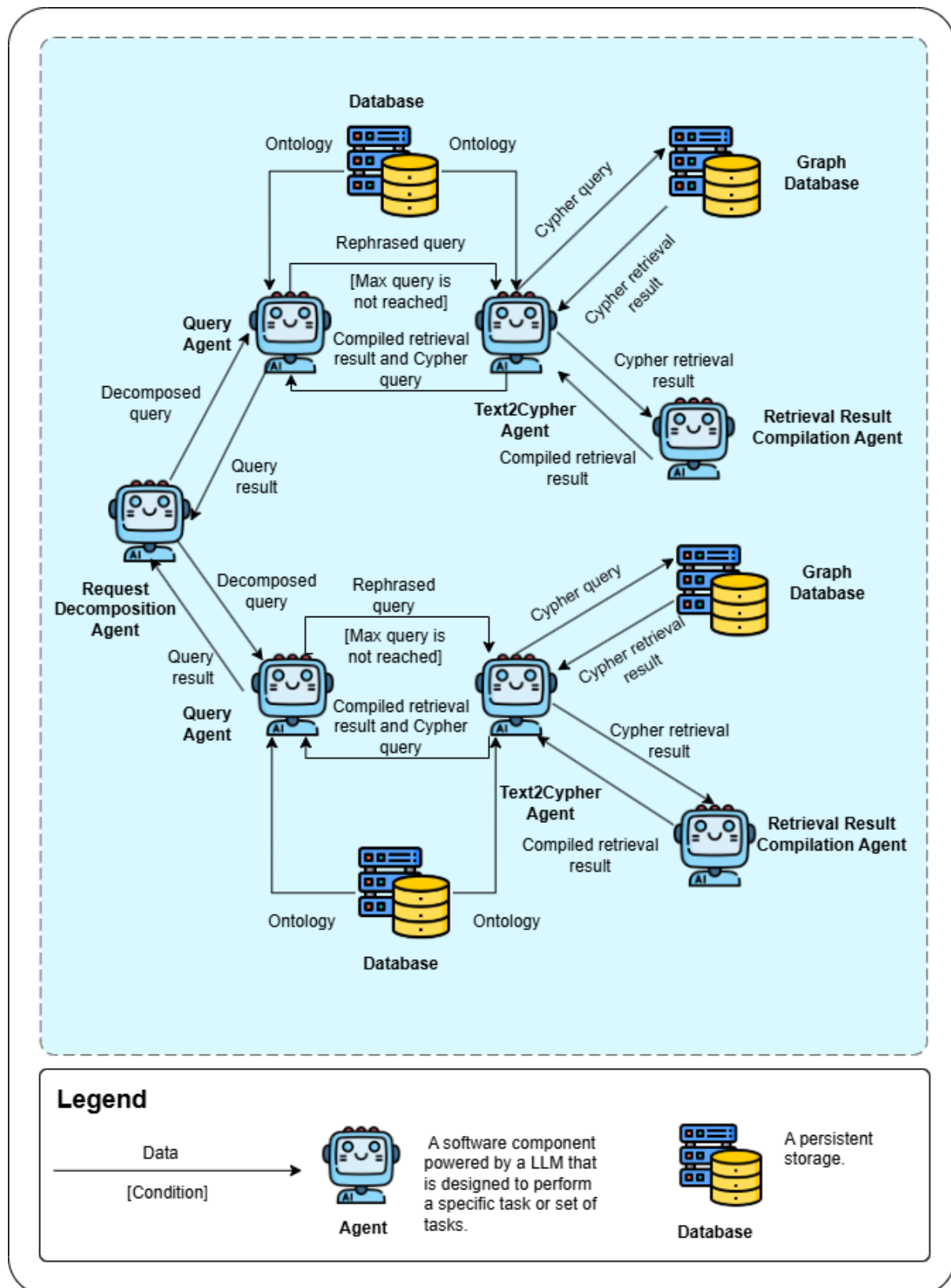


Figure 3.2 Graph RAG Agent Architecture Diagram



### 3.3 Use Case Diagram and Use Case Descriptions

#### 3.3.1 Use Case Diagram

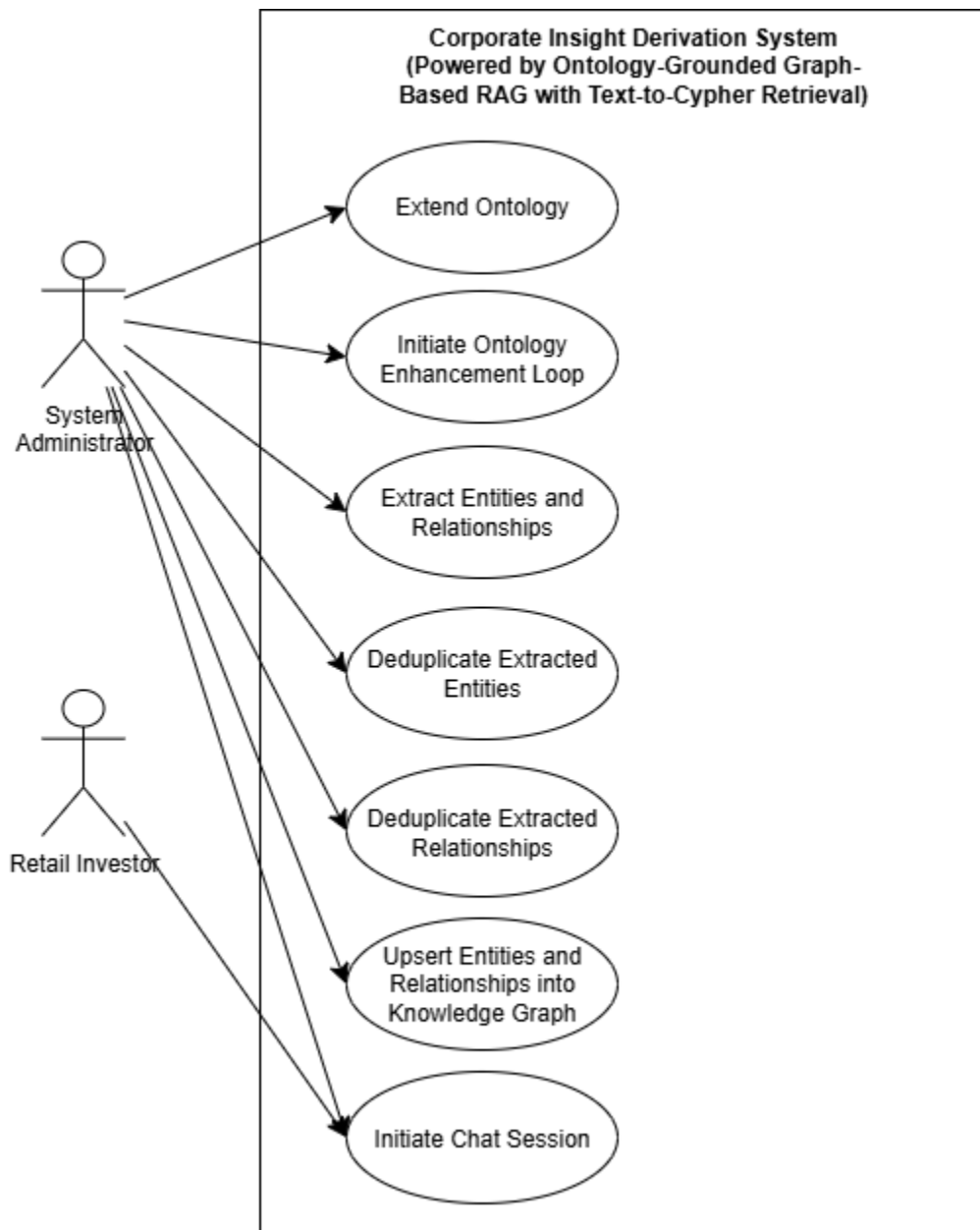
The allowable interactions between actors and the proposed corporate insight derivation module are illustrated in Figure 3.3. The system involves the following actors:

1. **Retail Investors**

Retail investors interact with the module through a web-based chatbot to obtain implicit insights that extend beyond surface-level text retrieval. For example, they might ask: “Which downstream clients of Company X are affected by the newly introduced Regulation Y that mandates ...?”.

2. **System Administrators**

System administrators are responsible for creating and extending ontologies, as well as initiating ontology enhancements. They use the constructed ontology to populate the knowledge graph by extracting entities and relationships from company disclosures, thereby ensuring that the system remains current and domain relevant. Furthermore, they perform deduplication on extracted entities and relationships. After deduplication, they can update or insert the entities and relationships into the knowledge graph.



*Figure 3.3 Use Case Diagram*

### 3.3.2 Use Case Descriptions

The use case descriptions for the use cases illustrated in Figure 3.3 are as follows.

#### UC001 Extend Ontology

<b>Use Case ID</b>	UC001	<b>Version</b>	1.0
<b>Use Case</b>	Extend Ontology		

<b>Purpose</b>	To extend an existing ontology or create a new one using data derived from provided documents and aligned with the defined ontology purpose.	
<b>Actor</b>	System administrator	
<b>Trigger</b>	The system administrator invokes the <code>extend_ontology()</code> function.	
<b>Pre-conditions</b>	The ontology purpose is clearly defined, and the Ontology Construction System is initialized with valid API keys.	
<b>Post-conditions</b>	The extended ontology is stored in the database.	
<b>Scenario Name</b>	<b>Step</b>	<b>Action</b>
<b>Main Flow</b>	1	The system administrator calls the <code>extend_ontology()</code> function, providing the company disclosure as input along with constraints for interpreting the source text (e.g., following an abbreviation list).
	2	The system retrieves the latest ontology (currently configured to always fetch the most recent version) from the database.
	3	The system invokes the Ontology Construction Agent with the source text, defined constraints, and the retrieved ontology.
	4	The Ontology Construction Agent returns new entities and relationships extracted from the source text, adhering to the specified constraints and ontology purpose.
	5	The system sets the “ <code>is_latest</code> ” attribute of the retrieved ontology to false.
	6	The system merges the newly extracted entities and relationships with the retrieved ontology, formatting it appropriately for upload. The “ <code>is_latest</code> ” attribute of this extended ontology is set to true.
	7	The system uploads the extended ontology to the database.
	8	The system logs the ontology update, including its version information.

<b>Alternate Flow A1</b> – <b>Empty response by the Ontology Construction Agent</b>	4.1	The system logs the reason provided by the Ontology Construction Agent for returning an empty response
	4.2	The system terminates the ontology extension process.
<b>Author</b>	Kam Chee Qin	

*Table 3.1 UC001 Extend Ontology Use Case Description*

### UC002 Initiate Ontology Enhancement Loop

<b>Use Case ID</b>	UC002	<b>Version</b>	1.0
<b>Use Case</b>	Initiate Ontology Enhancement Loop		
<b>Purpose</b>	To iteratively enhance the latest ontology through a feedback loop between the Ontology Evaluation Agent and the Ontology Enhancement Agent.		
<b>Actor</b>	System administrator		
<b>Trigger</b>	The system administrator invokes the <code>enhance_ontology_via_loop()</code> function.		
<b>Pre-conditions</b>	The ontology purpose is clearly defined, and the Ontology Construction System is initialized with valid API keys.		
<b>Post-conditions</b>	Each enhanced ontology generated during the iterations is stored in the database.		
<b>Scenario Name</b>	<b>Step</b>	<b>Action</b>	
<b>Main Flow</b>	1	The system administrator calls the <code>enhance_ontology_via_loop()</code> function.	
	2	The system invokes the Ontology Evaluation Agent with the latest ontology.	
	3	The Ontology Evaluation Agent evaluates the latest ontology against the ontology purpose and returns identified issues, their impact, and suggestions for improvement.	

	4	The system invokes the Ontology Enhancement Agent with the evaluation report and the latest ontology.
	5	The Ontology Enhancement Agent enhances the ontology based on the evaluation report, while adhering to the ontology purpose, and returns the enhanced ontology.
	6	The system stores the enhanced ontology in the database and marks it as the latest ontology.
	7	The system invokes the Ontology Evaluation Agent to re-evaluate the enhanced ontology.
	8	The ontology enhancement loop repeats from Step 3 until a predefined threshold—designed to increasingly constrain further improvements—is reached.
	9	The system logs the completion of the ontology enhancement loop.
<b>Alternate Flow A1</b> – <b>No issues identified</b>	3.1	The system logs the reason provided by the Ontology Evaluation Agent for returning an empty evaluation report.
	3.2	The system terminates the ontology enhancement loop.
<b>Alternate Flow A2</b> – <b>Threshold reached</b>	5.1	The system logs that the ontology enhancement loop has been completed.
	5.2	The system terminates the ontology enhancement loop
<b>Author</b>	Kam Chee Qin	

*Table 3.2 UC002 Initiate Ontology Enhancement Loop Use Case Description*

### UC003 Extract Entities and Relationships

<b>Use Case ID</b>	UC003	Version	1.0
<b>Use Case</b>	Extract Entities and Relationships		
<b>Purpose</b>	To extend the existing knowledge graph by extracting new entities and relationships, as defined in the ontology, from specific source documents.		
<b>Actor</b>	System administrator		

<b>Trigger</b>	The system administrator invokes the <code>extract_entities_relationships_from_unparsed_documents()</code> function.	
<b>Pre-conditions</b>	The company disclosure database is properly set up, and the Graph Construction System is initialized with valid API keys.	
<b>Post-conditions</b>	The newly extracted entities and relationships are stored in the database and marked with a “TO_BE_DEDUPLICATED” status.	
<b>Scenario Name</b>	<b>Step</b>	<b>Action</b>
<b>Main Flow</b>	1	The system administrator calls the <code>extract_entities_relationships_from_unparsed_documents()</code> function, specifying which documents from the company disclosure database should be used for extraction.
	2	The system retrieves the latest ontology, the specified unparsed documents, and any provided constraints for interpreting the disclosures (e.g., an abbreviation list).
	3	The system establishes an extraction sub-process for each document.
	4	The system logs the completion of entity and relationship extraction for the specified documents.
<b>Sub-flow S1 – Extract entities and relationships from a document</b>	3.1	The system partitions the ontology into multiple segments based on the preconfigured number of relationships per segment.
	3.2	For each ontology segment, the system invokes the Entity-Relationship Extraction Agent for the document.
	3.3	The system aggregates the entities and relationships extracted across all ontology segments and formats them for upload.
	3.4	The system uploads the extracted entities and relationships from the document into the database.
	3.5	The system updates the document’s “is_parsed” status to true.

<b>Alternate Flow A1</b> – <b>No unparsed documents</b>	2.1	The system logs that no unparsed documents are available for entity and relationship extraction.
	2.2	The system terminates the entity-relationship extraction process.
<b>Alternate Flow A2</b> – <b>Extraction failure in an ontology segment</b>	3.3.1	The system logs which Entity-Relationship Extraction Agent and ontology segment failed during processing.
	3.3.2	The system terminates the extraction process for the affected document.
<b>Author</b>	Kam Chee Qin	

*Table 3.3 UC003 Extract Entities and Relationships Use Case Description*

#### UC004 Deduplicate Extracted Entities

<b>Use Case ID</b>	UC004	<b>Version</b>	1.0
<b>Use Case</b>	Deduplicate Extracted Entities		
<b>Purpose</b>	To deduplicate extracted entities from company disclosures with the status “TO_BE_DEDUPLICATED” to ensure a compact knowledge graph.		
<b>Actor</b>	System administrator		
<b>Trigger</b>	The system administrator invokes the deduplicate_entities() function.		
<b>Pre-conditions</b>	The Graph Construction System is initialized with valid API keys.		
<b>Post-conditions</b>	Entities with the status “TO_BE_DEDUPLICATED” are updated to either “TO_BE_DELETED” or “TO_BE_UPSERTED_INTO_VECTOR_DB”, depending on the deduplication result. Entities must be transitioned to “TO_BE_UPSERTED_INTO_VECTOR_DB” before being set to “TO_BE_UPSERTED_INTO_GRAPH_DB”, ensuring they are first		

	updated or inserted into the vector database so the Text2Cypher Agent can correctly reference them in query composition.	
Scenario Name	Step	Action
<b>Main Flow</b>	1	The system administrator calls the deduplicate_entities() function, specifying the company of origin, batch size (number of entities per batch), maximum entity cache size, number of associated relationships to consider, and the similarity threshold for candidate filtering.
	2	The system initiates the entity cache size maintenance process.
	3	The system executes pending deduplication tasks from previous batches. These tasks include updating/inserting entities into the vector database or deleting entities from it.
	4	The system retrieves entities from the specified company with the status “TO_BE_DEDUPLICATED”, limited by the batch size.
	5	The system establishes a deduplication sub-process for each entity in the batch.
	6	The system repeats from Step 2 until no entities from the specified company remain with the status “TO_BE_DEDUPLICATED.”
	7	The system logs the completion of the entity deduplication process for the specified company.
<b>Sub-flow S1 – Maintain entity cache size</b>	2.1	The system checks the current entity cache size.
	2.2	If the current cache size $\geq$ (max cache size – batch size), the system adds delete tasks for the oldest entities equal to (current cache size – (max cache size – batch size)) into the pending deduplication task list.
<b>Sub-flow S2 – Resolve</b>	3.1	The system retrieves pending tasks involving updates, insertions, or deletions in the vector database.



<b>deduplication pending tasks</b>		
	3.2	The system executes the pending tasks.
<b>Sub-flow S3 – Deduplicate entity</b>	5.1	The system searches for the most similar entity in the cache that exceeds the similarity threshold.
	5.2	The system locks the matched entity to prevent concurrent access by other deduplication sub-processes.
	5.3	The system provides the attributes of both entities, along with their associated relationships (up to the configured limit), to the Entity Deduplication Agent.
	5.4	The Entity Deduplication Agent returns a decision on whether to merge the entities.
	5.5	The system enforces the decision: updates the entity status accordingly and adds the corresponding task to the pending deduplication list.
	5.6	The system releases the lock on the matched entity.
<b>Alternate Flow A1 – No entity to deduplicate</b>	4.1	The system terminates the entity deduplication process.
<b>Alternate Flow A2 – No similar entity found</b>	5.1.1	The system marks the entity as “TO_BE_UPSERTED_INTO_VECTOR_DB” and creates an insert task in the pending deduplication list.
<b>Alternate Flow A3 – Entity locked</b>	5.2.1	The system waits for the configured timeout.
	5.2.2	The system retries Step 5.2.1 until the entity is unlocked.
<b>Author</b>	Kam Chee Qin	

*Table 3.4 UC004 Deduplicate Entities Use Case Description*

#### UC005 Deduplicate Extracted Relationships

<b>Use Case ID</b>	UC005	<b>Version</b>	1.0
--------------------	-------	----------------	-----

<b>Use Case</b>	Deduplicate Extracted Relationships	
<b>Purpose</b>	To deduplicate extracted relationships from company disclosures with the status “TO_BE_DEDUPLICATED” to ensure a compact knowledge graph.	
<b>Actor</b>	System administrator	
<b>Trigger</b>	The system administrator invokes the deduplicate_relationships() function.	
<b>Pre-conditions</b>	The Graph Construction System is initialized with valid API keys.	
<b>Post-conditions</b>	Relationships with the status “TO_BE_DEDUPLICATED” are updated to either “TO_BE_DELETED” or “TO_BE_UPSERTED_INTO_GRAPH_DB”, depending on the deduplication result.	
<b>Scenario Name</b>	<b>Step</b>	<b>Action</b>
<b>Main Flow</b>	1	The system administrator calls the deduplicate_relationships() function, specifying the company of origin, batch size (number of relationships per batch), and maximum relationship cache size.
	2	The system initiates the relationship cache size maintenance process.
	3	The system retrieves relationships from the specified company with the status “TO_BE_DEDUPLICATED”, limited by the batch size.
	4	The system establishes a deduplication sub-process for each relationship in the batch.
	5	The system repeats from Step 2 until no relationships from the specified company remain with the status “TO_BE_DEDUPLICATED”.
	6	The system logs the completion of the relationship deduplication process for the specified company.
<b>Sub-flow S1 – Maintain</b>	2.1	The system checks the current relationship cache size.

<b>relationship cache size</b>		
	2.2	If the current cache size $\geq$ (max cache size – batch size), the system deletes the oldest relationships equal to (current cache size – (max cache size – batch size)).
<b>Sub-flow S2 – Deduplicate relationship</b>	4.1	The system searches for an existing relationship between the same source and target entities as the relationship to deduplicate.
	4.2	The system locks the matched relationship to prevent concurrent access by other deduplication sub-processes.
	4.3	The system provides the descriptions of both relationships to the Relationship Deduplication Agent, which merges the descriptions.
	4.4	The system reformats the matched relationship by merging the “valid_in” attributes of both relationships, the merged relationship description returned by the Relationship Deduplication Agent, and other metadata including the new status. It then uploads the relationship to the database and updates the status of the relationship to be deduplicated to "TO_BE_DELETED".
	4.5	The system releases the lock on the matched relationship.
<b>Alternate-flow A1 – No relationships to deduplicate</b>	3.1	The system terminates the relationship deduplication process.
<b>Alternate Flow A2 – No similar relationship found</b>	4.1.1	The system marks the relationship to deduplicate as “TO_BE_UPSERVED_INTO_GRAPH_DB”.
<b>Alternate Flow A3 – Relationship locked</b>	4.2.1	The system waits for the configured timeout.

	4.2.2	The system retries Step 4.2.1 until the relationship is unlocked.
<b>Author</b>	Kam Chee Qin	

*Table 3.5 UC005 Deduplicate Extracted Relationships Use Case Description*

### UC006 Upsert Entities and Relationships into Knowledge Graph

<b>Use Case ID</b>	UC006	<b>Version</b>	1.0
<b>Use Case</b>	Upsert Entities and Relationships into Knowledge Graph		
<b>Purpose</b>	To update or insert entities with the status “TO_BE_UPSERTEDED_INTO_VECTOR_DB” and relationships with the status “TO_BE_UPSERTEDED_INTO_GRAPH_DB” into the knowledge graph.		
<b>Actor</b>	System administrator		
<b>Trigger</b>	The system administrator invokes the upsert_entities_and_relationships_into_graph() function.		
<b>Pre-conditions</b>	The Graph Construction System is initialized with valid API keys.		
<b>Post-conditions</b>	Entities with the status “TO_BE_UPSERTEDED_INTO_VECTOR_DB” and relationships with the status “TO_BE_UPSERTEDED_INTO_GRAPH_DB” are successfully upserted and updated to “UPSERTEDED_INTO_GRAPH_DB”, remaining in the knowledge graph.		
<b>Scenario Name</b>	<b>Step</b>	<b>Action</b>	
<b>Main Flow</b>	1	The system administrator calls the upsert_entities_and_relationships_into_graph() function, specifying the company of origin.	
	2	The system fetches all entities with the status “TO_BE_UPSERTEDED_INTO_VECTOR_DB” originating from the specified company.	
	3	The system upserts all fetched entities from Step 2 into the vector database and marks them as “TO_BE_UPSERTEDED_INTO_GRAPH_DB”.	

	4	The system fetches all entities and relationships with the status “TO_BE_UPSERTED_INTO_GRAPH_DB” originating from the specified company.
	5	The system upserts all fetched entities and relationships from Step 4 into the graph database storing the knowledge graph, and updates their status to “UPSERTED_INTO_GRAPH_DB.”
	6	The system logs the completion of the upsert operation.
<b>Author</b>	Kam Chee Qin	

*Table 3.6 UC006 Upsert Entities and Relationships into Knowledge Graph Use Case Description*

#### UC007 Initiate Chat Session

<b>Use Case ID</b>	UC007	<b>Version</b>	1.0
<b>Use Case</b>	Initiate Chat Session		
<b>Purpose</b>	To enable retail investors and system administrators to interact with the corporate insight derivation module through a chatbot interface.		
<b>Actor</b>	Retail investor and system administrator		
<b>Trigger</b>	The retail investor or system administrator sends a message via the web-based chatbot.		
<b>Pre-conditions</b>	The Graph Retrieval System is initialized with valid API keys.		
<b>Post-conditions</b>	The retail investor or the system administrator receives a response to their request.		
<b>Scenario Name</b>	<b>Step</b>	<b>Action</b>	
<b>Main Flow</b>	1	The actor sends a message via the web-based chatbot, initiating interaction with the Graph Retrieval System.	
	2	The system evaluates the nature of the request.	
	3	The system calls the appropriate agent to retrieve information relevant to the request.	
	4	The system generates and returns a response to the actor based on the retrieved information.	

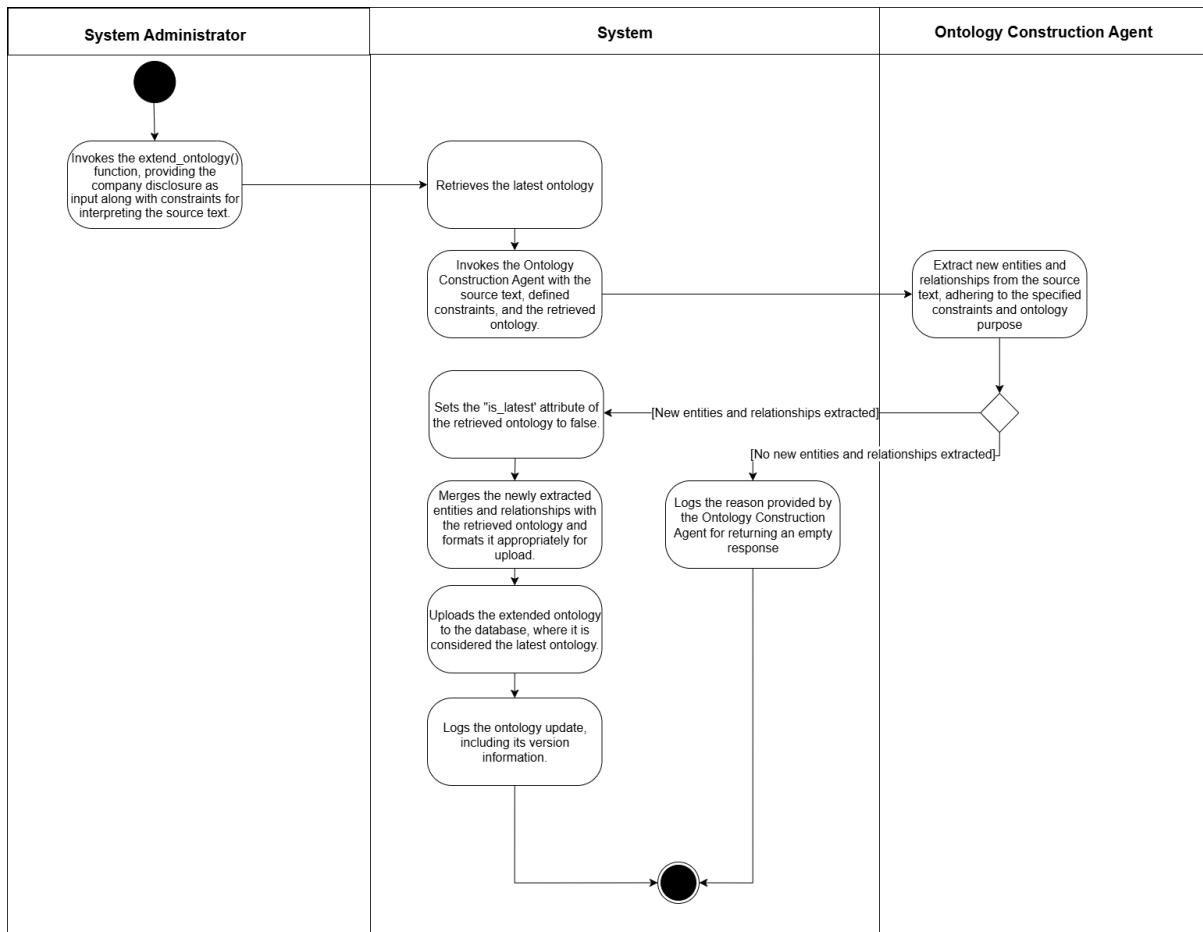
<b>Alternate Flow A1</b> – <b>User issues invalid request</b>	2.1	The system rejects the request and explains why it was rejected.
<b>Author</b>	Kam Chee Qin	

*Table 3.7 UC007 Initiate Chat Session Use Case Description*

### 3.4 Activity Diagram

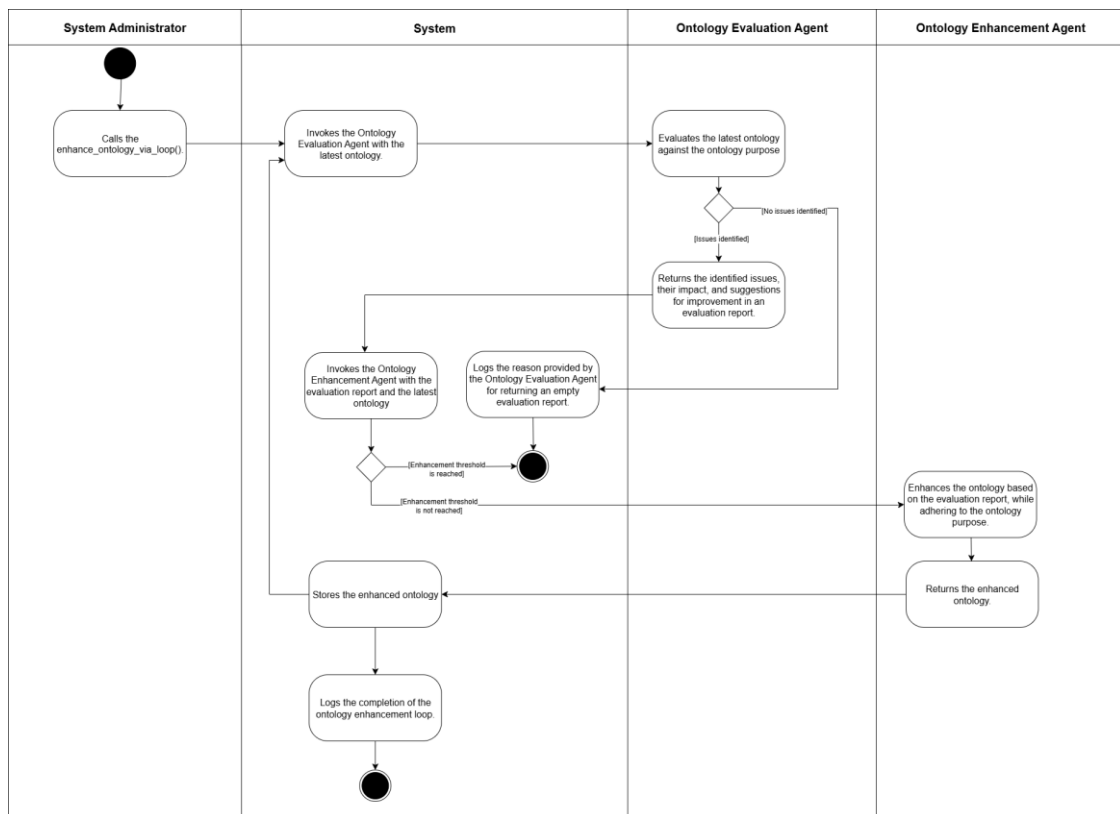
This section presents the activity diagrams that illustrate the key flows of the use cases highlighted in Section 3.3.

#### UC001 Extend Ontology



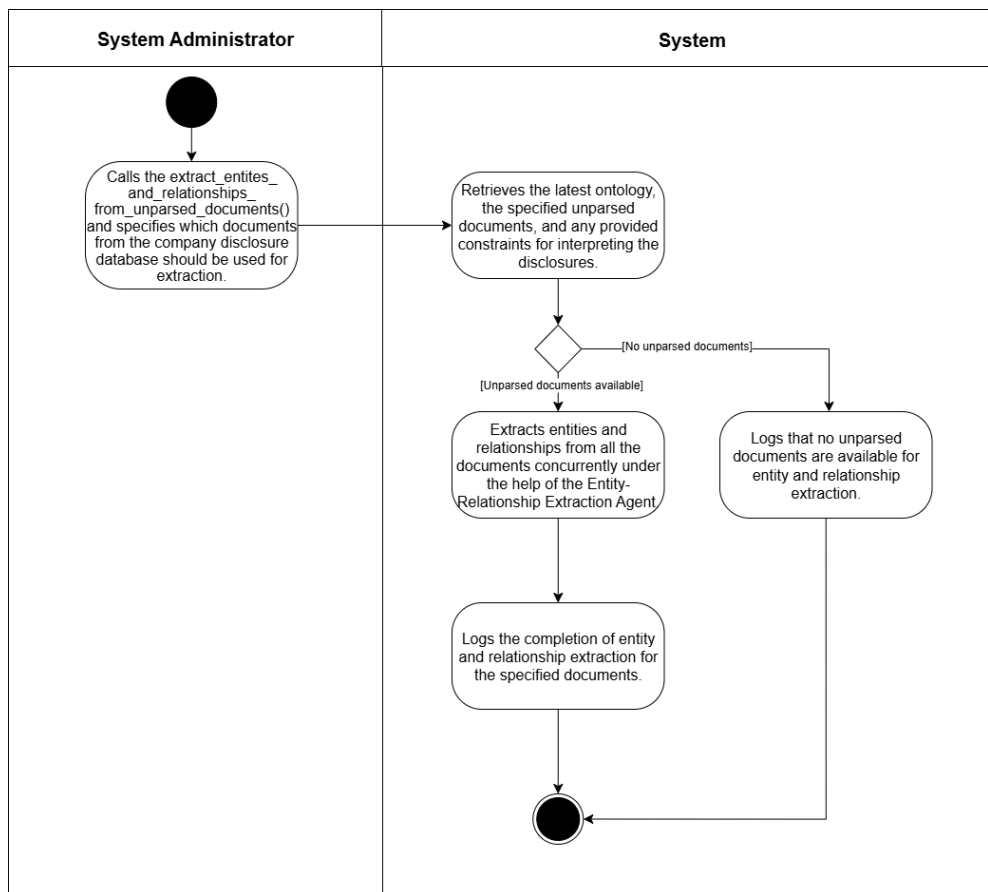
*Figure 3.4 UC001 Extend Ontology – Simplified Activity Diagram*

#### UC002 Initiate Ontology Enhancement Loop



*Figure 3.5 UC002 Initiate Ontology Enhancement Loop - Simplified Activity Diagram*

### UC003 Extract Entities and Relationships



*Figure 3.5 UC003 Extract Entities and Relationships - Simplified Activity Diagram*

#### UC004 Deduplicate Extracted Entities



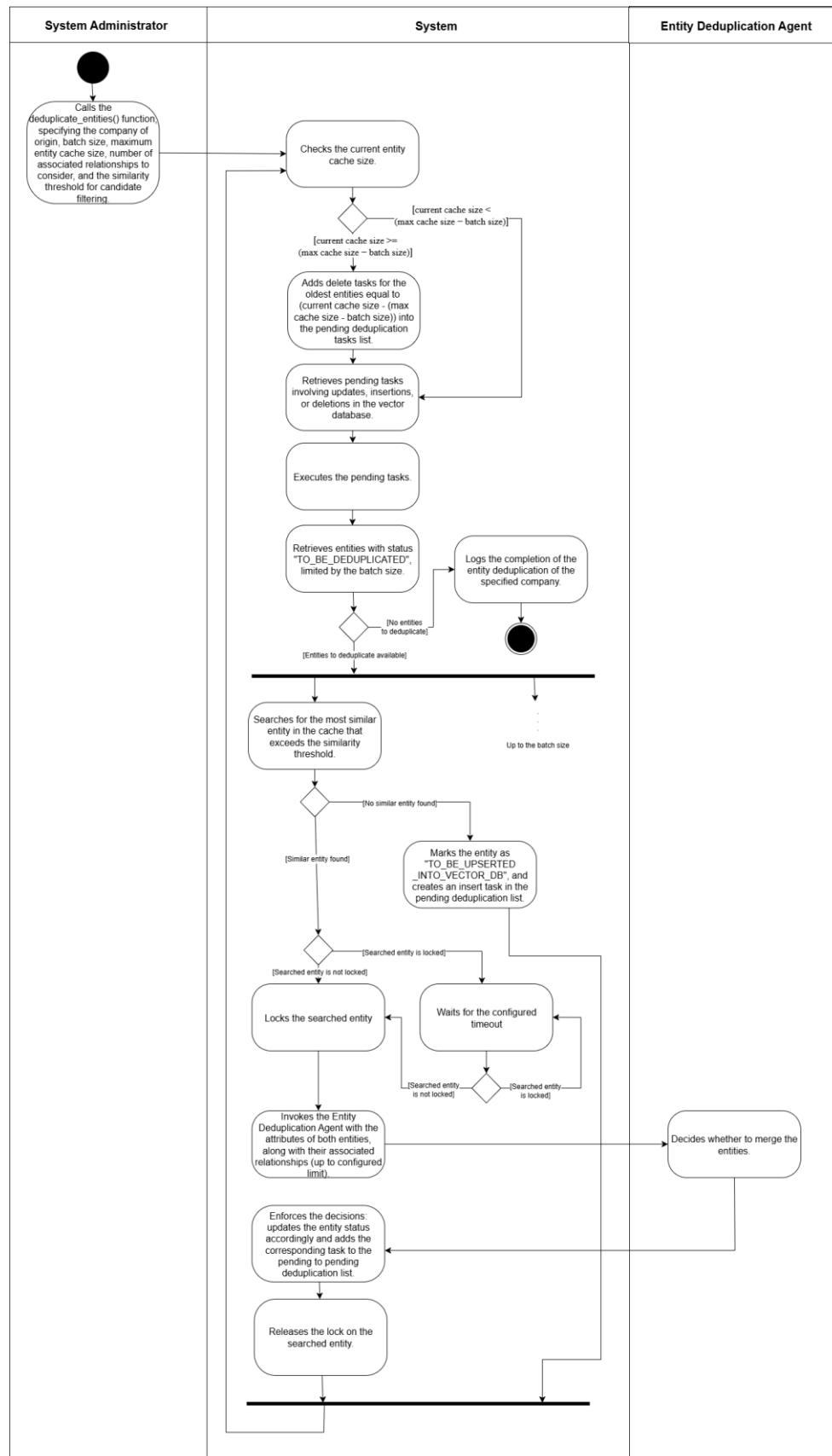


Figure 3.6 UC004 Deduplicate Extracted Entities - Simplified Activity Diagram

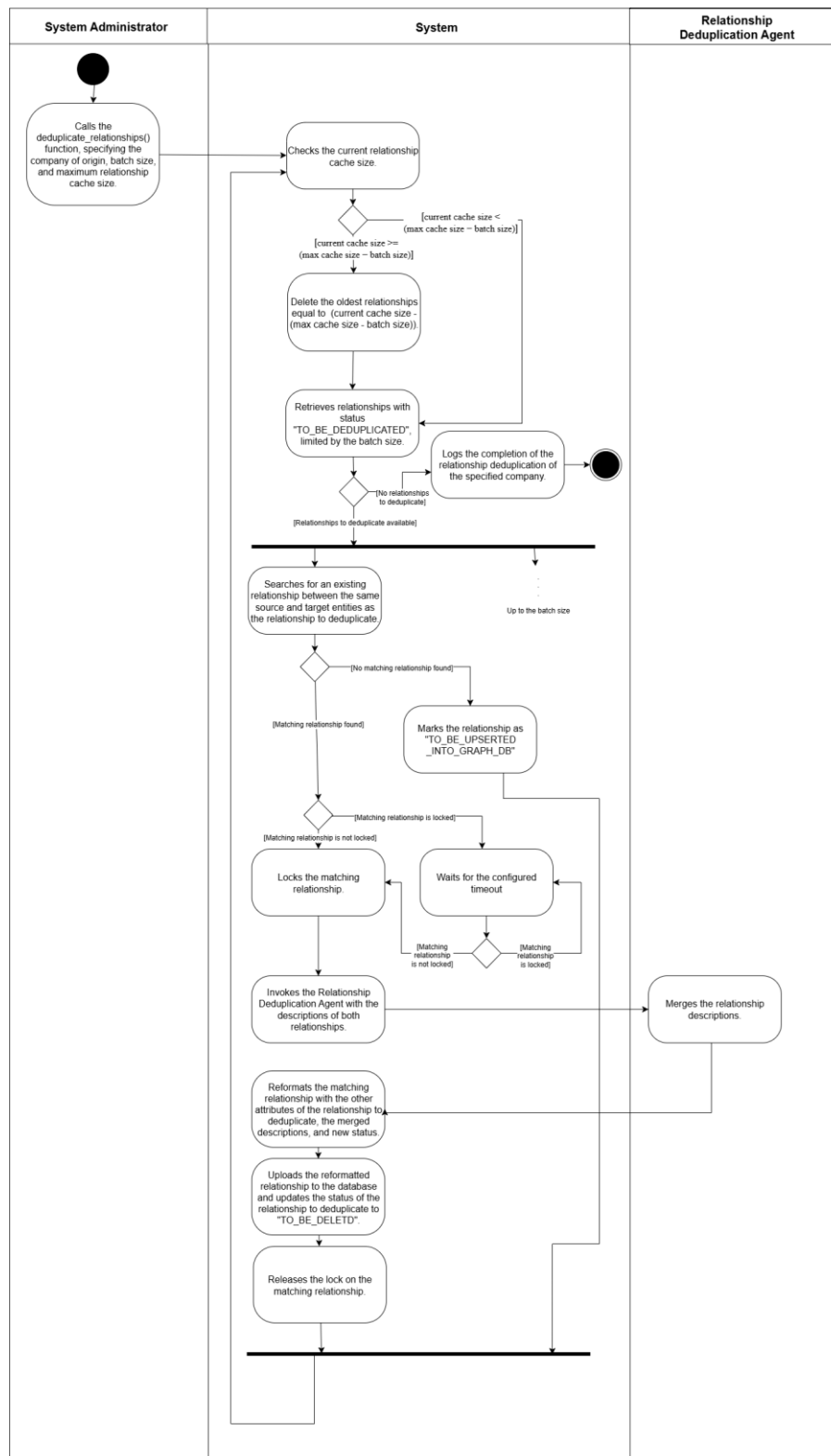
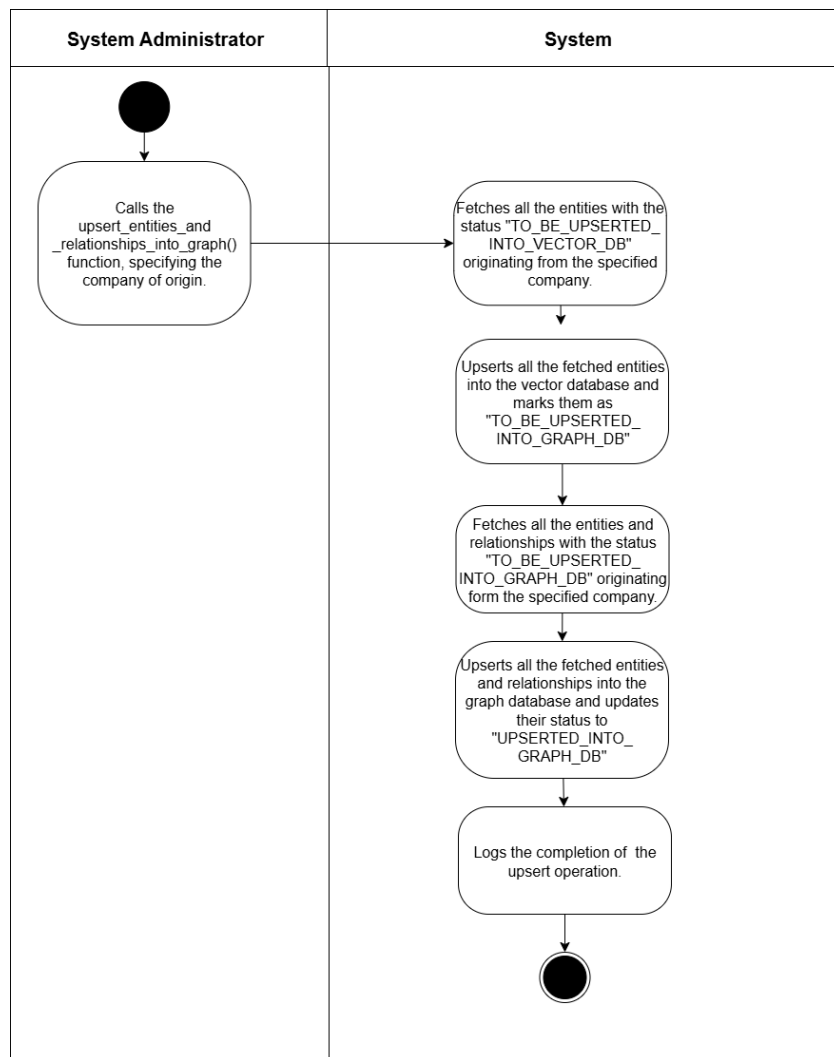
**UC005 Deduplicate Extracted Relationships**

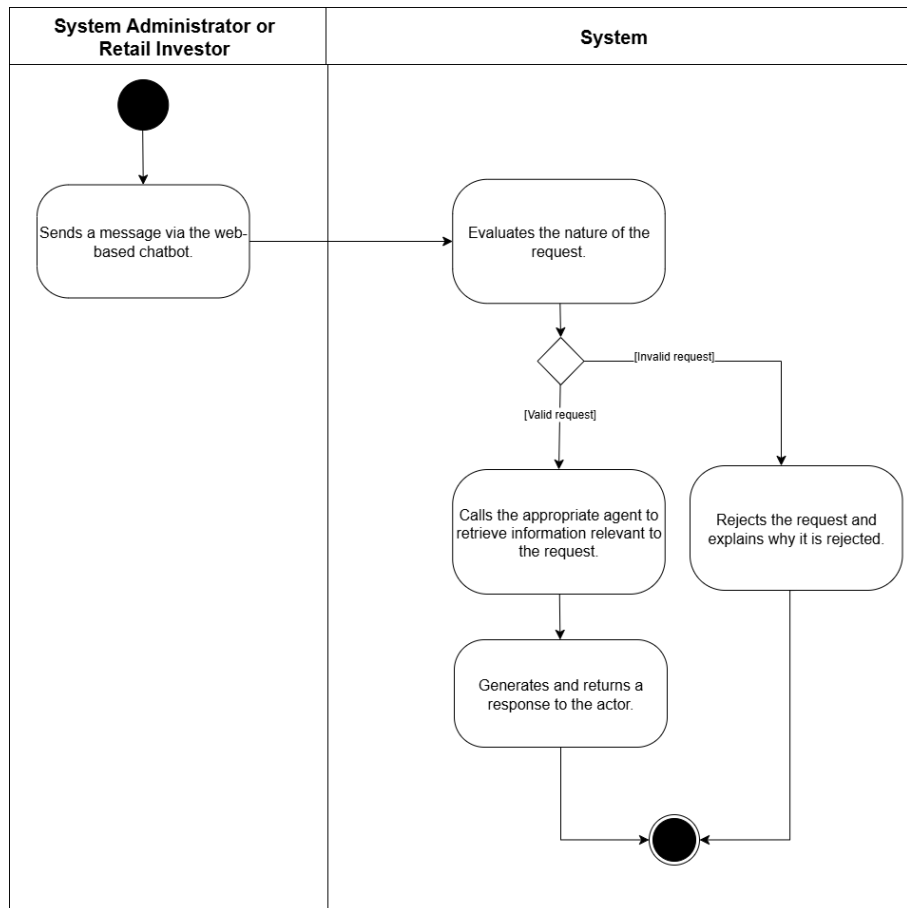
Figure 3.7 UC005 Deduplicate Extracted Relationships - Simplified Activity Diagram

**UC006 Upsert Entities and Relationships into Knowledge Graph**



*Figure 3.8 UC006 Upsert Entities and Relationships into Knowledge Graph - Simplified Activity Diagram*

### UC007 Initiate Chat Session



*Figure 3.9 UC007 Initiate Chat Session - Simplified Activity Diagram*

## Chapter 4 System Design

### 4.1 Overall System Design

This section presents a high-level overview of the proposed ontology-grounded, graph-based Retrieval-Augmented Generation (RAG) with text-to-Cypher retrieval. The system underpins the corporate insights derivation module, designed to assist retail investors in conducting fundamental analysis.

As illustrated in Figure 4.1, the proposed pipeline is organised into three sub-modules: the ontology construction module, the graph construction module, and the graph retrieval module. The design and operational details of each sub-module are discussed in Sections 4.2, 4.3, and 4.4, respectively.

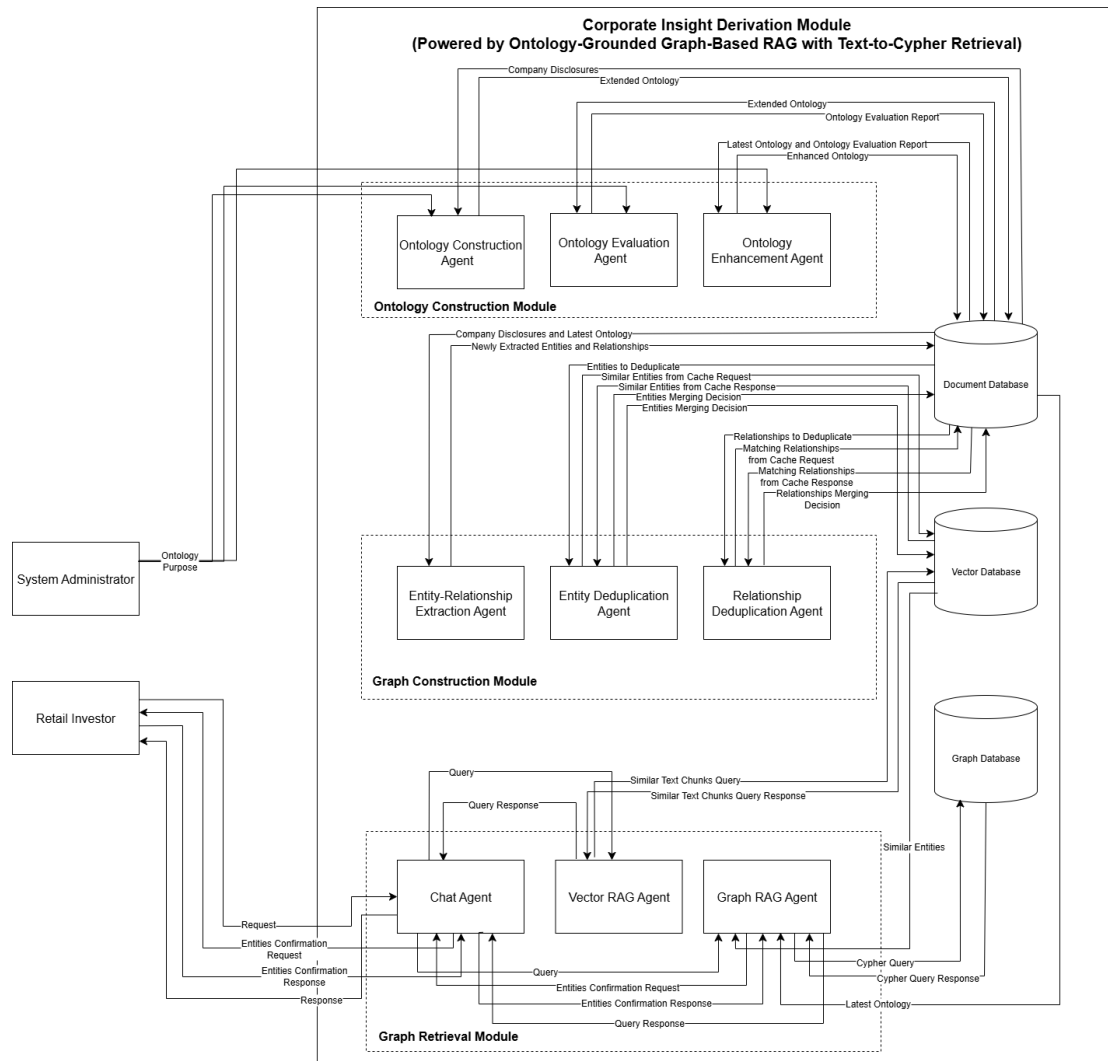


Figure 4.1 System Block Diagram of Proposed Module

Several important clarifications should be noted when interpreting Figure 4.1:

### 1. Company Disclosures

This term refers to the annual reports and prospectuses of companies listed on the Main Market and ACE Market of Bursa Malaysia. These documents are scraped, pre-processed, and stored in sections in plain-text format within the document database. This pipeline was developed by another student as part of a separate final-year project; therefore, its technical details are not included in this report. For the purposes of this work, it is assumed that the source documents required for ontology and graph construction are already available in the document database in a processable format.

### 2. Vector RAG Agent

This agent, which may be invoked by the Chat Agent when formulating responses to user queries, was also developed by another student in a separate final-year project. It is treated as a black box capable of generating responses to queries in a manner similar to the Graph RAG Agent. Accordingly, the details of its implementation are not covered in this work.

### 3. Types of Databases

Three types of databases are employed in this project, each serving distinct purposes:

- **Document Database** – A non-relational database that stores data in key–value pairs [17]. In this work, it is used to maintain outputs from the sub-modules, including ontologies, ontology evaluation reports, and entities and relationships extracted from source documents. It also serves as the cache for deduplication processes. All extracted entities and relationships are initially stored here before propagation to the vector and graph databases. This database thus acts as the single source of truth for the knowledge graph. The document database is adopted instead of a relational database, as the pipeline does not require relational joins.
- **Vector Database** – A database that indexes and stores vector embeddings to support similarity search [18]. It is employed to facilitate candidate filtering during entity deduplication and to enable precise entity referencing during Cypher query formulation.

- **Graph Database** – A database that represents data as nodes (entities), edges (relationships), and properties (attributes) [19]. It is utilised to construct and maintain the knowledge graph, enabling the derivation of implicit insights from explicitly modelled data.

### 4.2 Ontology Construction Module Design

This section outlines the key processes within the Ontology Construction Module. The module consists of three agents and two primary processes for ontology construction: the Ontology Extension process and the Ontology Refinement Loop process, as described in Sections 4.2.1 and 4.2.2, respectively.

#### 4.2.1 Ontology Extension

The ontology extension process is carried out by the Ontology Construction Agent. Given an ontology purpose, an optional existing ontology, a source text, and an optional source text constraint, the Ontology Construction Agent extracts entities and relationships that supplement the existing ontology while adhering to its design principles and the specified source text constraint. The extended ontology is then regarded as the latest version and stored in the document database.

The source text constraint refers to an additional requirement that the Ontology Construction Agent must follow when extracting entities and relationships from the source text. For example, it may involve interpreting certain abbreviations in a specific manner, as prospectuses published on Bursa Malaysia often include abbreviation lists such as the one illustrated in Figure 4.2. This mechanism helps prevent misinterpretation by the LLM underlying the Ontology Construction Agent.

DEFINITIONS	
The following definitions shall apply throughout this Prospectus unless the term is defined otherwise or the context otherwise requires:	
<b>Companies within our Group</b>	
ACSB	: Auto Count Sdn Bhd (Registration No.: 200601031841 (751600-A))
ACSPL	: Autocount (S) Pte Ltd (Registration No.: 201713604G)
ADB or Company	: Autocount Dotcom Berhad (Registration No.: 202201006885 (1452582-U))
ADB Group or Group	: ADB and its subsidiaries, namely ACSB, ACSPL, AOTGSB and ASSB, collectively
AOTGSB	: Autocount On The Go Sdn Bhd (Registration No.: 201601008185 (1179113-V))
ASSB	: Autocount Software Sdn Bhd (Registration No.: 202001018079 (1374399-V))
<b>General</b>	
ACE Market	: ACE Market of Bursa Securities
Acquisition of ACSB	: Acquisition by ADB of the entire equity interest of ACSB from the previous shareholders of ACSB i.e. CCP, CYT, Liaw Huah Seng, Lim Kim Seng, Lee Chern Siong, Tey Wah Sheng and Ng Boon Thye for a purchase consideration of RM8,007,509.00, satisfied via 456,914,998 ADB Shares, which was completed on 20 June 2022

*Figure 4.2 Example of Abbreviation List in a Prospectus*

Several key design decisions were incorporated into the prompt formulation powering this agent, as outlined below:

### 1. Ontology Design Principles

The Ontology Construction Agent is constrained by the following principles when extending the ontology:

- **Purpose-oriented** – Ensures adherence to the intended purpose of the ontology.
- **Compact** – Prevents the inclusion of redundant entities or relationships.
- **Relationship-driven** – Prioritises modelling interdependencies among entities while reducing the number of distinct entity types to be represented in the graph.
- **Unidirectional** – Simplifies query generation by the Text2Cypher Agent, thereby improving retrieval accuracy.
- **Non-taxonomic** – Avoids deep hierarchical structures, which would otherwise increase maintenance costs.

### 2. Ontology Format



The ontology is represented in natural language rather than in Web Ontology Language (OWL). This choice enables non-experts in OWL to adopt the proposed pipeline more readily and allows the Large Language Model (LLM) to leverage its general knowledge in interpreting the ontology. The approach trades rigid formalism for a degree of flexibility.

The attributes of each entity that must be defined by the Ontology Construction Agent in the ontology are:

- **Entity name** (string) – The entity name
- **LLM-guidance** (string) – The instruction provided to the Entity-Relationship Extraction Agent on how instances of the entity should be extracted
- **Examples** (list of strings) – Representative instances of the entity.

The attributes of each relationship that must be defined by the Ontology Construction Agent in the ontology are:

- **Relationship name** (string) – The relationship name.
- **Source** (string) – The source entity from which the relationship originates.
- **Target** (string) – The target entity to which the relationship points.
- **LLM-guidance** (string) – The instruction provided to the Entity-Relationship Extraction Agent on when to apply the relationship.
- **Examples** (list of strings) – Representative instances of the relationship.
- **Definition** (string) – A natural language description of the entity

### 4.2.2 Ontology Refinement Loop

The ontology refinement loop is carried out collaboratively by the Ontology Evaluation Agent and the Ontology Enhancement Agent. To prevent infinite iteration, the loop incorporates an increasing threshold, beyond which further enhancements are not performed.

Given an ontology and its intended purpose, the Ontology Evaluation Agent assesses the robustness of the ontology according to the same design principles applied in the Ontology Construction Agent. The evaluation is performed from two perspectives:

- **High-level perspective** – Examines the ontology as a whole, addressing the question: “Is this entity or relationship type necessary, or can its meaning be represented using an existing construct?”
- **Low-level perspective** – Focuses on the attributes of entities and relationships, addressing the question: “Would two annotators using this ontology interpret this entity or relationship consistently?”

The Ontology Evaluation Agent produces an evaluation report containing identified issues, their potential impact, and suggestions for improvement. This report is stored in the document database and later referenced by the Ontology Enhancement Agent.

Given an ontology, an ontology evaluation report, and the ontology purpose, the Ontology Enhancement Agent applies modifications to address the flagged issues. While adhering to the ontology design principles, the agent is not strictly bound to follow the exact suggestions provided by the Ontology Evaluation Agent, thereby reducing bias and improving enhancement outcomes. The resulting enhanced ontology is then stored in the document database and designated as the latest version.

For clarity, the Ontology Enhancement Agent does not perform modifications if the enhancement threshold has been reached, even if the ontology still contains unresolved issues. This threshold mechanism is illustrated in Figure 4.3.

```
- Iteration 1: always run (if problems exist).
- Iteration 2: always run (if problems exist).
- Iteration 3: run only if at least 2 problems remain.
- Iteration 4: run only if at least 3 problems remain.
- Iteration 5: run only if at least 4 problems remain.
- Iteration 6: run only if at least 5 problems remain.
- Iteration 7: run only if at least 6 problems remain (last iteration).
```

*Figure 4.3 Threshold Mechanism for Ontology Refinement Loop*

### 4.3 Graph Construction Module Design

This section outlines the key processes within the Graph Construction Module. The module comprises three agents and supports four primary processes: the Entity–Relationship Extraction process, the Entity Deduplication process, the Relationship Deduplication process,

and the Insertion of Entities and Relationships into the Knowledge Graph. These processes are detailed in Sections 4.3.1, 4.3.2, 4.3.3, and 4.3.4, respectively.

### 4.3.1 Entity-Relationship Extraction

The entity–relationship extraction process is performed by the Entity–Relationship Extraction Agent to extend the current knowledge graph. Given an ontology, a source text, an optional source text constraint (e.g., following an abbreviation list when interpreting the text), and the source text publication date, the agent extracts entity and relationship instances as defined in the ontology.

Several design decisions were incorporated into the prompt formulation for this agent:

#### 1. No Isolated Entity Rule

An entity is extracted only if it participates in at least one relationship, either as the source or target. Entities that do not connect to any relationship must not be extracted, even if they satisfy the entity definition in the ontology.

#### 2. Attributes to Define

The attributes required for each entity instance are:

- **ID** (string) – A temporary identifier (e.g., E1).
- **Name** (string) – A descriptive name extracted in accordance with the LLM-guidance specified in the ontology.
- **Type** (string) – A valid entity type defined in the ontology.
- **Description** (list of strings) – A comprehensive description of the entity written in reporting format, encoding temporal information relative to the source text publication date. The description follows the format: “... as of [source text publication date]”.

The attributes required for each relationship instance are:

- **ID** (string) – A temporary identifier (e.g., R1).
- **Source ID** (string) – The identifier of the source entity instance in the current extraction.
- **Target ID** (string) – The identifier of the target entity instance in the current extraction.
- **Type** (string) – A valid relationship type defined in the ontology.

- **Description** (list of strings) – A comprehensive description of the relationship written in reporting format, encoding temporal information relative to the source text publication date. The description follows the format: “... as of [source text publication date]”.
- **Valid In** (list of integers) – The years during which the relationship is valid. This may be inferred directly from the source text or derived from the publication date. If no temporal information is provided in the source text, the default value is the year of publication.

Since the ontology may contain a considerable number of entities and relationships, directly providing the entire ontology to the Entity–Relationship Extraction Agent can result in performance degradation. To address this, the ontology is partitioned into multiple segments based on relationships per segment. Similarly, a company disclosure is pre-segmented into multiple sections to avoid exceeding the context window of the LLM. Consequently, to improve performance, extraction is performed on a document-section–ontology-segment pair. These pairs are processed concurrently during extraction.

After extraction, the temporary identifiers are replaced with unique identifiers generated by the document database prior to insertion, ensuring global uniqueness. Both extracted entities and relationships are labelled as “TO\_BE\_DEDUPLICATED” so that they can be retrieved by the agents responsible for deduplication.

### 4.3.2 Entity Deduplication

The entity deduplication process is carried out by the Entity Deduplication Agent. All extracted entities must undergo deduplication before being inserted into the vector database and the graph database.

Several design decisions are implemented for entity deduplication:

#### 1. Cache-Based Deduplication

To balance between maintaining a compact graph and avoiding overly exhaustive extraction, a cache-based deduplication approach is employed. During deduplication, an entity searches for an existing entity with a semantically similar name in the cache of the company currently being processed. If such an entity exists, the attributes of both

entities, along with their associated relationships, are forwarded to the Entity–Deduplication Agent for a merging decision. If no such entity exists, the new entity is directly inserted into the entity cache. Thus, the degree of graph compactness can be controlled by adjusting the cache size. A smaller cache size tends to enforce more aggressive deduplication, resulting in a more compact graph. The cache size is maintained using the Least Recently Used (LRU) algorithm. Additionally, each company maintains its own entity cache.

## 2. Batch Deduplication

To improve efficiency, entity deduplication is performed in batches. The system administrator may configure the batch size to determine how many entities are deduplicated concurrently. However, since all entities in the same batch share the same entity cache, race conditions may occur if two entities attempt to access the same cached entry simultaneously. Therefore, a locking mechanism is required. Additionally, if two entities within the same batch are similar to each other but do not match any entity in the cache, both may be inserted, leading to a degree of duplication. This issue can be mitigated by adjusting the batch size. A smaller batch size reduces the likelihood of such cases, thereby producing a more compact graph.

## 3. Similarity Search for Pre-filtering

A similarity search is applied as a pre-filter to identify the most likely similar entities before they are passed to the Entity Deduplication Agent. A similarity threshold can be configured to reduce calls to the LLM, as entities with lower similarity scores are filtered out.

## 4. Merging Decision by LLM

The final merging decision is made by the Entity Deduplication Agent, which is powered by an LLM. The agent considers entity attributes, including descriptions and associated relationships (up to a configurable limit), to determine whether two entities should be merged. This addresses cases in which entities share similar names but are not identical. For instance, two entities of type *Person* may have the same name but represent different individuals. It also mitigates the issue of directly concatenating descriptions that convey the same information, as the agent determines the appropriate merged representation.

After deduplication, entities that are either newly inserted or result from merging are labelled “TO\_UPSERTEED\_INTO\_VECTOR\_DB”, whereas entities merged into others are labelled “TO\_BE\_DELETED.”

### 4.3.3 Relationship Deduplication

The relationship deduplication process is carried out by the Relationship Deduplication Agent. All extracted relationships must undergo deduplication before being inserted into the graph database.

Similar to entity deduplication, a cache-based and batch-performed deduplication approach facilitated by an LLM is employed for relationships. However, relationship deduplication is comparatively simpler, as a matching relationship is identified by checking whether a relationship already exists between the same source and target entities as the one being deduplicated. Once a matching relationship is found, their descriptions are merged by the Relationship–Deduplication Agent to avoid direct concatenation, which could otherwise result in redundant descriptions.

After deduplication, relationships that are either newly inserted or result from merging are labelled “TO\_UPSERTEED\_INTO\_GRAPH\_DB”, whereas relationships merged into others are labelled “TO\_BE\_DELETED.”

### 4.3.4 Insertion of Entities and Relationships into the Knowledge Graph

After the deduplication process, entities and relationships undergo different procedures before being inserted into the knowledge graph.

- **Insertion of Entities into the Knowledge Graph**

After deduplication, entities labelled “TO\_UPSERTEED\_INTO\_VECTOR\_DB” must first be inserted into the vector database. All entities are required to exist in the vector database, as Cypher queries demand precise entity references. When the Text2Cypher Agent formulates a Cypher query, it confirms which entities to use through a semantic search that verifies the existence of the entities referenced in the query. Once inserted into the vector database, these entities are relabelled

“TO\_UPSERTED\_INTO\_GRAPH\_DB” and subsequently inserted into the graph database together with their associated relationships labelled “TO\_UPSERTED\_INTO\_GRAPH\_DB.”

- **Insertion of Relationships into the Knowledge Graph**

After deduplication, relationships labelled “TO\_UPSERTED\_INTO\_GRAPH\_DB” are inserted into the graph database.

For clarity, both entity and relationship insertions into the databases—whether vector or graph—are designed to perform in batches. Items in the same batch are processed concurrently to improve performance. After insertion, entities and relationships are relabelled “UPSERTED\_INTO\_GRAPH\_DB.” This design supports idempotency, ensuring database integrity even in the event of unexpected failures.

### 4.4 Graph Retrieval Module Design

This section outlines the key processes within the Graph Retrieval Module. At a high level, the module comprises two agents and supports two primary processes, excluding the process performed by the Vector RAG Agent, which is addressed in a separate final year project. The two processes are: the Response Formulation process by the Chat Agent and Graph Retrieval Generation process by the Graph RAG Agent. These processes are described in Sections 4.4.1 and 4.4.2, respectively.

#### 4.4.1 Response Formulation by the Chat Agent

The Chat Agent is responsible for interacting with the user through a web-based chatbot interface. Its design centres on four primary tasks: Interaction, Tool Use, Retrieval Evaluation, and Response Generation.

##### 1. Interaction

The Chat Agent functions as the primary conversational interface with the user. Due to the open-ended nature of chatbot interactions, it is configured to reject requests that are either not information-related or unrelated to Malaysian listed companies. In such cases, the Chat Agent must provide a clear explanation for the rejection. To enhance usability, the Chat Agent maintains conversational memory within the current session, thereby enabling a more natural and user-friendly interaction.

### 2. Tool Use (Agent Call)

At a high level, the Chat Agent has access to two retrieval tools: the Vector RAG Agent and the Graph RAG Agent. Internally, however, it operates with three distinct tools. The Chat Agent is autonomous, selecting which tool to invoke based on the user's request and the description of each tool.

- **Entity Validation Tool**

This hidden auxiliary tool is always invoked before calling the Graph RAG Agent. Its purpose is to validate the entities mentioned in the user's request against actual entities stored in the vector database. Since the Text2Cypher Agent requires precise entity references when formulating Cypher queries, this validation step ensures accuracy.

After validation, the Chat Agent confirms with the user which entities are intended. As a result, all queries forwarded to the Graph RAG Agent reference only the correct entities present in the vector database, which is aligned with the graph database.

- **Vector RAG Agent**

This agent handles simple or direct requests and supplements the Graph RAG Agent by retrieving information that may not be modelled in the knowledge graph. Using the Vector RAG Agent avoids unnecessary overhead when the Graph RAG Agent would be excessive for the query. The detailed implementation of this agent is not described here, as it is addressed in a separate final year project.

- **Graph RAG Agent**

This agent derives implicit insights from explicit graph data. Its design and processes are discussed in Section 4.4.2.

### 3. Retrieval Evaluation

The Chat Agent evaluates all retrieval responses returned by the Vector RAG Agent and the Graph RAG Agent. The evaluation is based on two criteria:

- **Relevance** – whether the retrieved response matches the user's request.
- **Decision-readiness** – whether the response provides sufficient information to support decision-making.



If necessary, the Chat Agent may issue additional calls to the same or different retrieval agents, provided that the maximum configured number of tool calls has not been exceeded.

### 4. Response Generation

Finally, the Chat Agent generates a response to the user when (i) the retrieval result is satisfactory, (ii) the maximum tool call limit is reached, or (iii) the user's request is deemed invalid. The response is presented in the most appropriate format, depending on the type and structure of the retrieved data.

#### 4.4.2 Graph Retrieval Generation by the Graph RAG Agent

This section describes the Graph RAG Agent, which facilitates retail investors in deriving implicit insights from explicit data to support fundamental analysis. Internally, the agent comprises a set of sub-agents that work collaboratively to generate a response to the query passed by the Chat Agent, based on information retrieved from the knowledge graph. The Graph RAG Agent consists of four main sub-agents: the Request Decomposition Agent, the Query Agent, the Text2Cypher Agent, and the Retrieval Result Compilation Agent.

##### 1. Request Decomposition Agent

The Request Decomposition Agent improves retrieval performance by decomposing queries received from the Chat Agent into multiple independent sub-queries, where possible, before forwarding them to the Query Agent. If decomposition is not feasible, the original query is passed intact. This design not only enhances efficiency by enabling concurrent execution but also allows the Query Agent to focus on narrower, more manageable tasks. The maximum number of sub-queries is predetermined.

##### 2. Query Agent

The Query Agent is responsible for three key tasks:

- **Query Rephrasing** – It converts the input query (or sub-query) into an ontology-aware natural language query. This allows the Text2Cypher Agent to focus solely on Cypher conversion without ambiguity regarding the intended meaning.
- **Retrieval Result Evaluation** – It evaluates whether re-retrieval is necessary based on two criteria:
  - **Relevance**: whether the retrieved response aligns with the user's request.

- Decision-readiness: whether the response provides sufficient information to support decision-making.

If the Cypher query generated by the Text2Cypher Agent yields an empty result, the Query Agent must verify whether the Cypher query was formulated correctly or whether the ontology-aware query was misunderstood.

- **Response Generation** – It produces a final response to the query passed by the Request Decomposition Agent. This response preserves all relevant details and temporal information from the retrieval results, which are pre-processed by the Retrieval Result Compilation Agent.

The Query Agent may perform multiple re-query attempts until the predefined query limit is reached.

### 3. Text2Cypher Agent

The Text2Cypher Agent is dedicated to converting ontology-aware, rephrased queries from the Query Agent into valid Cypher queries for execution on the knowledge graph.

### 4. Retrieval Result Compilation Agent

The Retrieval Result Compilation Agent addresses issues of redundancy and non-readable output in raw Cypher query results, which can hinder evaluation by the Query Agent and increase computational costs if the results are large. This agent compiles and restructures the retrieved information, preserving relevant factual details and temporal information while ensuring logical readability. To improve cost-effectiveness, this component is powered by a lighter-weight LLM, as its task complexity is relatively low.

## Chapter 5 System Implementation

### 5.1 Hardware Setup

This section outlines the hardware setup of the project. The only hardware utilised in this project is a laptop. Accordingly, the specifications presented in Table 5.1 refer exclusively to the laptop employed.

Description	Specifications
Visual	HUAWEI MateBook D14
Processor	AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx 2.30 GHz
Operating System	Windows 11
Graphic	AMD Radeon (TM) RX Vega 10 Graphics
Memory	8 GB DDR4
Storage	512 GB PCIe NVMe SSD

*Table 5.1 Hardware Specification (Laptop)*

### 5.2 Software Setup

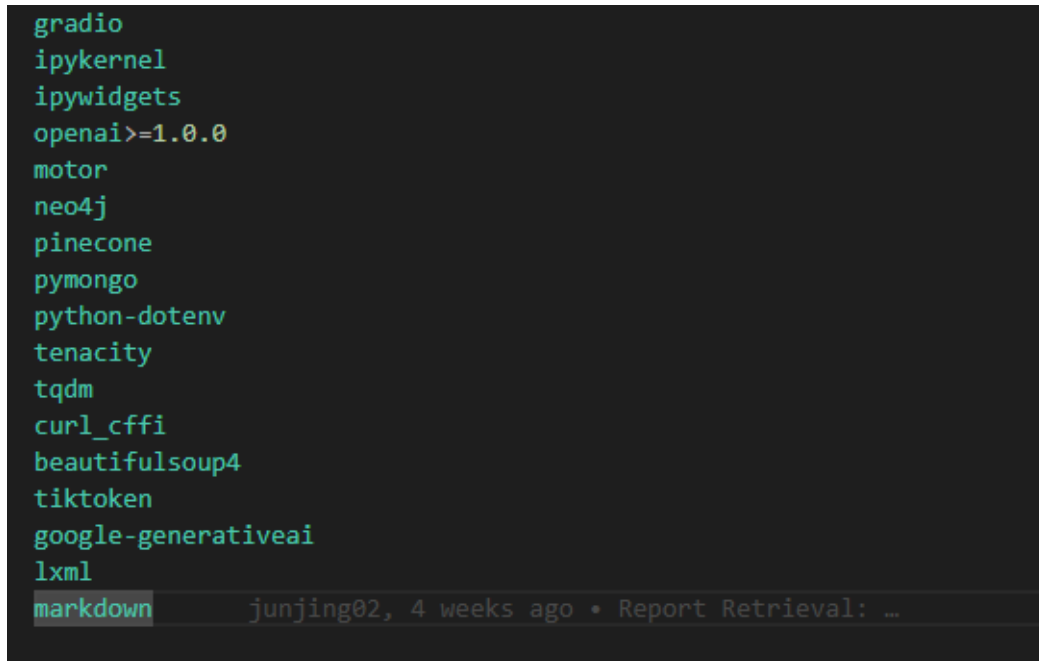
This section outlines the software setup of the project. Table 5.2 lists the required software together with their respective roles in the project.

Name	Application in Project
Visual Studio Code 1.99.3	Integrated Development Environment (IDE) for development
Anaconda Prompt 24.5.0	Environment management tool for the project
MongoDB	Serves as the document database for storing ontology, ontology evaluation report, and etc.
Pinecone	Serves as the vector storage for the entity cache and entities.
Neo4j	Serves as the graph storage for entities and relationships
Sourcetree	Local version control client
GitHub	Remote version control

OpenAI API	Enables access to OpenAI LLMs and text embedding model
Python 3.10	Programming language for development
Gradio	Chatbot interface
Hugging Face Spaces	Deployment platform for Gradio Chatbot

*Table 5.2 Software specifications*

In addition, the Python libraries required for the project are presented in Figure 5.1.



```

gradio
ipykernel
ipywidgets
openai>=1.0.0
motor
neo4j
pinecone
pymongo
python-dotenv
tenacity
tqdm
curl_cffi
beautifulsoup4
tiktoken
google-generativeai
lxml
markdown

```

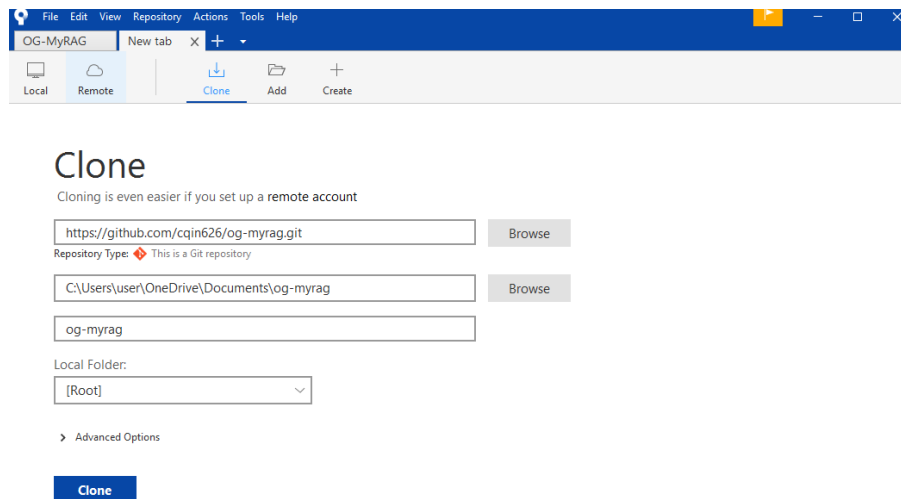
*Figure 5.1 Python Libraries Required*

### 5.3 Setting and Configuration

The source code of the project is publicly available on GitHub at: <https://github.com/cqin626/og-myrag>

To set up the project, the following steps should be performed:

1. Clone the repository to a local directory as shown in Figure 5.2



*Figure 5.2 Cloning the Repository using SourceTree*

2. Create a new environment via Anaconda Prompt with Python version  $\geq 3.10$  as shown in Figure 5.3

```

Anaconda Prompt
(base) C:\Users\user>conda create -n testenv python=3.10
Retrieving notices: ...working... done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\user\anaconda3\envs\testenv

added / updated specs:
  - python=3.10
  
```

*Figure 5.3 Creating a New Environment in Anaconda Prompt*

3. Activate the environment.

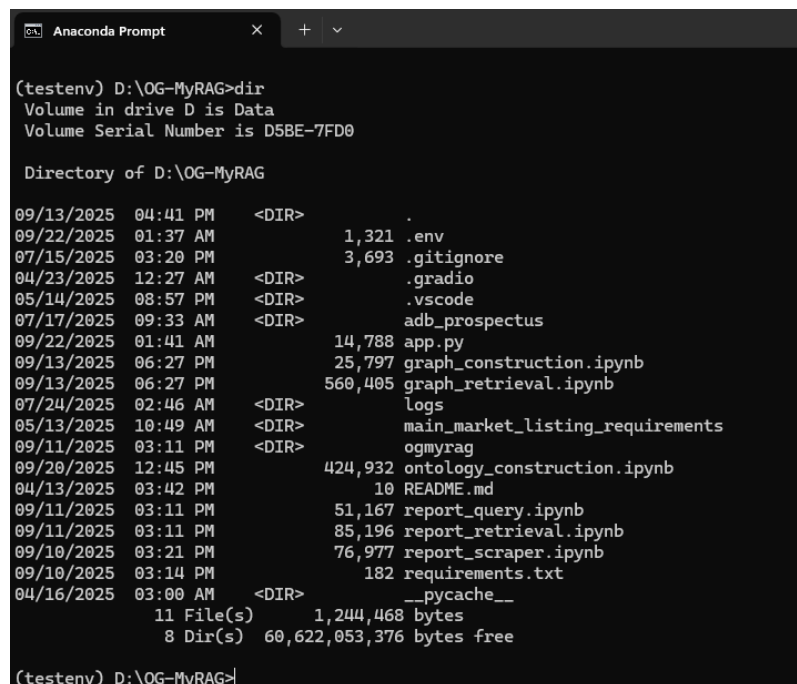
```

(base) C:\Users\user>conda activate testenv

(testenv) C:\Users\user>
  
```

*Figure 5.4 Activating the Created Environment in Anaconda Prompt*

4. Navigate to the directory where the repository was cloned.



```

Anaconda Prompt
(testenv) D:\OG-MyRAG>dir
Volume in drive D is Data
Volume Serial Number is D5BE-7FD0

Directory of D:\OG-MyRAG

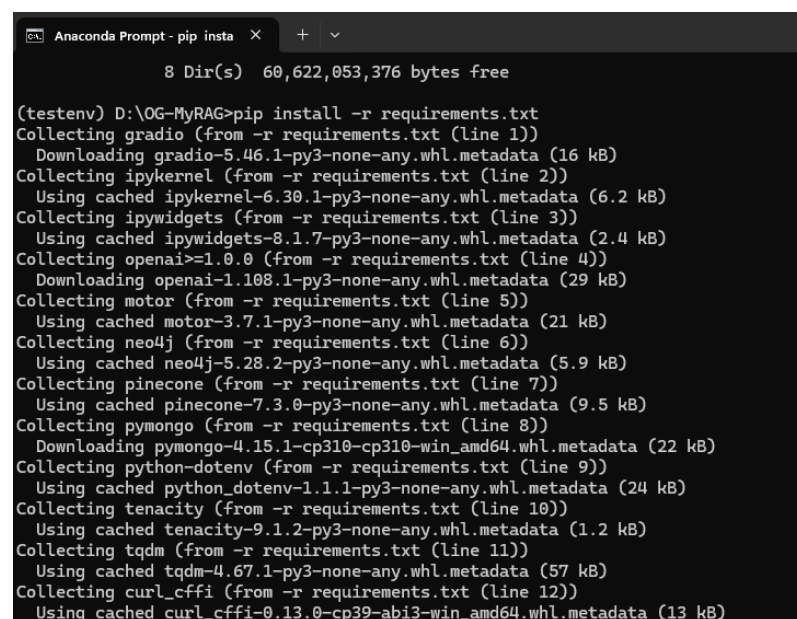
09/13/2025  04:41 PM  <DIR>          .
09/22/2025  01:37 AM             1,321 .env
07/15/2025  03:20 PM             3,693 .gitignore
04/23/2025  12:27 AM  <DIR>          .gradio
05/14/2025  08:57 PM  <DIR>          .vscode
07/17/2025  09:33 AM  <DIR>          adb_prospectus
09/22/2025  01:41 AM             14,788 app.py
09/13/2025  06:27 PM             25,797 graph_construction.ipynb
09/13/2025  06:27 PM            560,405 graph_retrieval.ipynb
07/24/2025  02:46 AM  <DIR>          logs
05/13/2025  10:49 AM  <DIR>          main_market_listing_requirements
09/11/2025  03:11 PM  <DIR>          ogmyrag
09/20/2025  12:45 PM            424,932 ontology_construction.ipynb
04/13/2025  03:42 PM              10 README.md
09/11/2025  03:11 PM            51,167 report_query.ipynb
09/11/2025  03:11 PM            85,196 report_retrieval.ipynb
09/10/2025  03:21 PM            76,977 report_scraper.ipynb
09/10/2025  03:14 PM              182 requirements.txt
04/16/2025  03:00 AM  <DIR>          __pycache__
               11 File(s)          1,244,468 bytes
                8 Dir(s)        60,622,053,376 bytes free

(testenv) D:\OG-MyRAG>

```

Figure 5.5 Navigating to the Cloned Repository in Anaconda Prompt

5. Install the required Python libraries listed in Fig. 5.1 using the requirements.txt file.



```

Anaconda Prompt - pip insta
      8 Dir(s)  60,622,053,376 bytes free

(testenv) D:\OG-MyRAG>pip install -r requirements.txt
Collecting gradio (from -r requirements.txt (line 1))
  Downloading gradio-5.46.1-py3-none-any.whl.metadata (16 kB)
Collecting ipykernel (from -r requirements.txt (line 2))
  Using cached ipykernel-6.30.1-py3-none-any.whl.metadata (6.2 kB)
Collecting ipywidgets (from -r requirements.txt (line 3))
  Using cached ipywidgets-8.1.7-py3-none-any.whl.metadata (2.4 kB)
Collecting openai>=1.0.0 (from -r requirements.txt (line 4))
  Downloading openai-1.108.1-py3-none-any.whl.metadata (29 kB)
Collecting motor (from -r requirements.txt (line 5))
  Using cached motor-3.7.1-py3-none-any.whl.metadata (21 kB)
Collecting neo4j (from -r requirements.txt (line 6))
  Using cached neo4j-5.28.2-py3-none-any.whl.metadata (5.9 kB)
Collecting pinecone (from -r requirements.txt (line 7))
  Using cached pinecone-7.3.0-py3-none-any.whl.metadata (9.5 kB)
Collecting pymongo (from -r requirements.txt (line 8))
  Downloading pymongo-4.15.1-cp310-cp310-win_amd64.whl.metadata (22 kB)
Collecting python-dotenv (from -r requirements.txt (line 9))
  Using cached python_dotenv-1.1.1-py3-none-any.whl.metadata (24 kB)
Collecting tenacity (from -r requirements.txt (line 10))
  Using cached tenacity-9.1.2-py3-none-any.whl.metadata (1.2 kB)
Collecting tqdm (from -r requirements.txt (line 11))
  Using cached tqdm-4.67.1-py3-none-any.whl.metadata (57 kB)
Collecting curl_cffi (from -r requirements.txt (line 12))
  Using cached curl_cffi-0.13.0-cp39-abi3-win_amd64.whl.metadata (13 kB)

```

Figure 5.6 Installing the Python Libraries in Anaconda Prompt

6. Configure the necessary API keys

```

# For access to OpenAI
OPENAI_API_KEY=

# For document database
MONGO_DB_URI=

# For document database (company disclosures)
MONGO_DB_URI_REPORTS=

# For graph database
NEO4J_URI=
NEO4J_USERNAME=
NEO4J_PASSWORD=

# For vector database (entities)
PINECONE_ENTITIES_API_KEY=
PINECONE_ENTITIES_ENVIRONMENT=
PINECONE_ENTITIES_CLOUD=
PINECONE_ENTITIES_METRIC=
PINECONE_ENTITIES_DIMENSIONS=

# For vector database (entity cache)
PINECONE_CACHE_API_KEY=
PINECONE_CACHE_ENVIRONMENT=
PINECONE_CACHE_CLOUD=
PINECONE_CACHE_METRIC=
PINECONE_CACHE_DIMENSIONS=

# For vector database (Vector RAG)
PINECONE_API_KEY_JJ =

```

Figure 5.7 Configuring the Necessary API Keys in .env

- Interact with any of the sub-modules of the proposed system using the sample Jupyter Notebooks provided.

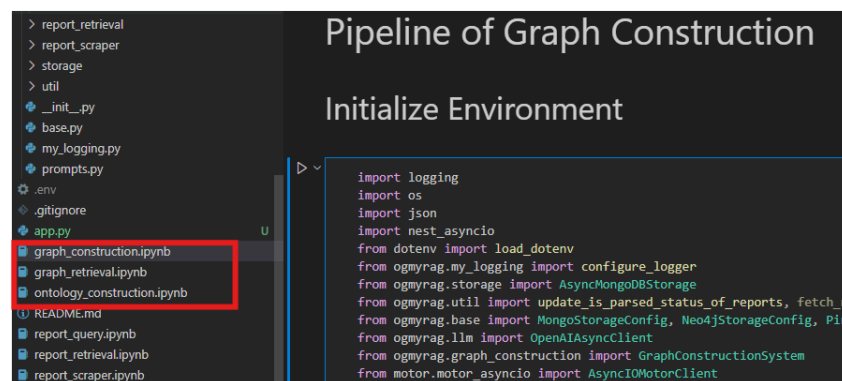


Figure 5.8 Selecting a Jupyter Notebook

## 5.4 System Operation

This section outlines the operational aspects of the proposed corporate insight derivation module, which is powered by the ontology-grounded graph-based RAG with text-to-Cypher retrieval. The detailed workings of its core sub-modules—the Ontology Construction Module, the Graph Construction Module, and the Graph Retrieval Module—are presented in Sections 5.4.1, 5.4.2, and 5.4.3, respectively. At the current stage, interaction with the sub-modules is carried out through Jupyter Notebooks or Python scripts, as the proposed RAG pipeline is

designed to function as a Python library. Additionally, the Graph Retrieval Module is accessible via a Gradio chatbot hosted on Hugging Face Spaces.

### 5.4.1 Ontology Construction

To begin ontology construction, the user must initialize the Ontology Construction Module by specifying the appropriate database connection variables, defining the ontology purpose, and selecting the LLMs to be used for the Ontology Construction Agent, the Ontology Evaluation Agent, and the Ontology Enhancement Agent.

An ontology construction process is considered complete after two key stages are executed: the Ontology Extension and the Ontology Enhancement Loop. Each modification to the ontology—whether through extension or enhancement—is treated as a PATCH version update to the existing ontology under the current design. Ontology versioning follows the convention: MAJOR.MINOR. PATCH.

```

Initialize Variables for Database Connection

ontology_config: MongoStorageConfig = {
    'database_name': 'ogmyrag',
    'collection_name': 'ontology_test'
}

ontology_evaluation_config: MongoStorageConfig = {
    'database_name': 'ogmyrag',
    'collection_name': 'ontology_evaluation_test'
}

company_disclosures_config: MongoStorageConfig = {
    'database_name': 'FYP',
    'collection_name': 'company_disclosures'
}

constraints_config: MongoStorageConfig = {
    'database_name': 'FYP',
    'collection_name': 'constraints'
}

mongo_client_onto = AsyncIOMotorClient(
    mongo_db_uri,
    serverSelectionTimeoutMS=5000,
)

mongo_client_reports = AsyncIOMotorClient(
    mongo_db_uri_reports,
    serverSelectionTimeoutMS=5000,
)

async_mongo_storage_reports = AsyncMongoDBStorage(client=mongo_client_reports)

```

*Figure 5.9 Initializing Variables for Database Connection*



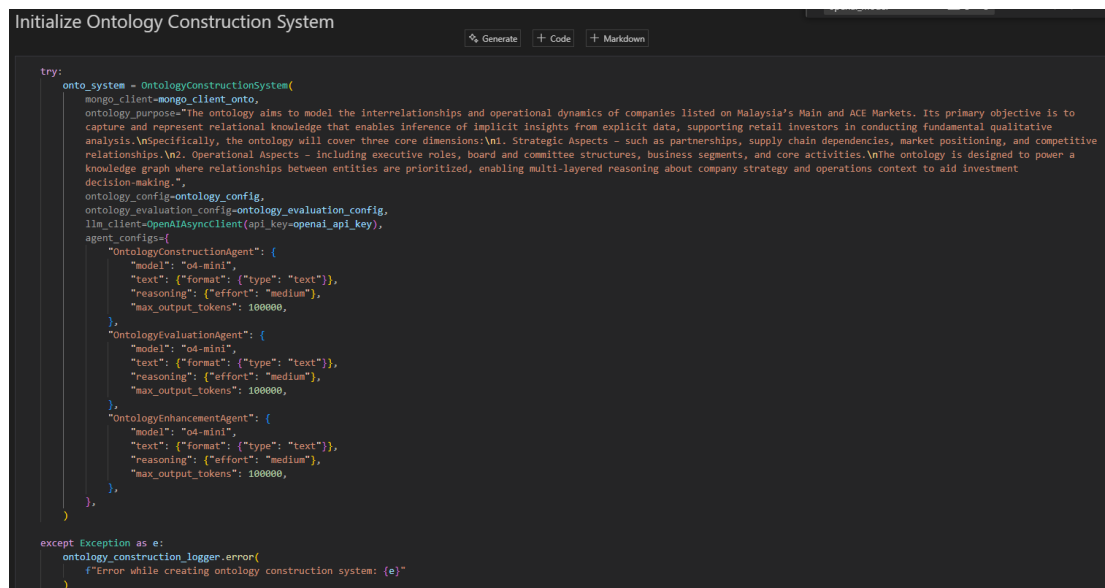


Figure 5.10 Initializing Ontology Construction System

## Ontology Extension

Ontology construction begins with the extension process. If no ontology exists in the system, a new one is created from scratch based on the defined purpose. The process is initiated when the user calls the `extend_ontology()` function, providing it with the source text and, optionally, a source text constraint.

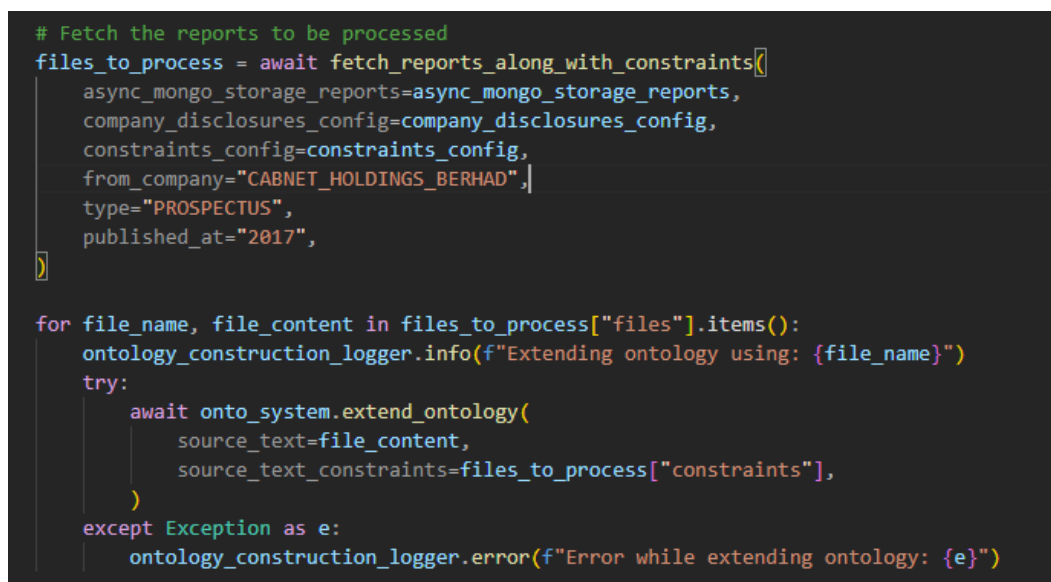


Figure 5.11 Extending Ontology

Once the `extend_ontology()` function is executed, a new document is added to the ontology collection in MongoDB. This document stores the extended ontology along with relevant metadata for auditing purposes.

```

▶ _id: ObjectId('68bd6569f8dc0f9390051003')
  ▼ ontology : Object
    ▶ entities : Object
    ▶ relationships : Object
    onto_purpose : "The ontology aims to model the interrelationships and operational dyna..."
    ▼ modifications : Object
      type : "EXTENSION"
      ▶ changes : Array (22)
      note : ""
    model_used : "o4-mini"
    created_at : 2025-09-07T10:58:49.118+00:00
    is_latest : false
    version : "1.0.1"

```

Figure 5.12 Ontology Extension Entry in MongoDB

## Ontology Enhancement Loop

After the ontology extension is completed, the ontology enhancement loop must be executed to further refine the ontology from two perspectives: (i) high-level, which evaluates the ontology as a whole, and (ii) low-level, which focuses on the attributes of entities and relationships. This process is triggered when the user calls the `enhance_ontology_via_loop()` function. The function automatically retrieves the latest ontology and applies the enhancement loop to it.

```

Enhance the Ontology

try:
    await onto_system.enhance_ontology_via_loop()
except Exception as e:
    ontology_construction_logger.error(f"Error while enhancing ontology: {e}")

```

Figure 5.13 Enhancing Latest Ontology via Loop

Once invoked, the enhancement loop initiates by calling the Ontology Evaluation Agent, which assesses the ontology at both high and low levels. Upon completion of the evaluation, a report is generated and stored in the `ontology_evaluation` collection within MongoDB for auditing purposes, irrespective of whether issues are detected.

```

_id: ObjectId('68bd84e72f26d27899ad7da1')
  ▼ evaluation_result : Array (3)
    ▼ 0: Object
      issue : "Inconsistent entity naming conventions (e.g., softwareProduct, authori..."
      impact : "Violates CamelCase naming convention for entity types, leading to mism..."
      suggestion : "Rename all entity types to use CamelCase (e.g., SoftwareProduct, Autho..."
    ▶ 1: Object
    ▶ 2: Object
    note : ""
  onto_purpose : "The ontology aims to model the interrelationships and operational dyna..."
  onto_version : "1.0.15"
  model_used : "o4-mini"
  created_at : 2025-09-07T13:13:11.304+00:00
  is_latest : false

```

*Figure 5.14 Ontology Evaluation Report Entry in MongoDB*

After the Ontology Evaluation Agent completes its evaluation, the system verifies whether the enhancement threshold has been reached. If the threshold is not met, the Ontology Enhancement Agent is invoked to refine the ontology at both high and low levels. The enhanced ontology, together with the justification for the enhancement, is stored in ontology collection in MongoDB for auditing purposes and is automatically designated as the latest version of the ontology. This enhancement loop continues iteratively until the enhancement threshold is satisfied.

```

    _id: ObjectId('68bd85482f26d27899ad7da2')
  ▾ ontology : Object
    ▸ entities : Object
    ▸ relationships : Object
    onto_purpose : "The ontology aims to model the interrelationships and operational dyna..."
  ▾ modifications : Object
    type : "ENHANCEMENT"
  ▾ changes : Array (10)
    ▾ 0: Object
      modification_made : "Renamed entity types 'country','businessActivity','fundingPurpose' to ..."
      justification : "Ensure CamelCase naming convention and consistency."
    ▸ 1: Object
    ▸ 2: Object
    ▸ 3: Object
    ▸ 4: Object
    ▸ 5: Object
    ▸ 6: Object
    ▸ 7: Object
    ▸ 8: Object

```

*Figure 5.15 Ontology Enhancement Entry in MongoDB*

### 5.4.2 Graph Construction

To begin graph construction, the user must initialize the Graph Construction Module by specifying the appropriate database connection variables and selecting the LLMs to be used for the Entity-Relationship Extraction Agent, the Entity Deduplication Agent, and the Relationship Deduplication Agent.

```

graph_system = GraphConstructionSystem(
    async_mongo_client=async_mongo_client,
    async_mongo_client_reports=async_mongo_client_reports,
    ontology_config=ontology_config,
    disclosure_config=disclosure_config,
    constraints_config=constraints_config,
    entity_config=entity_config,
    relationship_config=relationship_config,
    entity_vector_config=entity_vector_config,
    graphdb_config=graphdb_config,
    entity_cache_config=entities_cache_config,
    relationship_cache_config=relationships_cache_config,
    entity_cache_vector_config=entity_cache_vector_config,
    entities_deduplication_pending_tasks_config=entities_deduplication_pending_tasks_config,
    llm_client=OpenAIAsyncClient(api_key=openai_api_key),
    agent_configs={
        "EntityRelationshipExtractionAgent": {
            "model": "o4-mini",
            "text": {"format": {"type": "text"}},
            "reasoning": {"effort": "medium"},
            "max_output_tokens": 100000,
            "stream": False,
        },
        "EntityDeduplicationAgent": {
            "model": "gpt-5-mini",
            "text": {"format": {"type": "text"}},
            "reasoning": {"effort": "medium"},
            "max_output_tokens": 100000,
            "stream": False,
        },
        "RelationshipDeduplicationAgent": {
            "model": "gpt-5-mini",
            "text": {"format": {"type": "text"}},
            "reasoning": {"effort": "medium"},
            "max_output_tokens": 100000,
            "stream": False,
        },
    },
)

```

*Figure 5.16 Initializing Graph Construction System*

A graph construction process is considered complete after four key stages are executed: the Entity-Relationship Extraction, the Entity Deduplication, the Relationship Deduplication, and the Insertion of Entities and Relationships into the Knowledge Graph.

### Entity-Relationship Extraction

Entity and relationship extraction from the source text constitutes the initial activity in graph construction. This process is initiated when the user invokes the `extract_entities_relationships_from_unparsed_documents()` function, specifying the documents within the disclosures database to be processed.

## Extract Entities and Relationships from the Reports

```

try:
    await graph_system.extract_entities_relationships_from_unparsed_documents(
        from_company="VETECE_HOLDINGS_BERHAD",
        document_type="PROSPECTUS",
        published_at="08 Aug 2024",
        num_of_relationships_per_onto=6,
        exclude_documents=[],
    )
except Exception as e:
    graph_construction_logger.error(
        f"GraphConstructionSystem\nError while processing entities and relationships from unparsed documents:\n {e}"
    )

```

Figure 5.17 Extracting Entities and Relationships from Source Text

Following extraction, the identified entities and relationships are labelled as “TO\_BE\_DEDUPLICATED” and stored in the entities and relationships collections in MongoDB, respectively.

```

_id: ObjectId('68d03937a7309e06e1224dd5')
name: "Edeltec Holdings Berhad"
type: "Company"
▼ description: Array (1)
  0: "Edeltec Holdings Berhad was admitted to the Official List and listed o..."
▼ originated_from: Array (1)
  0: "EDELTEC_HOLDINGS_BERHAD"
status: "TO_BE_DEDUPLICATED"
created_at: 2025-09-21T17:43:19.149+00:00
last_modified_at: 2025-09-21T17:43:19.149+00:00

```

Figure 5.18 Extracted Entity in MongoDB

```

_id: ObjectId('68d03937a7309e06e1224dd7')
source_id: ObjectId('68d03937a7309e06e1224dd5')
target_id: ObjectId('68d03937a7309e06e1224dd6')
type: "hasCertification"
▼ description: Array (1)
▼ valid_in: Array (1)
▼ originated_from: Array (1)
status: "TO_BE_DEDUPLICATED"
created_at: 2025-09-21T17:43:19.149+00:00
last_modified_at: 2025-09-21T17:43:19.149+00:00

```

Figure 5.19 Extracted Relationship in MongoDB

## Entity Deduplication

Entity deduplication is performed subsequent to entity–relationship extraction. This process is initiated when the user invokes the `deduplicate_entities()` function, specifying the companies from which the entities originate.

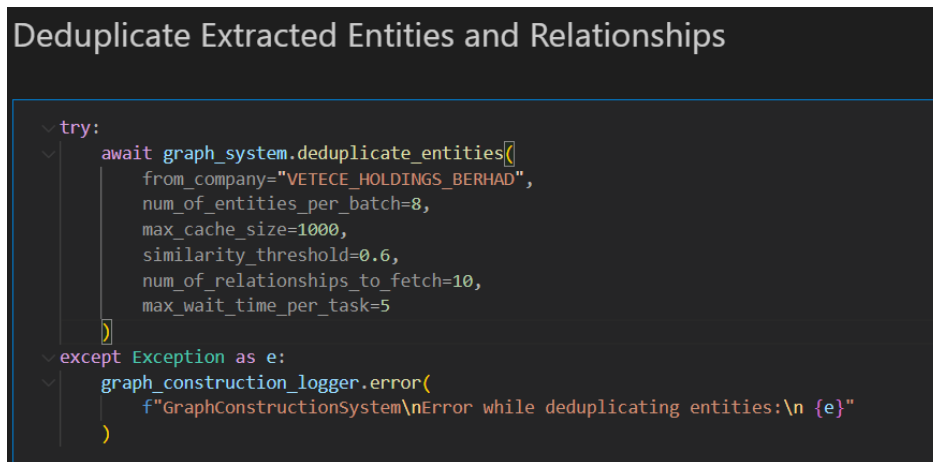


Figure 5.20 Deduplicating Entities

During deduplication, Pinecone is employed as an entity cache to support similarity-based candidate filtering. As deduplication is executed in batches, a locking mechanism is required. However, since Pinecone does not natively support locking, MongoDB is utilised as the single source of truth for the entity cache. Modifications to the cache are first applied in MongoDB and subsequently propagated to Pinecone. To ensure idempotency, this propagation requires the use of an additional collection in MongoDB, namely `entities_deduplication_pending_tasks`, which records all propagation tasks. This mechanism maintains consistency between the distributed databases in the event of a failure in either system.

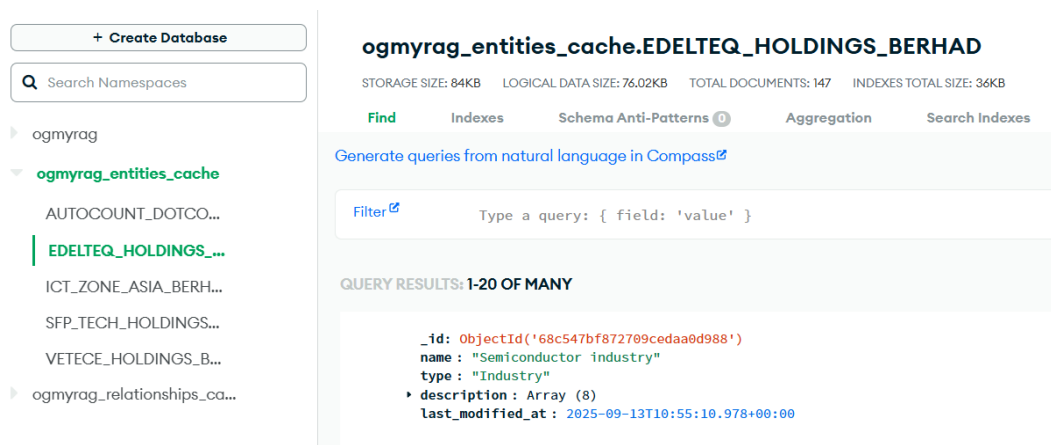


Figure 5.21 Entity Cache in MongoDB

Name	Records	Actions
VETECE_HOLDINGS_BERHAD	303	...
AUTOCOUNT_DOTCOM_BERHAD	287	...
ICT_ZONE_ASIA_BERHAD	234	...
EDELTEQ_HOLDINGS_BERHAD	147	...
SFP_TECH_HOLDINGS_BERHAD	146	...

Figure 5.22 Entity Cache in Pinecone

```

_id: ObjectId('68beb39c26565a1707dc9943')
from_company: "AUTOCOUNT_DOTCOM_BERHAD"
pending: false
type: "UPSERT"
▼ payload: Object
  _id: ObjectId('68be87fd26565a1707dc9543')
  name: "Section 212(8) of the Capital Markets and Services Act 2007"
  type: "RegulatoryProvision"
  ▶ description: Array (1)
  created_at: 2025-09-08T10:44:44.359+00:00

```

Figure 5.23 Entity Upsert Task Entry in MongoDB

After entity deduplication, each entity is either labelled as “TO\_BE\_UPSERTED\_INTO\_VECTOR\_DB” if it is not merged into another entity, or as “TO\_BE\_DELETED” if it has been merged into another entity.

## Relationship Deduplication

Relationship deduplication is performed after entity deduplication. This process is initiated when the user invokes the `deduplicate_relationships()` function, specifying the company from which the relationships are to be deduplicated.

```

try:
    await graph_system.deduplicate_relationships(
        from_company="VETECE_HOLDINGS_BERHAD",
        num_of_relationships_per_batch=8,
        max_cache_size=1000,
        max_wait_time_per_task=5
    )
except Exception as e:
    graph_construction_logger.error(
        f"GraphConstructionSystem\nError while deduplicating relationships:\n {e}"
    )

```

Figure 5.24 Deduplicating Relationship

After relationship deduplication, each relationship is either labelled as “TO\_BE\_UPSERTED\_INTO\_GRAPH\_DB” if it is not merged into another relationship, or as “TO\_BE\_DELETED” if it has been merged into another relationship.

```

_id: ObjectId('68be883226565a1707dc960a')
source_id: ObjectId('68be87fd26565a1707dc954d')
target_id: ObjectId('68be882826565a1707dc95c2')
type: "operatesIn"
▸ description: Array (1)
▸ valid_in: Array (1)
▸ originated_from: Array (1)
status: "TO_BE_DELETED"
created_at: 2025-09-08T07:39:30.040+00:00
last_modified_at: 2025-09-08T11:38:00.656+00:00

```

Figure 5.25 A Relationship Labelled “TO\_BE\_DELETED” in MongoDB

### Insertion of Entities and Relationships into the Knowledge Graph

After deduplication, the entities and relationships are prepared for upsertion into the knowledge graph in Neo4j. Prior to this step, the entities must first be upserted into Pinecone to ensure that they can be correctly referenced by the Text2Cypher Agent. Once upserted into Neo4j, the entities and relationships are labelled as “UPSERTED\_INTO\_GRAPH\_DB”.

Showing 10 hits	
1	<p>ID: 68be881e26565a1707dc955d</p> <p><b>description:</b> [ "Autocount Dotcom Berhad is a technology company.", "As of April 2023, it sought admission to the ACE Market of Bursa Securitie...</p> <p><b>name:</b> "Autocount Dotcom Berhad"</p> <p><b>type:</b> "Company"</p> <p>SCORE 0.9996</p>
2	<p>ID: 68be87fd26565a1707dc9547</p> <p><b>description:</b> [ "Autocount Dotcom Berhad was incorporated in Malaysia on 25 February 2022.", "It was incorporated as a public limited company ...</p> <p><b>name:</b> "Autocount Dotcom Berhad"</p> <p><b>type:</b> "Company"</p> <p>SCORE 0.9996</p>

Figure 5.26 Entities in Pinecone



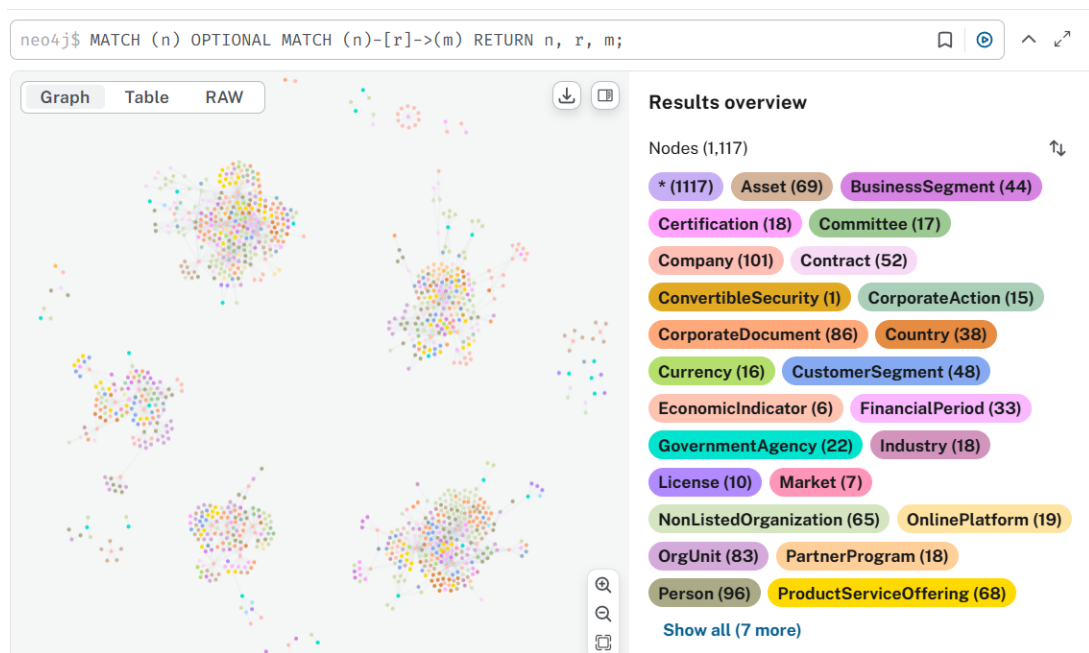


Figure 5.27 Entities and Relationships in Neo4J

### 5.4.3 Graph Retrieval

To retrieve information from the graph, the user currently accesses it through a Gradio-based chatbot. The chatbot may be accessed either via the instance hosted on Hugging Face Spaces at <https://huggingface.co/spaces/CheeQin/ogmyrag>, or through the locally hosted version implemented in the Graph Retrieval Jupyter Notebook. The hosted chatbot becomes inactive if no interaction occurs for 48 hours, and therefore cannot be accessed unless it is deliberately reactivated by the author.

For the locally hosted chatbot, the user must initialise the Graph Retrieval System and configure the large language models (LLMs) utilised by the Chat Agent, the Request Composition Agent, the Query Agent, the Text2Cypher Agent, and the Retrieval Result Compilation Agent. At present, the Vector Agent is not integrated into the Graph Retrieval System for configuration due to incompatibilities in its implementation, which was developed as part of a separate final-year project.

```

graph_system = GraphRetrievalSystem(
    mongo_client=mongo_client,
    ontology_config=ontology_config,
    entity_vector_config=entity_vector_config,
    graphdb_config=graphdb_config,
    rag_vector_config=rag_vector_config,
    llm_client=OpenAIAsyncClient(api_key=openai_api_key),
    agent_configs={
        "ChatAgent": {
            "model": "o4-mini",
            "text": {"format": {"type": "text"}},
            "reasoning": {"effort": "medium"},
            "max_output_tokens": 100000,
        },
        "RequestDecompositionAgent": {
            "model": "gpt-5-mini",
            "text": {"format": {"type": "text"}},
            "reasoning": {"effort": "medium"},
            "max_output_tokens": 100000,
        },
        "QueryAgent": {
            "model": "o4-mini",
            "text": {"format": {"type": "text"}},
            "reasoning": {"effort": "high"},
            "max_output_tokens": 100000,
        },
        "Text2CypherAgent": {
            "model": "o4-mini",
            "text": {"format": {"type": "text"}},
            "reasoning": {"effort": "high"},
            "max_output_tokens": 100000,
        },
        "RetrievalResultCompilationAgent": {
            "model": "gpt-4.1-mini",
            "text": {"format": {"type": "text"}},
            "max_output_tokens": 32768,
        },
    },
)

```

Figure 5.28 Initializing Graph Retrieval System

## Set Up User Interface

```

async def respond(message: str, history: list):
    accumulated_content = ""
    try:
        async for chunk in graph_system.query(
            user_request=message,
            top_k_for_similarity=20,
            similarity_threshold=0.6,
            max_tool_call=4,
        ):
            accumulated_content += chunk + "\n\n"
            yield accumulated_content
    except Exception as e:
        graph_retrieval_logger.error(
            f"GraphRetrievalSystem\nError while processing user request: {e}"
        )
        yield "***Unexpected error occurred. Please contact the developer.**"

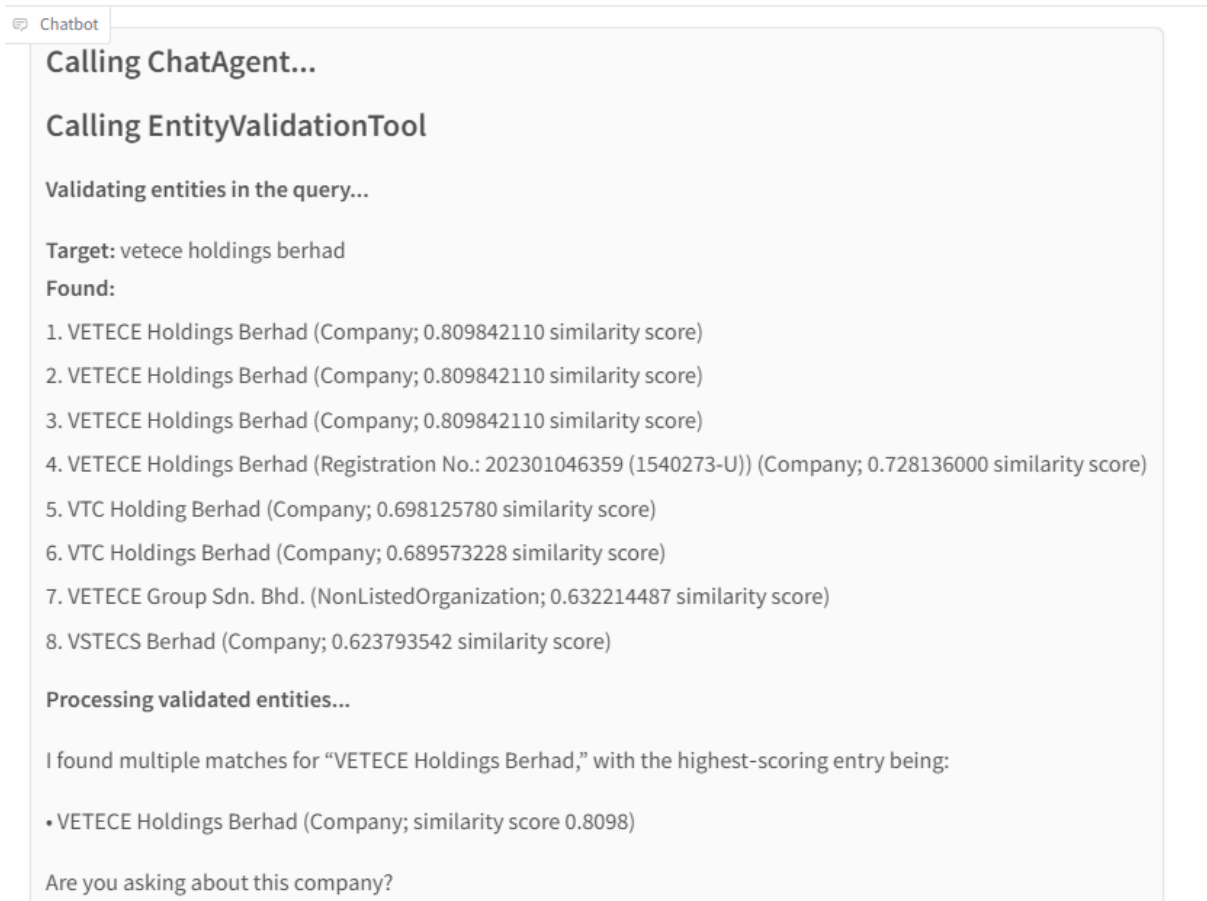
demo = gr.ChatInterface(
    fn=respond,
    type="messages",
    title="Ontology-Grounded Graph-Based RAG",
)

demo.launch()

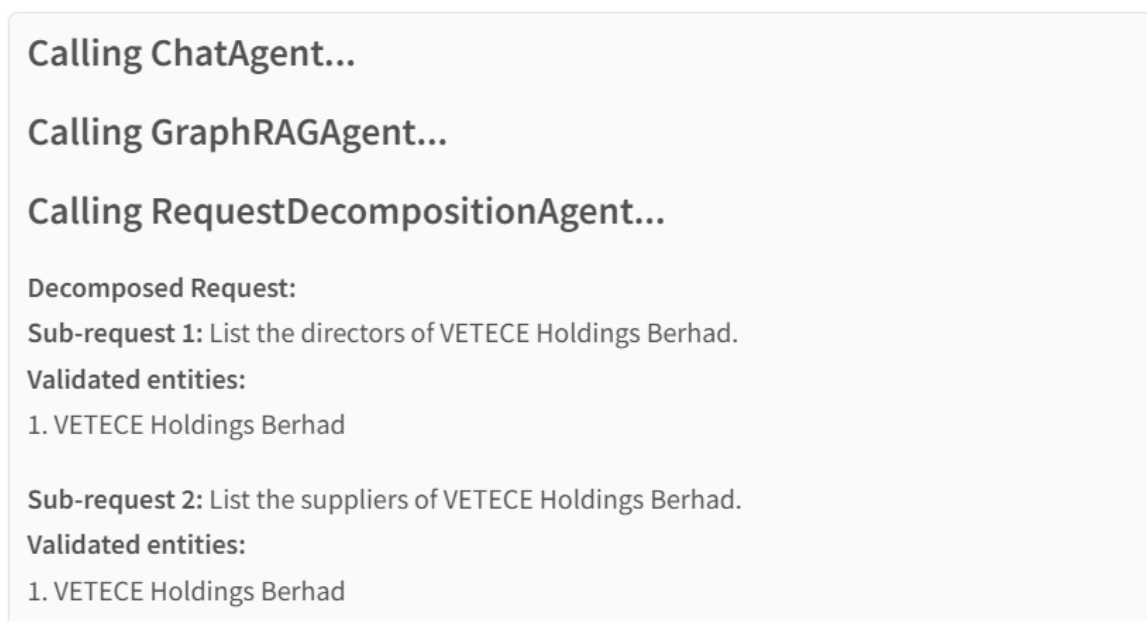
```

*Figure 5.29 Setting Up Gradio Chatbot Locally*

It should be noted that the chatbot has been configured such that the internal processes of the Chat Agent are also displayed, thereby demonstrating the underlying operations occurring behind the scenes.



*Figure 5.30 The Chat Agent Confirming Entity to be Queried with the User*



*Figure 5.31 The Request Decomposition Agent Decomposing the Query*

### Calling Text2Cypher Agent for Query Agent 1 (Iteration 1/3)...

Executing the generated Cypher query:

```
MATCH (supplier:Company)-[r:suppliesTo]->(receiver:Company {name: 'VETECE Holdings Berhad'})
RETURN supplier { .name, .type, .description } AS supplier,
r { .valid_in, .description } AS suppliesToRelationship,
receiver { .name, .type, .description } AS receiver
```

*Figure 5.32 Cypher Query Generated by the Text2Cypher Agent*

### Calling ChatAgent to process combined final response...

Here's the information for VETECE Holdings Berhad:

1. Directors:

- Tee Chee Chiang
  - Non-Independent Executive Vice Chairman
  - Director since incorporation (22 Nov 2023)
  - Founder, promoter; substantial shareholder (61.32% post-IPO)
- Chan Wai Hoong
  - Non-Independent Executive Director & Chief Executive Officer
  - Appointed 22 Nov 2023; leads operations & strategy
  - Shareholding ~~3.36~~ 4.00% post-IPO

*Figure 5.33 The Chat Agent Formulating Final Response*

## 5.5 Implementation Issues and Challenges

The proposed ontology-grounded, graph-based RAG with text-to-Cypher retrieval pipeline, which powers the corporate insight derivation module, can be divided into three sub-processes, corresponding to three sub-modules. The implementation issues and challenges are therefore discussed for each sub-module.

### Ontology evaluation phase:

The primary challenge lies in enhancing the ontology after extraction. The Ontology Construction Agent frequently extracts entities and relationships that are overly nuanced, limiting their potential for reuse or generalisation, even when explicitly prompted otherwise. An enhancement mechanism is therefore required to prevent the ontology from becoming excessively large. The initial mechanism was implemented as a sequential pipeline, with one

agent responsible for high-level enhancement and another for low-level enhancement. However, the absence of collaboration or discussion between the agents led to undesirable outcomes: relevant entities and relationships were often deleted, as no alternative perspectives were available for conflict resolution. Furthermore, employing two separate agents for enhancement introduced unnecessary complexity. An alternative approach, in which two agents collaboratively discussed potential refinements, alleviated this issue but proved difficult to converge. A more effective solution was subsequently devised in the form of an enhancement loop with an increasing threshold. In this design, one agent flags issues, identifies their impact, and proposes resolutions, while another agent performs the modifications at both high and low levels without being required to adopt the suggestions verbatim. Although not flawless, this method produces an ontology that captures broader aspects of the data while remaining manageable and generalisable.

### **Graph construction phase:**

The main challenge occurs during entity deduplication. Exhaustive deduplication is costly, and sequential processing is slow. To mitigate this, a cache-based, batch-oriented deduplication process was introduced. However, batching combined with the shared resource of an entity cache can lead to race conditions, necessitating a locking mechanism. Pinecone, the vector database employed as the entity cache, does not support the implementation of a Least Recently Used (LRU) algorithm for cache size management. Therefore, a copy of the entity cache must also be maintained in MongoDB, which supports the operation. This dual-storage design introduces a synchronisation problem, as updates cannot be executed within a single transaction across both databases. Without careful handling, this may lead to inconsistencies if an update succeeds in one database but fails in the other. The final solution employs an idempotent mechanism. Each modification to the MongoDB entity cache triggers the creation of a corresponding deduplication task with status “pending” in a separate MongoDB collection. These two operations can be executed within a single transaction. Propagation to Pinecone is then carried out by executing the deduplication tasks before updating their status. Even in the event of sudden failures, re-execution of the tasks ensures consistency, as Pinecone operations consist of upserts and deletions, both of which are idempotent.

### **Graph retrieval phase:**

The main issue lies in task delegation, which must balance precise Cypher query retrieval with maintaining a compact system that avoids excessive complexity, slower performance, and

higher costs. The Cypher Agent must reference the correct entity when formulating a query. To prevent wasted computation, this validation must occur before Cypher query formulation; otherwise, additional search time is incurred if the entity is later found to be invalid. To address this, entity confirmation is delegated to the Chat Agent, enabling early identification of invalid queries. This, however, comes at the cost of user experience, as the user is required to confirm their query whenever the Chat Agent interacts with the Graph RAG Agent. Furthermore, the Text2Cypher Agent may misinterpret queries or generate incorrect Cypher statements, even when the relevant data exist in the graph database. To mitigate this, the Query Agent is introduced, enabling the Text2Cypher Agent to focus solely on query generation. The Query Agent evaluates both the generated Cypher query and its retrieval results to determine whether re-execution is necessary. Although this additional step may increase retrieval latency, it leads to more precise and reliable query results.

## Chapter 6 System Evaluation and Discussion

### 6.1 System Testing and Performance Metrics

The proposed ontology-grounded, graph-based RAG with text-to-Cypher retrieval pipeline, which powers the corporate insight derivation module, is divided into three sub-modules. Accordingly, testing is conducted for each sub-module to evaluate the soundness of its deliverables. The testing procedures and performance metrics for the Ontology Construction Agent, the Graph Construction Agent, and the Graph Retrieval Agent are presented in Sections 6.1.1, 6.1.2, and 6.1.3, respectively.

#### 6.1.1 Testing and Performance Metrics for the Ontology Construction Module

The deliverable of the Ontology Construction Module is an ontology. However, due to the absence of domain experts for assessing its robustness, a modified evaluation approach was adopted. Inspired by the competency-question-based evaluation method in [14], an ontology-purpose-oriented, competency-question-based, and LLM-powered evaluation framework was employed to evaluate the robustness of the ontology.

The objective of the testing is **to evaluate the competency of the constructed ontology by assessing it against a set of competency questions.**

The definition of a modelled competency question follows that in [14]: “A competency question is regarded as being modelled within an ontology if and only if there exists a SPARQL query capable of retrieving its answer from the ontology, regardless of the ontology’s modelling quality or compliance with best practices.”

For the purposes of the proposed project, Cypher is used in place of SPARQL for retrieval. Accordingly, the definition is adapted as follows: “A competency question is regarded as being modelled within an ontology if and only if there exists a Cypher query capable of retrieving its answer from the ontology, regardless of the ontology’s modelling quality or compliance with best practices.”

The testing procedure is as follows:



1. A set of competency questions is generated by an LLM, based on the stated purpose of the ontology.
2. A second LLM then evaluates whether each competency question can be modelled within the ontology, thereby assessing its robustness.

### 6.1.2 Testing and Performance Metrics for the Graph Construction Module

The deliverable of the Graph Construction Module is a knowledge graph stored in the graph database. However, the capability of the knowledge graph to support the derivation of implicit insights from explicit data is assessed in the testing of the Graph Retrieval Module. Consequently, another core component of the Graph Construction Module—the entity and relationship deduplication process—is evaluated instead.

To assess the effectiveness of the deduplication process, a metric termed the **deduplication rate** is defined. It is calculated as the number of entities deduplicated within a company divided by the total number of entities for that company.

### 6.1.3 Testing and Performance Metrics for the Graph Retrieval Module

The deliverable of the Graph Retrieval Module is a response generated by the underlying Vector RAG Agent and Graph RAG Agent. The Vector RAG Agent is omitted here, as it was developed by another student. To evaluate the quality of responses in deriving implicit insights from explicit data, a methodology similar to that used for the Ontology Construction Module is adopted.

The testing procedures is as below:

1. A set of competency questions is generated based on the ontology, containing both easy and hard questions, with difficulty levels defined as in [14]
  - Easy: The competency questions require at most 2 entities and 1 relationship to answer.
  - Hard: The competency questions require at least 2 entities and 1 relationship to answer.
2. The competency questions are adapted to the context of the companies present in the graph database. The Graph Retrieval Module is then tasked with generating responses for these questions.

3. The response by the Graph Retrieval Module are then evaluated based on the metrics proposed in [12], extended with a five-point Liker scale as shown in Table 6.1. The evaluation criteria are as follows:

- **Comprehensiveness:**

This criterion assesses the extent to which the response thoroughly addresses all relevant aspects of the question, including supporting details and explanations.

- **Diversity:**

This criterion evaluates whether the response presents different insights that enhance the reader's understanding of the topic.

- **Empowerment:**

This criterion measures the extent to which the response enables the reader to make well-informed decisions.

- **Directness:**

This criterion assesses how directly and concisely the response addresses the core of the question. It serves as a baseline for comparing the soundness of results under the other criteria, as it contrasts with comprehensiveness and diversity.

Score	Definition	Description
1	Very Poor	The response does not meet the criterion at all.
2	Poor	The response minimally meets the criterion.
3	Fair	The response adequately meets the criterion but lacks clarity or depth.
4	Good	The response clearly meets the criterion with sufficient clarity and relevance.
5	Excellent	The response fully meets the criterion and exceeds expectations in depth and quality.

*Table 6.1 Five-point Likert scale definitions for response quality evaluation*

The competency questions are generated from ontology not ontology purpose in this testing to limit the testing scope.

## 6.2 Testing Setup and Results

The testing setup and results for the Ontology Construction Agent, the Graph Construction Agent, and the Graph Retrieval Agent are presented in Sections 6.2.1, 6.2.2, and 6.2.3, respectively.

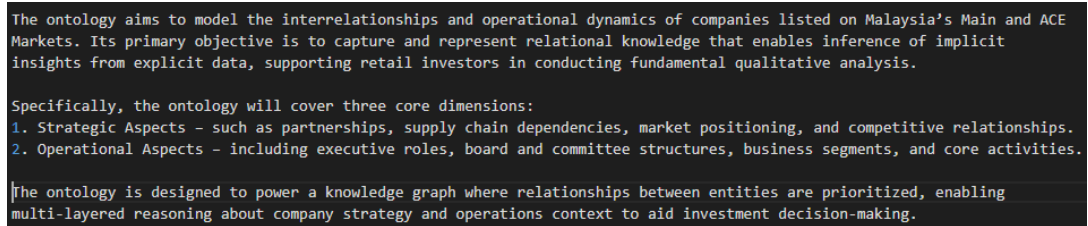
### 6.2.1 Testing Setup and Results for Ontology Construction Agent

As detailed in Section 6.1.1, the testing on an ontology-purpose-oriented, competency-question-based, and LLM-powered evaluation.

#### Testing Setup

##### 1. Ontology purposed used

The ontology purpose, which guided the construction of the ontology and the generation of competency questions, is illustrated in Figure 6.1.



The ontology aims to model the interrelationships and operational dynamics of companies listed on Malaysia's Main and ACE Markets. Its primary objective is to capture and represent relational knowledge that enables inference of implicit insights from explicit data, supporting retail investors in conducting fundamental qualitative analysis.

Specifically, the ontology will cover three core dimensions:

1. Strategic Aspects - such as partnerships, supply chain dependencies, market positioning, and competitive relationships.
2. Operational Aspects - including executive roles, board and committee structures, business segments, and core activities.

The ontology is designed to power a knowledge graph where relationships between entities are prioritized, enabling multi-layered reasoning about company strategy and operations context to aid investment decision-making.

*Figure 6.1 Ontology Purpose*

##### 2. Competency Question Generation

The competency questions are generated using o4-mini by OpenAI with medium reasoning effort. The prompt employed for question generation is shown in Figure 6.2, while the resulting competency questions are presented in Figure 6.3.

```
PROMPT["CQ_GENERATION"] = """
You are a Competency Question (CQ) Generation Agent. Your task is to generate competency questions aligned with the given ontology purpose.

Guidelines:
1. Ontology Purpose Alignment
   - All competency questions must directly reflect and be centered on the provided ontology purpose.

2. Quantity Requirement
   - Generate exactly {cq_num} competency questions. Do not exceed or fall short of this number.

3. Output Format
   - Return the results strictly as raw JSON (no code blocks, explanations, or additional text).
   - Use the structure below:
   {
     "competency_questions": [
       "cq_1",
       "cq_2"
     ]
   }

Proceed to generate the competency questions, strictly adhering to these guidelines.
```

*Figure 6.2 Competency Question Generation Prompt for Ontology Evaluation*

1. Which companies have strategic partnerships with each other?
2. Which companies are direct suppliers to a specified company?
3. Which companies compete within the same market segment?
4. Which companies share one or more board members?
5. Which individuals serve as executives in more than one listed company?
6. What are the defined business segments of a specified company?
7. Which companies have established joint ventures with others?
8. Which companies depend on a specified supplier for core inputs?
9. Which companies hold shareholding stakes in other companies, and what are their percentages?
10. Which committees (e.g., audit, risk, remuneration) exist within a specified company?
11. What core activities are undertaken by a specified company?
12. Which companies co-invest in the same subsidiary or project?
13. Which companies lead the market in a specified business segment?
14. Which companies have multi-tier supply chain dependencies?
15. Which executives serve concurrently on the boards of two or more companies?
16. Which companies have overlapping membership in their board committees?
17. Which companies are concurrently listed on both the Main and ACE Markets?
18. Which companies have formal strategic alliances targeting a specified industry?
19. Which companies' operational structures include a specified committee type?
20. Which companies are indirectly connected through a chain of supply dependencies?

*Figure 6.3 Competency Questions Generated for Ontology Evaluation*

### 3. Ontology Competency Evaluation

The ontology competency evaluation is performed using o4-mini from OpenAI with medium reasoning effort. The evaluation prompt is shown in Figure 6.4. A total of 20 competency questions are used to assess the constructed ontology, the details of which are provided in Appendix A.

```
PROMPT["ONTOLOGY_CQ_EVALUATION"] = """
You are an Ontology Competency Evaluation Agent. Your task is to evaluate whether the given ontology can answer each competency question.

Guidelines
1. Evaluation Criteria
  - For every competency question, provide:
    - "is_competent": Indicate if the ontology can answer the question ("TRUE" or "FALSE").
    - "justification" : A concise explanation supporting your evaluation decision.

2. Definition of Competent
  - A competency question is regarded as being modelled within an ontology if and only if there exists a Cypher query capable of retrieving its answer from the ontology, regardless of the ontology's modelling quality or compliance with best practices.

3. Output Format
  - Return the results strictly as raw JSON (no code blocks, explanations, or additional text).
  - Use the following structure:

{
  "competency_evaluation_result": [
    {
      "competency_question": "cq_1",
      "is_competent": "TRUE_OR_FALSE",
      "justification": "your_justification"
    }
  ]
}
```

*Figure 6.4 Ontology Competency Evaluation Prompt*

## Results

The evaluation results of the ontology are presented in Figures 6.5, 6.6, and 6.7. Among the 20 competency questions assessed, 15 were evaluated as successfully modelled by the Ontology Competency Evaluation Agent, achieving a competency rate of 75%.

The following questions were evaluated as not modelled by the ontology:

1. Which companies have established joint ventures with others?
2. Which companies hold shareholding stakes in other companies, and what are their percentages?
3. What core activities are undertaken by a specified company?
4. Which companies co-invest in the same subsidiary or project?
5. Which companies lead the market in a specified business segment?

These five questions are believed to be addressable through the ontology enhancement loop, which would improve the robustness of the ontology. The testing also indicates that competency evaluation represents a potential approach to enhance the robustness of ontology generation within the current construction pipeline.

1	Which companies have strategic partnerships with each other?	TRUE	The hasStrategicPartnership relationship directly links Company nodes in partnership.
2	Which companies are direct suppliers to a specified company?	TRUE	The suppliesTo relationship from Company to Company enables querying direct suppliers.
3	Which companies compete within the same market segment?	TRUE	The competesWith relationship between Company nodes captures competition in segments.
4	Which companies share one or more board members?	TRUE	Person-holdsPositionAtCompany edges (e.g., Director) allow identifying persons on multiple company boards.
5	Which individuals serve as executives in more than one listed company?	TRUE	Person-holdsPositionAtCompany relationships can be queried for persons linked to multiple companies in executive roles.
6	What are the defined business segments of a specified company?	TRUE	The hasBusinessSegment relationship links a Company to its BusinessSegment nodes.
7	Which companies have established joint ventures with others?	FALSE	No relationship for joint ventures is defined in the ontology.
8	Which companies depend on a specified supplier for core inputs?	TRUE	Reverse suppliesTo relationships identify companies depending on a given supplier.
9	Which companies hold shareholding stakes in other companies, and what are their percentages?	FALSE	No relationship models Company-to-Company equity stakes with percentage details.
10	Which committees (e.g., audit, risk, remuneration) exist within a specified company?	TRUE	The hasCommittee relationship links a Company to its Committee nodes.

*Figure 6.5 Ontology Competency Evaluation Result - Part One*

10	Which committees (e.g., audit, risk, remuneration) exist within a specified company?	TRUE	The hasCommittee relationship links a Company to its Committee nodes.
11	What core activities are undertaken by a specified company?	FALSE	No generic Activity or core operations entity or relationship is modeled.
12	Which companies co-invest in the same subsidiary or project?	FALSE	No co-investment or joint project relationship is defined.
13	Which companies lead the market in a specified business segment?	FALSE	Market leadership or ranking is not represented in the ontology.
14	Which companies have multi-tier supply chain dependencies?	TRUE	Chains of suppliesTo relationships can be traversed to identify multi-tier dependencies.
15	Which executives serve concurrently on the boards of two or more companies?	TRUE	Person-holdsPositionAtCompany can identify individuals holding board roles at multiple companies.

*Figure 6.6 Ontology Competency Evaluation Result - Part Two*

16	Which companies have overlapping membership in their board committees?	TRUE	hasCommittee and hasCommitteeMember relationships allow finding shared committee members across companies.
17	Which companies are concurrently listed on both the Main and ACE Markets?	TRUE	Multiple listedOn edges to Market nodes enable querying companies listed on both markets.
18	Which companies have formal strategic alliances targeting a specified industry?	TRUE	hasStrategicPartnership combined with hasIndustry on partner companies supports this query.
19	Which companies' operational structures include a specified committee type?	TRUE	hasCommittee links companies to named Committee nodes, which can be filtered by committee type.
20	Which companies are indirectly connected through a chain of supply dependencies?	TRUE	Transitive traversal over suppliesTo relationships identifies indirect supply connections.

Figure 6.7 Ontology Competency Evaluation Result - Part Three

## 6.2.2 Testing Setup and Results for Graph Construction Agent

### Testing Setup

To determine the deduplication rate, two functions, `get_entity_count()` and `get_relationship_count()`, were introduced in the Graph Construction System. For clarity, the entity cache size for each company is set to 1,000, and the model used for entity deduplication is **gpt-5-mini** from OpenAI. Similarly, the relationship cache size is also set to 1,000.

```

try:
    node_count = await graph_system.get_entity_count(
        {
            "status": {"$in": ["UPSERVED_INTO_GRAPH_DB", "TO_BE_DELETED"]},
            "originated_from": {"$in": ["VETECE_HOLDINGS_BERHAD"]},
        }
    )
    relationship_count = await graph_system.get_relationship_count(
        {
            "status": {"$in": ["UPSERVED_INTO_GRAPH_DB", "TO_BE_DELETED"]},
            "originated_from": {"$in": ["VETECE_HOLDINGS_BERHAD"]},
        }
    )
    graph_construction_logger.info(
        f"GraphConstructionSystem\nEntity count based on given filter: {node_count}\nRelationship count based on given filter: {relationship_count}"
    )
except Exception as e:
    graph_construction_logger.error(
        f"GraphConstructionSystem\nError while creating graph construction system: {e}"
    )

```

2025-09-22 10:48:30,807 - graph\_construction - INFO - GraphConstructionSystem  
Entity count based on given filter: 1041  
Relationship count based on given filter: 849

Figure 6.8 `get_entity_count()` and `get_relationship_count()` in the Graph Construction System

### Results

The evaluation shows that entity-cache-based deduplication (with an entity cache per company), which utilises similarity search for pre-filtering and an LLM for merging decisions,

achieves notable results. The deduplication rates range from a minimum of 52.1% to a maximum of 73.0%, with an overall deduplication rate of 66.5%, all exceeding 50%.

In contrast, the comparatively simpler relationship deduplication approach, which merges relationships occurring between the same source and target entities, achieves lower performance, with a minimum deduplication rate of 13.9%, a maximum of 36.4%, and a total of 29.9%. This lower performance is likely due to the relatively small number of relationships established between the same pairs of entities.

Company	Entities Extracted	Entities To Be Deleted	Entity Deduplication Rate	Relationships Extracted	Relationships To Be Deleted	Relationship Deduplication Rate
VETECE_HOLDINGS_BERHAD	1,041	738	70.9%	849	309	36.4%
AUTOCOUNT_DOTCOM_BERHAD	1,062	775	73.0%	927	286	30.9%
ICT_ZONE_ASIA_BERHAD	618	384	62.1%	500	161	32.2%
EDELTEQ_HOLDINGS_BERHAD	313	166	53.1%	251	35	13.9%
SFP_TECH_HOLDINGS_BERHAD	305	159	52.1%	269	46	17.1%
<b>Total</b>	<b>3,339</b>	<b>2,222</b>	<b>66.5%</b>	<b>2,796</b>	<b>837</b>	<b>29.9%</b>

*Figure 6.9 Entity and Relationship Deduplication Result*

### 6.2.3 Testing Setup and Results for Graph Retrieval Agent

#### Testing Setup

##### 1. Competency Question Generation

The prompt employed for generating competency questions to evaluate the knowledge graph is illustrated in Figure 6.10. Competency questions are generated using OpenAI's o4-mini model, configured with a medium reasoning effort. In total, three questions were generated for evaluation purposes: two classified as easy and one as hard, as shown in Figure 6.11.



```

PROMPT[
  "CQ_GENERATION_GRAPH"
] = """
You are a Competency Question (CQ) Generation Agent. Your task is to generate competency questions based on the given ontology.

Guidelines:
1. Ontology as the Base
   - All competency questions must be derived directly from the entity and relationship types defined in the ontology.
   You, 4 minutes ago • Uncommitted changes
2. Difficulty Levels
   - Categorize each competency question as either "EASY" or "HARD" based on the following criteria:
     - EASY: Requires at most 2 entities and 1 relationship to answer.
     - HARD: Requires at least 2 entities and 1 relationship to answer.
3. Quantity Requirement
   - Generate exactly {easy_cq_num} easy competency questions and {hard_cq_num} hard competency questions. Do not exceed or fall short of these numbers.
4. Question Requirement
   - Each question must be a query template that enables a user to retrieve or insert specific instances in the graph database.
   - Avoid general questions. For example, instead of: "Which Person holds a ShareOption granted by a Company?", use a template like: "Find all Persons holding a ShareOption granted by {Company_Name}." For the current stage, only one placeholder is enough for each question.
5. Output Format
   - Return the results strictly as raw JSON (no code blocks, explanations, or additional text).
   - Follow this structure exactly:
     {
       "competency_questions": [
         {
           "difficulty": "HARD_OR_EASY",
           "question": "your_question_here"
         }
       ]
     }

```

Figure 6.10 Competency Question Generation Prompt for Knowledge Graph Evaluation

#	Difficulty	Question
1	EASY	Find all BusinessSegments associated with [Company_Name].
2	EASY	Find all Currencies in which [Company_Name] transacts.
3	HARD	Find all Persons serving on committees of companies listed on [Market_Name].

Figure 6.11 Competency Questions Generated Prompt for Knowledge Graph Evaluation

## 2. Knowledge Graph Competency Evaluation

The prompt used to assess the responses of the Graph Retrieval Agent to the competency questions is presented in Figure 6.12. The evaluation agent is powered by OpenAI's o4-mini model, also configured with medium reasoning effort.

```

You are the Competency Question Response Evaluation Agent. Your task is to evaluate the quality of a response to a given competency question.

Guidelines:
1. Evaluation Criteria
- Evaluate the response based on the following criteria:
  - Comprehensiveness: Assesses how thoroughly the response addresses all relevant aspects of the question, including supporting details and explanations.
  - Diversity: Evaluates whether the response presents multiple perspectives or insights that enhance understanding of the topic.
  - Empowerment: Measures the extent to which the response enables the reader to make well-informed decisions.
  - Directness: Assesses how directly and concisely the response addresses the core of the question. This serves as a baseline for comparing results under other criteria, contrasting with comprehensiveness and diversity.
2. Evaluation Metric
- For each criterion, assign one of the following metrics:
  - Very Poor: The response does not meet the criterion at all.
  - Poor: The response minimally meets the criterion.
  - Fair: The response adequately meets the criterion but lacks clarity or depth.
  - Good: The response clearly meets the criterion with sufficient clarity and relevance.
  - Excellent: The response fully meets the criterion and exceeds expectations in depth and quality.
3. Output Requirements
- Return the evaluation strictly as raw JSON.
- Do not include code blocks, explanations, or additional text.
- Use this exact JSON structure:
{
  "competency_question": "",
  "response": "",
  "evaluation_report": {
    "comprehensiveness": {
      "metric": "",
      "justification": ""
    },
    "diversity": {
      "metric": "",
      "justification": ""
    },
    "empowerment": {
      "metric": "",
      "justification": ""
    },
    "directness": {
      "metric": "",
      "justification": ""
    }
  }
}

```

*Figure 6.12 Knowledge Graph Competency Evaluation Prompt*

## Results

This section presents the responses of the Graph Retrieval Agent to the competency questions, the corresponding evaluation results, and a review of both the responses and their evaluations.

### 1. Competency Question One (Easy)

#### Ontology-Grounded Graph-Based RAG

Chatbot

Combined Final Response from QueryAgents

**Query Agent 1:**

**Final Response:** VETECE Holdings Berhad has the following principal business segments: Enterprise IT solutions; Implementation Services; Maintenance, Support and Professional Services; Resale of Hardware and Software.

**Note:** NA

**Calling ChatAgent to process combined final response...**

VETECE Holdings Berhad's principal business segments are:

- Enterprise IT Solutions
- Implementation Services
- Maintenance, Support and Professional Services
- Resale of Hardware and Software

These segments represent the core areas through which VETECE delivers and supports its technology offerings.

*Figure 6.13 Response of the Graph Retrieval System to Competency Question One*

**Competency Question:**

Find all BusinessSegments associated with [Company\_Name].

**Response:**

VETECE Holdings Berhad's principal business segments are:

- Enterprise IT Solutions
- Implementation Services
- Maintenance, Support and Professional Services
- Resale of Hardware and Software

These segments represent the core areas through which VETECE delivers and supports its technology offerings.

**Evaluation Report:**

Criterion	Metric	Justification
Comprehensiveness	Good	The response clearly lists all relevant business segments and provides a concise description of their role.
Diversity	Poor	The response offers only a straightforward list without multiple perspectives, examples, or deeper insights.
Empowerment	Fair	While it identifies the segments, it lacks guidance on how to leverage this information for decision-making or further analysis.
Directness	Excellent	The answer directly and succinctly addresses the question by listing the business segments without extraneous information.

Figure 6.14 Evaluation Report of the Graph Retrieval System Response to Competency Question One

2. Competency Question Two (Easy)

Ontology-Grounded Graph-Based RAG

Chatbot

Agent 1:

**Final Response:** VETECE Holdings Berhad transacts in the following currencies: Malaysian Ringgit (MYR), Singapore Dollar (SGD), and United States Dollar (USD).  
**Note:** NA

**Combined Final Response from QueryAgents**

**Query Agent 1:**  
**Final Response:** VETECE Holdings Berhad transacts in the following currencies: Malaysian Ringgit (MYR), Singapore Dollar (SGD), and United States Dollar (USD).  
**Note:** NA

**Calling ChatAgent to process combined final response...**

VETECE Holdings Berhad transacts in the following currencies:

- Malaysian Ringgit (MYR)
- Singapore Dollar (SGD)
- United States Dollar (USD)

Figure 6.15 Response of the Graph Retrieval System to Competency Question Two

<b>Competency Question:</b>		
Find all Currencies in which [Company_Name] transacts.		
<b>Response:</b>		
VETECE Holdings Berhad transacts in the following currencies:		
<ul style="list-style-type: none"> <li>• Malaysian Ringgit (MYR)</li> <li>• Singapore Dollar (SGD)</li> <li>• United States Dollar (USD)</li> </ul>		
<b>Evaluation Report:</b>		
Criterion	Metric	Justification
Comprehensiveness	Excellent	The response fully lists all relevant currencies in which the company transacts, directly addressing the question.
Diversity	Poor	The response provides a simple list without additional perspectives, context, or comparative insights.
Empowerment	Fair	While the response identifies the currencies, it offers no guidance on implications or how to use this information effectively.
Directness	Excellent	The answer is concise and directly addresses the competency question without unnecessary information.

Figure 6.16 Evaluation Report of the Graph Retrieval System Response to Competency Question Two

### 3. Competency Question Three (Hard)

#### Ontology-Grounded Graph-Based RAG

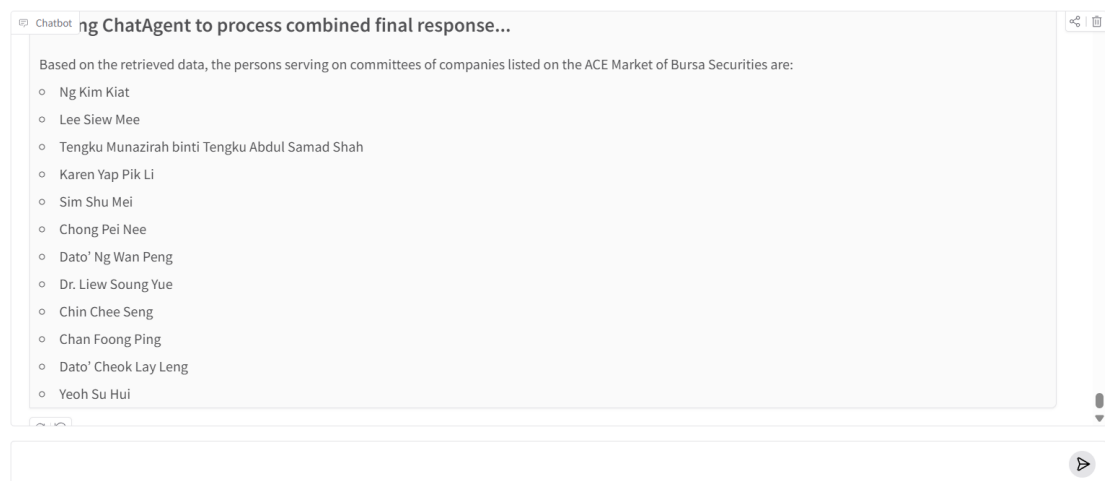


Figure 6.17 Response of the Graph Retrieval System to Competency Question Three

Find all Persons serving on committees of companies listed on [Market_Name].		
<b>Response:</b>		
Based on the retrieved data, the persons serving on committees of companies listed on the ACE Market of Bursa Securities are:		
<ul style="list-style-type: none"> <li>• Ng Kim Kiat</li> <li>• Lee Siew Mee</li> <li>• Tengku Munazirah binti Tengku Abdul Samad Shah</li> <li>• Karen Yap Pik Li</li> <li>• Sim Shu Mei</li> <li>• Chong Pei Nee</li> <li>• Dato' Ng Wan Peng</li> <li>• Dr. Liew Soung Yue</li> <li>• Chin Chee Seng</li> <li>• Chan Foong Ping</li> <li>• Dato' Cheok Lay Leng</li> <li>• Yeoh Su Hui</li> </ul>		
<b>Evaluation Report:</b>		
Criterion	Metric	Justification
Comprehensiveness	Excellent	The response provides a complete list of all individuals serving on committees as requested, fully addressing the question.
Diversity	Poor	The response consists solely of a list without additional context, explanation, or multiple perspectives.
Empowerment	Fair	While it identifies the committee members, it lacks guidance on how to use this information or next steps.
Directness	Excellent	The answer is concise and directly fulfills the request without extraneous information.

*Figure 6.18 Evaluation Report of the Graph Retrieval System Response to Competency Question Three*

All responses to the competency questions presented above are generated solely by the Graph RAG Agent in a single call, meaning that no subsequent calls are made by the Chat Agent to enrich the responses.

The responses exhibit limitations in terms of diversity, as the final outputs returned by the Chat Agent to users are overly compact. An inspection of the response generation process indicates that the root cause lies in the output from the Query Agent to the Chat Agent. Specifically, the Query Agent tends to filter out factual details, preserving only the most relevant points, even though richer details are returned by the Retrieval Result Compilation Agent. Consequently, further prompt tuning for the Query Agent is required to produce more detailed and informative responses.

Competency Questions One and Two are relatively straightforward and can be addressed by most LLMs with Internet access. In contrast, the primary contribution of the proposed corporate analysis module is illustrated by Competency Question Three: “Find all Persons serving on

committees of companies listed on [Market\_Name].” This question highlights the system’s competitive advantage, as conventional LLMs lack an internal graph database capable of representing such interconnections, which enables the derivation of insights. Answering such questions typically requires processing numerous documents, whereas the proposed graph-based RAG system needs only a Cypher query, since the graph is compact due to the deduplication process. Currently, the Chat Agent returns only a limited set of answers because the graph database contains data for just five companies.

Although the responses are not yet optimal, the potential of the knowledge graph is partially illustrated through Competency Question Three. Further refinement of the Query Agent is necessary to ensure that the responses are semantically richer and more comprehensive.

### 6.3 Project Challenges

Several challenges were identified during the development and evaluation of the proposed system. These include: (i) modelling an ontology that is both valid and purpose-driven, (ii) limitations arising from the static nature of the ontology, and (iii) difficulties in prompting the LLM to perform specific tasks reliably.

Although the ontology evaluation process demonstrates the potential for automating ontology construction, the proposed module is not yet enterprise-ready. One contributing factor appears to be the degradation of LLM performance when handling large contexts. This issue emerged even before reaching the maximum context window, as the ontology expanded. During development, the largest ontology contained approximately 100+ relationships. At this scale, the ontology enhancement process often produced suboptimal results, including isolated entities and relationships, weak definitions, and poor LLM guidance. These issues persisted even when the ontology enhancement loop was executed manually multiple times. Furthermore, the Ontology Evaluation Agent frequently failed to detect or flag such problems. To mitigate this, engineering techniques such as ontology segmentation—similar to that applied in the entity–relationship extraction phase—should be explored if LLMs continue to exhibit such limitations.

The second challenge concerns the static nature of the current ontology construction pipeline, which conflicts with the inherently dynamic environment being modelled. In the existing architecture, significant changes to the ontology structure render previously extracted entities,

relationships, and graph retrieval incompatible with the updated ontology. This limitation delayed graph construction during the project, as it was difficult to determine when the ontology was sufficiently stable for use. Without addressing this issue, the system cannot be considered commercial-ready.

The third challenge lies in tuning prompts for the agents. It is often unclear whether a task should be framed declaratively or imperatively, and identifying the optimal strategy requires extensive trial and error. Declarative prompts grant the LLM greater flexibility to leverage its general knowledge, which may compensate for human gaps in domain expertise. However, this flexibility can also introduce unexpected behaviours, potentially leading to system failures despite the use of core constraints. Conversely, imperative prompts yield more predictable behaviour but risk overfitting, as the LLM may adhere too rigidly to instructions, resulting in degraded performance. This challenge is further compounded by the fact that different LLMs exhibit varying behaviours under declarative versus imperative prompting, posing additional barriers to enterprise adoption.

### **6.4 Objectives Evaluation**

The main objective of the proposed project is to develop a corporate insight derivation module powered by LLMs for fundamental analysis. To support this objective, three sub-objectives were defined. The evaluation against these sub-objectives is presented below.

#### **1. Development of an Automated Ontology Construction Module**

The proposed Ontology Construction Module consists of two core processes: ontology extension and ontology enhancement. The incorporation of ontology design principles into the agents within this module reduced the manpower required to construct a usable ontology in an ontology-grounded, graph-based RAG system. Although the current implementation is constrained by scalability issues with large ontologies and by the static nature of the ontology, the project has demonstrated the feasibility of automation. This is evidenced by the successful construction of a knowledge graph that supports Cypher-based retrieval via the proposed Ontology Construction Module.

#### **2. Development of a Knowledge Graph Construction Module**

The proposed Graph Construction Module successfully produced a knowledge graph that is scalable, as it can integrate information from heterogeneous data sources;

auditable, as all retrieved data can be traced back to Cypher queries; and time-aware, as temporal information is encoded within the attributes of modelled entities and relationships. Notably, the proposed entity deduplication process was highly effective, resulting in a more compact and coherent graph. Together, these contributions enabled the delivery of a knowledge graph capable of deriving implicit knowledge from explicit data. Hence, this sub-objective is considered achieved.

### **3. Development of a Text-to-Cypher Retrieval Module**

The proposed Text-to-Cypher Retrieval Module demonstrated the ability to query and retrieve data from the constructed knowledge graph. Although the final responses were not always optimal due to limitations in the prompting strategy adopted by one of the agents in this module, the overall functionality required to meet this sub-objective has been achieved.

In summary, while there remains room for improvement in the proposed pipeline, the three sub-objectives collectively contribute to the achievement of the main objective: the development of a corporate insight derivation module capable of supporting retail investors in performing fundamental analysis.



### Chapter 7 Conclusion and Recommendations

#### 7.1 Conclusion

Fundamental analysis is a critical tool for retail investors engaged in long-term investment, as the long-term value of a stock is ultimately driven by its profitability, and profitability forms the core of fundamental analysis. At its essence, fundamental analysis relies on the ability to derive implicit insights (e.g., future growth potential, governance quality, or operational resilience) from explicit data such as financial disclosures, regulatory filings, and corporate announcements. However, retail investors often struggle to perform such analysis effectively due to limited expertise, resources, and analytical experience. To address this gap, a corporate insight derivation module is proposed to support retail investors in conducting fundamental analysis.

To power the corporate insight derivation module, a graph-based Retrieval-Augmented Generation (RAG) approach is adopted, as its architecture inherently excels at uncovering implicit insights from explicit data through structured reasoning over interconnected information. Nevertheless, contemporary graph-based RAG systems, while strong in certain aspects, also reveal limitations in ontology grounding, scalability, and retrieval robustness. To overcome these challenges, a novel ontology-grounded graph-based RAG pipeline with text-to-Cypher retrieval is proposed, incorporating an automated ontology construction process to realise ontology grounding in practice. The proposed pipeline leverages company disclosures from Bursa Malaysia as input and comprises three sub-modules: the Ontology Construction Module, the Graph Construction Module, and the Graph Retrieval Module. Together, these modules utilise state-of-the-art Large Language Models (LLMs) to achieve the central goal of deriving implicit insights from explicit data.

Evaluation results indicate that although each sub-module has certain limitations, the proposed pipeline demonstrates significant potential for implicit insight derivation. In particular, the Entity Deduplication process within the Graph Construction Module achieved strong performance, with a maximum deduplication rate of 73.0% and an overall rate of 66.5%. The deduplication rate is defined as the proportion of entities deduplicated within a company relative to the total number of entities for that company, as duplication is measured on a per-

company basis. These results highlight the pipeline’s capacity to produce a compact and coherent graph, which is crucial for efficient insight derivation.

The evaluation also surfaced key challenges that must be addressed before the pipeline can be considered commercial-ready, including ontology scalability, dynamic adaptability, and prompt robustness. Nonetheless, the deliverables of this project establish a strong foundation for further enhancements. With continued refinement, the proposed corporate insight derivation module holds the potential to become a practical tool for retail investors, enabling them to conduct fundamental analysis more effectively by systematically transforming explicit disclosures into implicit insights.

### **7.2 Recommendation**

Despite the potential of the proposed ontology-grounded graph-based RAG pipeline with text-to-Cypher retrieval in powering the corporate insight derivation module, several areas remain for improvement to achieve a more robust and commercial-ready system.

First, the system can be enhanced to allow the constructed knowledge graph to adapt to changes in the ontology, regardless of the extent of these changes. Currently, the ontology is static; any modifications result in incompatibility with the knowledge graph. Consequently, newly extracted entities and relationships defined after an ontology update cannot be integrated, and the Text2Cypher Agent, responsible for generating Cypher queries, may fail to function correctly. While this limitation may be acceptable in domains where the ontology rarely changes, it is not aligned with the dynamic nature of the corporate environment, which is essential for effective insight derivation.

Second, the ontology enhancement loop, while effective for relatively small ontologies (fewer than 100 relationships), becomes less reliable as the ontology grows. The current method, which provides the entire ontology to the LLM, encounters performance degradation with larger ontologies, even when the LLM’s context window is not fully utilised. To address this, an extended approach to the ontology enhancement loop, such as ontology segmentation, should be explored.

Finally, the current entity deduplication process—which combines a cache-based approach, similarity search for candidate filtering, and LLM-powered verification—achieves strong

results at the company level. However, there is room for improvement: the entity cache is currently company-specific, meaning that an entity appearing in one company's cache cannot be merged with an identical entity in another company's cache. Implementing a global cache could address this limitation and could similarly be applied to relationship deduplication. Together, these improvements would enable the construction of a more compact and coherent knowledge graph.

## REFERENCES

- [1] TD Bank, “What is the stock market?,” TD.com. Accessed: Apr. 25, 2025. [Online]. Available: <https://www.td.com/ca/en/investing/direct-investing/articles/what-is-stock-market>
- [2] N. Petrusheva and I. Jordanoski, “Comparative analysis between the fundamental and technical analysis of stocks,” *J. Process Manag. New Technol.*, vol. 4, no. 2, pp. 26–31, 2016, doi: 10.5937/JPMNT1602026P.
- [3] Pepperstone, “What is market sentiment and how do you trade it?,” Pepperstone. Accessed: Apr. 25, 2025. [Online]. Available: <https://pepperstone.com/en-au/learn-to-trade/trading-guides/what-is-market-sentiment/>
- [4] Kotak Securities, “Qualitative vs quantitative analysis in stock valuation: Key differences and insights,” Kotak Securities. Accessed: Apr. 26, 2025. [Online]. Available: <https://www.kotaksecurities.com/stockshaala/introduction-to-fundamental-analysis/qualitative-vs-quantitative-analysis-in-stock-valuation/>
- [5] OpenAI, “Introducing OpenAI o3 and o4-mini,” OpenAI. Accessed: Apr. 26, 2025. [Online]. Available: <https://openai.com/index/introducing-o3-and-o4-mini/>
- [6] K. Sharma, P. Kumar, and Y. Li, “OG-RAG: Ontology-grounded retrieval-augmented generation for large language models,” *arXiv*, Dec. 2024, doi: 10.48550/arXiv.2412.15235.
- [7] IBM, “What is a knowledge graph?,” IBM. Accessed: Apr. 27, 2025. [Online]. Available: <https://www.ibm.com/think/topics/knowledge-graph>
- [8] T. W. Teo, S. Y. Liew, C. H. Chng, J. Y. Ng, and Z. Z. Khoo, “GPT-4 powered virtual analyst for fundamental stock investment by leveraging qualitative data,” in *Proc. 5th Int. Conf. Artif. Intell. Data Sci. (AiDAS)*, Bangkok, Thailand, Sep. 2024, pp. 304–309, doi: 10.1109/aidas63860.2024.10730589.
- [9] D. Edge et al., “From local to global: A graph RAG approach to query-focused summarization,” *arXiv*, Feb. 2025, doi: 10.48550/arXiv.2404.16130.
- [10] Microsoft, “Overview - GraphRAG,” GraphRAG. Accessed: Apr. 29, 2025. [Online]. Available: <https://microsoft.github.io/graphrag/query/overview/>
- [11] B. Li, H. Trinh, D. Edge, and J. Larson, “GraphRAG: Improving global search via dynamic community selection,” Microsoft Research. Accessed: Apr. 29, 2025. [Online]. Available: <https://www.microsoft.com/en-us/research/blog/graphrag-improving-global-search-via-dynamic-community-selection/>
- [12] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, “LightRAG: Simple and fast retrieval-augmented generation,” *arXiv*, Apr. 2025, doi: 10.48550/arXiv.2410.05779.

## REFERENCES

- [13] P. Rasmussen, P. Paliychuk, T. Beauvais, J. Ryan, and D. Chalef, “Zep: A temporal knowledge graph architecture for agent memory,” arXiv, Jan. 2025, doi: 10.48550/arXiv.2501.13956.
- [14] Assessing the Capability of Large Language Models for Domain-Specific Ontology Generation. [Online]. Available: [add URL / publisher info if found]
- [15] From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. [Online]. Available: [add URL / publisher info if found]
- [16] W3C, “OWL,” World Wide Web Consortium. Accessed: Apr. 29, 2025. [Online]. Available: <https://www.w3.org/OWL/>
- [17] MongoDB, “Document Databases Basics,” MongoDB. Accessed: Apr. 29, 2025. [Online]. Available: <https://www.mongodb.com/resources/basics/databases/document-databases>
- [18] Pinecone, “Learn: Vector Database,” Pinecone. Accessed: Apr. 29, 2025. [Online]. Available: <https://www.pinecone.io/learn/vector-database/>
- [19] Neo4j, “Getting Started with Graph Database,” Neo4j. Accessed: Apr. 29, 2025. [Online]. Available: <https://neo4j.com/docs/getting-started/graph-database/>

## APPENDIX A

The following illustrates the ontology used for graph construction and retrieval.

### Entities:

#### 1. Company

- definition: A publicly listed corporate entity on Malaysia's Main or ACE Market.
- llm-guidance: When to use: Referencing companies listed on Bursa Malaysia's Main or ACE Market. Format: Full company name.
- examples: Autocount Dotcom Berhad

#### 2. Person

- definition: A natural person who interacts with a company, including corporate officers, directors, advisors, investors, or other individuals.
- llm-guidance: When to use: Identifying any individual related to a listed company by name. Format: Full personal name.
- examples: Choo Chin Peng, Ng Wan Peng

#### 3. Committee

- definition: A formal committee established by a company's board to oversee specific functions such as audit, remuneration, or nomination.
- llm-guidance: When to use: Referring to board committees by their official names. Format: Full committee name.
- examples: Audit and Risk Management Committee, Remuneration Committee

#### 4. Market

- definition: A securities market or exchange where companies are listed.
- llm-guidance: When to use: Referring to listing venues for public companies. Format: Market name.
- examples: ACE Market of Bursa Securities

## 5. Country

- definition: A sovereign state or territory in which the company operates or plans to expand its business activities.
- llm-guidance: When to use: Specifying geographic markets or jurisdictions. Format: Standard country name.
- examples: Malaysia, Singapore

## 6. BusinessSegment

- definition: A distinct aggregated line of business or market area in which a company operates at a strategic level.
- llm-guidance: When to use: Denoting principal high-level business areas. Format: Concise description of the segment.
- examples: Financial management software

## 7. Currency

- definition: A medium of exchange or monetary unit used in transactions by a company.
- llm-guidance: When to use: Referencing currencies in which a company's revenues or purchases are denominated. Format: ISO 4217 three-letter currency code.
- examples: MYR, SGD, USD

## 8. Certification

- definition: A formal recognition of compliance with a standard or status, awarded to a company.
- llm-guidance: When to use: Referencing a certification or official status granted to a company. Format: [Standard/Status] certification.
- examples: ISO/IEC 27001:2013 certification, MSC Malaysia certification

## 9. ShareClass

- definition: A classification of shares issued by a company, delineating rights and privileges attached to each class.
- llm-guidance: When to use: Referencing the category of shares issued by a company. Format: Share class name as stated in corporate disclosures.
- examples: ordinary shares

#### 10. ShareOption

- definition: A contractual right granted by a company allowing a specified person to subscribe for a defined number of its shares at predetermined terms.
- llm-guidance: When to use: Referring to any share option scheme or individual option grants by a company to persons. Format: Description of option terms, including number of shares and exercise price.
- examples: Option to subscribe to 100,000 ordinary shares at RM0.50 per share

#### 11. ConvertibleSecurity

- definition: A debt or equity instrument issued by a company that is convertible into a specified number of its shares under defined terms.
- llm-guidance: When to use: Referencing any warrants, convertible debentures, or similar instruments issued by a company. Format: Description including type of instrument and conversion terms.
- examples: 5-year convertible debentures convertible into ordinary shares at RM0.40 per share

#### 12. CDSAccount

- definition: An account established by Bursa Malaysia Depository Sdn Bhd to record a depositor's securities and enable dealings in such securities.
- llm-guidance: When to use: Referencing an individual's or entity's CDS Account required for trading. Format: Numeric account identifier.
- examples: 123-456789-0



### 13. NonListedOrganization

- definition: A non-listed corporate or institutional entity, excluding government bodies, that interacts with a Company.
- llm-guidance: When to use: Referencing any non-listed firm or corporate body (excluding government agencies) interacting with a listed company. Format: Full official name.
- examples: Skrine, Baker Tilly Monteiro Heng PLT

### 14. TaxIncentive

- definition: A fiscal benefit granted by a regulatory authority to a company for a specified period.
- llm-guidance: When to use: For instances of corporate tax incentives such as pioneer status exemptions. Format: [type of incentive] incentive of [amount] for [period].
- examples: Pioneer status incentive of RM524,996 for FYE 2020

### 15. CorporateAction

- definition: A corporate event undertaken by a company, capturing actions like share splits, consolidations, capital reductions, or capitalisations.
- llm-guidance: When to use: Referencing corporate action events. Format: [Action type] details including ratio or amount and date.
- examples: Share split 35:100 on 28July 2022

### 16. Contract

- definition: A formal agreement entered by a Company with another party, specifying terms such as scope, tenure, payment, and responsibilities.
- llm-guidance: When to use: Referencing any formal agreement, contract, or purchase order entered by a Company. Format: Unique contract identifier or concise descriptive title including counterparty and key terms.

- examples: Yearly maintenance contract with Angkatan Koperasi Kebangsaan Malaysia Berhad (3-year tenure)

#### 17. FinancialPeriod

- definition: A fiscal reporting period for a company, corresponding to a financial year or a specified period under review.
- llm-guidance: When to use: Referencing specific reported financial periods in corporate disclosures. Format: 'FYE YYYY' or 'FPE YYYY'.
- examples: FYE 2021, FPE 2024

#### 18. GovernmentAgency

- definition: A government body empowered to enact policies, regulations, issue approvals and licenses across different sectors.
- llm-guidance: When to use: Referencing government bodies that implement policies or regulations or issue approvals affecting companies. Format: Full official agency name.
- examples: Bank Negara Malaysia, Construction Industry Development Board

#### 19. EconomicIndicator

- definition: A measurable economic statistic or metric that reflects the status of the economy and influences business environments.
- llm-guidance: When to use: Indicating specific economic metrics that impact company operations. Format: Name of the economic indicator.
- examples: inflation rate, GDP, unemployment rate

#### 20. License

- definition: A formal authorization granted by a government agency permitting a company to undertake specified activities, often with conditions or grade classifications.
- llm-guidance: When to use: Referencing specific licenses or registration grades a company holds. Format: "Grade [number] [type] license".

- examples: Grade 7 contractor license

#### 21. RemunerationBand

- definition: A categorical salary range representing the aggregate remuneration and material benefits-in-kind paid to a person for services rendered to a company.
- llm-guidance: When to use: Referencing disclosed remuneration ranges for company directors or key management. Format: "LowerBound - UpperBound".
- examples: 200,001 - 250,000, 0 - 50,000

#### 22. RegulatoryProvision

- definition: A specific statutory provision or section within an act or regulation that governs or exempts corporate transactions.
- llm-guidance: When to use: Referencing particular sections of regulatory acts. Format: Section number followed by act name and year.
- examples: Section 212(8) of the Capital Markets and Services Act, 2007

#### 23. RegulatoryRequirement

- definition: A formal requirement or condition imposed by a regulatory authority on listed companies, such as equity thresholds or accreditation programmes.
- llm-guidance: When to use: Referencing specific regulatory requirements or conditions that companies must meet. Format: Descriptive name of the requirement.
- examples: Equity Requirement for public listed companies

#### 24. OnlinePlatform

- definition: A digital platform, typically accessed via the internet, owned or operated by a company to facilitate the sale of products or services.

- llm-guidance: When to use: Referencing a company's proprietary e-commerce platform or portal. Format: Full domain name or platform name.

- examples: www.komputermurah.my, Sahabatku

#### 25. ProductServiceOffering

- definition: A specific individual product or service offering by a Company at the operational level.

- llm-guidance: When to use: Referring to individual products or services offered by a company. Format: Descriptive title of the offering.

- examples: Enterprise software suite, Management consulting, Technical support

#### 26. PartnerProgram

- definition: A formal vendor or technology partner program sponsored by an organization that companies can join.

- llm-guidance: When to use: Referencing specific partner programs in which a company participates. Format: Program name.

- examples: Oracle Gold Partner program, Microsoft Partner Network

#### 27. Asset

- definition: A tangible or intangible resource owned or used by a company for operational or strategic purposes.

- llm-guidance: When to use: Referencing any asset held or used by a Company. Format: Descriptive title including key characteristics.

- examples: CNC grinding machine, Patent for software, Head office building

#### 28. Industry

- definition: A classification representing a sector of economic activity in which a company operates.

- llm-guidance: When to use: Associating a company with its primary industry sector. Format: Industry sector name.

- examples: Semiconductor industry

#### 29. ShariahComplianceStatus

- definition: A classification status indicating whether a company's securities comply with Shariah principles.

- llm-guidance: When to use: Referring to Shariah compliance classification statuses. Format: "Shariah-compliant" or "Non-compliant".

- examples: Shariah-compliant, Non-compliant

#### 30. RevenueModel

- definition: A mechanism through which a company generates revenue from its operations or services.

- llm-guidance: When to use: Referencing the type of revenue generation model used by a company. Format: Descriptive model name or phrase.

- examples: leasing/rental fees, subscription fees

#### 31. CustomerSegment

- definition: A group of customers or market group targeted by a company's offerings.

- llm-guidance: When to use: Specifying distinct customer groups a company targets. Format: Concise segment description.

- examples: retail customers, small and medium enterprises

#### 32. OrgUnit

- definition: An internal department or division within a company responsible for specific functions or operations.

- llm-guidance: When to use: Referring to a company's internal organizational unit. Format: Full unit name.

- examples: Sales and Marketing, Research and Development

### 33. CorporateDocument

- definition: A formal written record or report produced, filed, or issued by a company, such as regulatory filings, prospectuses, or annual reports. Excludes contractual agreements.
- llm-guidance: When to use: Referencing corporate documents filed or published by a company. Format: [Document type] [date or year].
- examples: Prospectus dated 28 April 2017, Annual Report 2022

#### Relationships:

##### 1. hasBusinessSegment

- source: Company
- target: BusinessSegment
- llm-guidance: When to use: Associating a company with one of its principal business segments.
- examples: Autocount Dotcom Berhad hasBusinessSegment Financial management software

##### 2. hasCurrency

- source: Company
- target: Currency
- llm-guidance: When to use: Indicating a currency in which a company transacts or reports.
- examples: Autocount Dotcom Berhad hasCurrency MYR

##### 3. hasCertification

- source: Company
- target: Certification
- llm-guidance: When to use: Linking a company to a certification it has obtained.
- examples: Autocount Dotcom Berhad hasCertification ISO/IEC 27001:2013 certification

4. issuesShareClass

- source: Company
- target: ShareClass
- llm-guidance: When to use: Indicating a class of shares issued by a company.
- examples: Autocount Dotcom Berhad issuesShareClass ordinary shares

5. grantsShareOption

- source: Company
- target: ShareOption
- llm-guidance: When to use: Linking a company to a share option grant.
- examples: Autocount Dotcom Berhad grantsShareOption Option to subscribe to 100,000 ordinary shares at RM0.50 per share

6. issuesConvertibleSecurity

- source: Company
- target: ConvertibleSecurity
- llm-guidance: When to use: Linking a company to a convertible security it has issued.
- examples: Autocount Dotcom Berhad issuesConvertibleSecurity 5-year convertible debentures convertible into ordinary shares at RM0.40 per share

7. hasCdsAccountForPerson

- source: Person
- target: CDSAccount
- llm-guidance: When to use: Linking a person to their CDS Account.
- examples: Choo Chin Peng hasCdsAccountForPerson 123-456789-0

8. hasCdsAccountForNonListedOrganization

- source: NonListedOrganization
- target: CDSAccount

- llm-guidance: When to use: Linking a non-listed organization to its CDS Account.

- examples: Baker Tilly Monteiro Heng PLT  
hasCdsAccountForNonListedOrganization 987-654321-0

#### 9. hasIndustry

- source: Company
- target: Industry
- llm-guidance: When to use: Associating a company with its primary industry sector.
- examples: Autocount Dotcom Berhad hasIndustry Semiconductor industry

#### 10. hasTaxIncentive

- source: Company
- target: TaxIncentive
- llm-guidance: When to use: Linking a company to a tax incentive it has received.
- examples: Autocount Dotcom Berhad hasTaxIncentive Pioneer status incentive of RM524,996 for FYE 2020

#### 11. undertakesCorporateAction

- source: Company
- target: CorporateAction
- llm-guidance: When to use: Linking a company to a corporate action event it has executed.
- examples: Autocount Dotcom Berhad undertakesCorporateAction Share split 35:100 on 28July 2022

#### 12. entersContract

- source: Company
- target: Contract
- llm-guidance: When to use: Linking a company to a formal contract it has entered.



- examples: Autocount Dotcom Berhad entersContract Yearly maintenance contract with Angkatan Koperasi Kebangsaan Malaysia Berhad (3-year tenure)

### 13. hasContractCounterpartyPerson

- source: Contract
- target: Person
- llm-guidance: When to use: Linking a Contract to a natural person counterparty involved in the agreement.
- examples: Yearly maintenance contract with Angkatan Koperasi Kebangsaan Malaysia Berhad hasContractCounterpartyPerson Choo Chin Peng

### 14. hasContractCounterpartyCompany

- source: Contract
- target: Company
- llm-guidance: When to use: Linking a Contract to a listed company counterparty involved in the agreement.
- examples: Supply agreement contract CA-2024 hasContractCounterpartyCompany Petronas

### 15. hasContractCounterpartyNonListedOrganization

- source: Contract
- target: NonListedOrganization
- llm-guidance: When to use: Linking a Contract to a non-listed organization counterparty involved in the agreement.
- examples: Yearly maintenance contract with Angkatan Koperasi Kebangsaan Malaysia Berhad hasContractCounterpartyNonListedOrganization Angkatan Koperasi Kebangsaan Malaysia Berhad

### 16. reportsFinancialPeriod

- source: Company

- target: FinancialPeriod
- llm-guidance: When to use: Indicating the financial period for which a company reports.
- examples: Autocount Dotcom Berhad reportsFinancialPeriod FYE 2021

17. hasLicense

- source: Company
- target: License
- llm-guidance: When to use: Linking a company to a license it holds.
- examples: Autocount Dotcom Berhad hasLicense Grade 7 contractor license

18. issuesLicense

- source: GovernmentAgency
- target: License
- llm-guidance: When to use: Linking a government agency to a license it issues.
- examples: Construction Industry Development Board issuesLicense Grade 7 contractor license

19. influencedByEconomicIndicator

- source: Company
- target: EconomicIndicator
- llm-guidance: When to use: Indicating an economic indicator affecting a company.
- examples: Autocount Dotcom Berhad influencedByEconomicIndicator inflation rate

20. receivesRemunerationBand

- source: Person
- target: RemunerationBand
- llm-guidance: When to use: Linking a person to their disclosed remuneration band.

- examples: Choo Chin Peng receivesRemunerationBand 200,001 – 250,000

#### 21. subjectToRegulatoryProvision

- source: Company
- target: RegulatoryProvision
- llm-guidance: When to use: Indicating a regulatory provision governing a company.
- examples: Autocount Dotcom Berhad subjectToRegulatoryProvision Section 212(8) of the Capital Markets and Services Act, 2007

#### 22. hasRegulatoryRequirement

- source: Company
- target: RegulatoryRequirement
- llm-guidance: When to use: Linking a company to a regulatory requirement it must meet.
- examples: Autocount Dotcom Berhad hasRegulatoryRequirement Equity Requirement for public listed companies

#### 23. operatesOnlinePlatform

- source: Company
- target: OnlinePlatform
- llm-guidance: When to use: Linking a company to its online platform.
- examples: Autocount Dotcom Berhad operatesOnlinePlatform [www.komputermurah.my](http://www.komputermurah.my)

#### 24. hasRevenueModel

- source: Company
- target: RevenueModel
- llm-guidance: When to use: Linking a company to a revenue model it employs.
- examples: Autocount Dotcom Berhad hasRevenueModel leasing/rental fees

25. targetsCustomerSegment

- source: Company
- target: CustomerSegment
- llm-guidance: When to use: Indicating a customer segment targeted by a company.
- examples: Autocount Dotcom Berhad targetsCustomerSegment retail customers

26. hasOrgUnit

- source: Company
- target: OrgUnit
- llm-guidance: When to use: Linking a company to one of its internal organizational units.
- examples: Autocount Dotcom Berhad hasOrgUnit Sales and Marketing

27. hasCorporateDocument

- source: Company
- target: CorporateDocument
- llm-guidance: When to use: Linking a company to a corporate document it produces or files.
- examples: Autocount Dotcom Berhad hasCorporateDocument Prospectus dated 28 April 2017

28. participatesInPartnerProgram

- source: Company
- target: PartnerProgram
- llm-guidance: When to use: Linking a company to a partner program it joins.
- examples: Autocount Dotcom Berhad participatesInPartnerProgram Microsoft Partner Network

29. hasProductServiceOffering

- source: Company

- target: ProductServiceOffering
- llm-guidance: When to use: Linking a company to a product or service offering.
- examples: Autocount Dotcom Berhad hasProductServiceOffering Enterprise software suite

### 30. hasAsset

- source: Company
- target: Asset
- llm-guidance: When to use: Linking a company to an asset it holds.
- examples: Autocount Dotcom Berhad hasAsset CNC grinding machine

### 31. listedOn

- source: Company
- target: Market
- llm-guidance: When to use: Linking a company to the market on which it is listed.
- examples: Autocount Dotcom Berhad listedOn ACE Market of Bursa Securities

### 32. operatesIn

- source: Company
- target: Country
- llm-guidance: When to use: Linking a company to a country in which it operates.
- examples: Autocount Dotcom Berhad operatesIn Malaysia

### 33. hasCommittee

- source: Company
- target: Committee
- llm-guidance: When to use: Linking a company to a board committee it has established.

- examples: Autocount Dotcom Berhad hasCommittee Audit and Risk Management Committee

34. hasCommitteeMember

- source: Committee
- target: Person
- llm-guidance: When to use: Linking a committee to a person who is a member of that committee.
- examples: Audit and Risk Management Committee hasCommitteeMember Choo Chin Peng

35. suppliesTo

- source: Company
- target: Company
- llm-guidance: When to use: Use when a company provides products or services as a supplier to another company.
- examples: Top Glove Berhad suppliesTo Yinson Holdings Berhad

36. competesWith

- source: Company
- target: Company
- llm-guidance: When to use: Use when two companies compete in the same market or segment.
- examples: Petronas competesWith Shell

37. personHoldsEquityInterest

- source: Person
- target: Company
- llm-guidance: When to use: Indicating an equity stake held by a person in a company.
- examples: Choo Chin Peng personHoldsEquityInterest Autocount Dotcom Berhad

### 38. organizationHoldsEquityInterest

- source: NonListedOrganization
- target: Company
- llm-guidance: When to use: Indicating an equity stake held by an organization in a company.
- examples: Baker Tilly Monteiro Heng PLT  
organizationHoldsEquityInterest Autocount Dotcom Berhad

### 39. hasShariahComplianceStatus

- source: Company
- target: ShariahComplianceStatus
- llm-guidance: When to use: Indicating the Shariah compliance classification status of a company.
- examples: Autocount Dotcom Berhad hasShariahComplianceStatus  
Shariah-compliant

### 40. hasRelatedPartyTransactionWithPerson

- source: Company
- target: Person
- llm-guidance: When to use: Indicating that a company has conducted a related party transaction with a person.
- examples: Autocount Dotcom Berhad  
hasRelatedPartyTransactionWithPerson Choo Chin Peng

### 41. hasRelatedPartyTransactionWithOrganization

- source: Company
- target: NonListedOrganization
- llm-guidance: When to use: Indicating that a company has conducted a related party transaction with an organization.
- examples: Autocount Dotcom Berhad  
hasRelatedPartyTransactionWithOrganization Baker Tilly Monteiro Heng  
PLT

42. personHasShareOption

- source: Person
- target: ShareOption
- llm-guidance: When to use: Linking a person to a specific share option grant.
- examples: Choo Chin Peng personHasShareOption Option to subscribe to 100,000 ordinary shares at RM0.50 per share

43. hasStrategicPartnership

- source: Company
- target: Company
- llm-guidance: When to use: Indicating a strategic partnership between two companies.
- examples: Autocount Dotcom Berhad hasStrategicPartnership CESB

44. issuesRegulatoryProvision

- source: GovernmentAgency
- target: RegulatoryProvision
- llm-guidance: When to use: Linking a government agency to a regulatory provision it issues.
- examples: Securities Commission Malaysia issuesRegulatoryProvision Section 212(8) of the Capital Markets and Services Act, 2007

45. imposesRegulatoryRequirement

- source: GovernmentAgency
- target: RegulatoryRequirement
- llm-guidance: When to use: Linking a government agency to a regulatory requirement it imposes.
- examples: Securities Commission Malaysia imposesRegulatoryRequirement Equity Requirement for public listed companies

46. hasCompanySecretary



- source: Company
- target: NonListedOrganization
- llm-guidance: When to use: Linking a company to the non-listed organization serving as its company secretary.
- examples: Autocount Dotcom Berhad hasCompanySecretary Baker Tilly Monteiro Heng PLT

#### 47. grantsTaxIncentive

- source: GovernmentAgency
- target: TaxIncentive
- llm-guidance: When to use: Linking a government agency to a tax incentive it has granted.
- examples: Malaysian Investment Development Authority grantsTaxIncentive Pioneer status incentive of RM524,996 for FYE 2020

#### 48. holdsPositionAtCompany

- source: Person
- target: Company
- llm-guidance: When to use: Linking a person to the position they hold at a specific company, encoding the position title in the instance label.
- examples: Choo Chin Peng holdsPositionAtCompany Autocount Dotcom Berhad as Chief Financial Officer

#### 49. worksIn

- source: Person
- target: OrgUnit
- llm-guidance: When to use: Linking a person to an internal organizational unit within a company.
- examples: Choo Chin Peng worksIn Sales and Marketing

# FUNDAMENTAL STOCK ANALYSIS WITH LLMs AND QUALITATIVE DATA

## Development of Ontology-Grounded Graph-Based RAG with Text-to-Cypher Retrieval for Malaysian Listed Companies



### Background Information

#### Fundamental Analysis

Fundamental analysis can help retail investors make long-term investments in the stock market, and its essence lies in deriving implicit insights from explicit data—for example, a company may seem untouched by a policy, but its clients' gains from incentives could indirectly raise its revenue.

### Problem Statement

#### Difficulty in Deriving Implicit Insights

Retail investors lack:

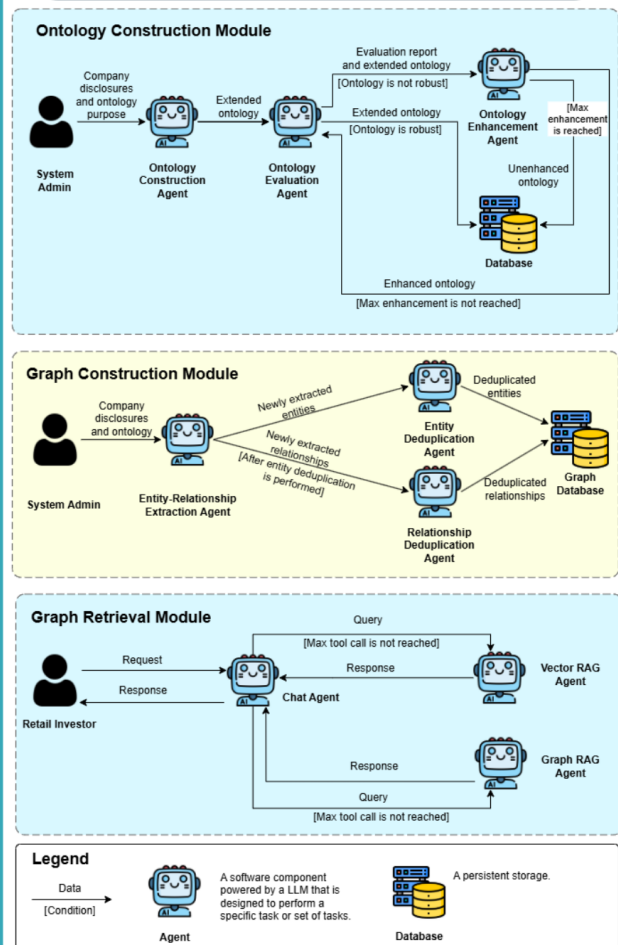
1. Domain expertise
2. Resources
3. Analytical experience

### Proposed Solution

#### Corporate Insights Derivation Module

A corporate insights module powered by a novel graph-based RAG pipeline—an ontology-grounded, graph-based RAG with text-to-Cypher retrieval that requires minimal involvement in ontology construction.

### Proposed Graph-Based RAG Architecture



Kam Chee Qin 22ACB05553  
Supervised by Prof. Ts. Dr. Liew Soung Yue

