FUNDAMENTAL STOCK ANALYSIS WITH LLMS AND QUALITATIVE DATA – DEVELOPMENT OF A VECTOR DATABASE FOR COMPANY REPORTS

By

Ting Jun Jing

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

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Prof Ts Dr Liew Soung Yue who has given me this bright opportunity to engage in a stock market analysis and large language models related project. It is my first step to establish a career in artificial intelligence field. A million thanks to you.

I must say thanks to my parents and family for their love, support and continuous encouragement throughout the course. Next, I would also like to thank my friends and teammates for their suggestions and emotional supports that enabled me to complete the project in a timely manner.

ABSTRACT

This project explores the use of natural language processing techniques, specifically Large Language Models (LLMs), for fundamental stock analysis by leveraging qualitative data in corporate financial reports and disclosures. It addresses the challenge of information overload faced by retail investors by automating the collection, processing, and interpretation of fundamental data. The system employs a multi-agent architecture integrating web scraping of financial reports, LLM-based report processing of lengthy documents, and embedding the resulting processed data into a vector database to enable semantic search and efficient information retrieval. Using vector embeddings and retrieval-augmented generation, the system acts as a "virtual analyst" that retrieves relevant information and synthesizes coherent responses to complex investor queries about a company's fundamentals. The results demonstrate that this LLM-driven approach efficiently distills key insights from enormous unstructured texts, thereby making qualitative analysis more accessible and bridging the gap in analytical capability for retail investors. The project provided a functional proof of concept and highlighted opportunities for further improvements, including expanding data sources, improving summary accuracy, and strengthening the system's real-time information integration capabilities.

Area of Study: Application Development, Large Language Models

Keywords: Fundamental Analysis, Natural Language Processing, Web Scraping, Information Retrieval, Semantic Search

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
LLMs	Large Language Models
NLP	Natural Language Processing
ML	Machine Learning
RAG	Retrieval-Augmented Generation
API	Application Programming Interface

CHAPTER 1 INTRODUCTION

In this chapter, we present the background and motivation for our project, highlighting the significance of fundamental analysis in the stock market for individual investors. We outline our contributions to simplifying and automating this process and provide an overview of the project structure.

1.1 Problem Statement and Motivation

Retail investors often struggle with fundamental stock analysis due to the enormous volume and complexity of information available. Critical data is scattered across lengthy financial reports, company prospectuses, news articles, and policy announcements, making it very challenging for an individual to digest and interpret all relevant qualitative information. This problem is compounded by the fact that professional analysts have teams and tools to work through such data, whereas retail investors are usually on their own. As a result, there is an information asymmetry, where some important signals about a company's performance or risks may be overlooked by non-experts, leading to suboptimal investment decisions. Therefore, there is a clear need for a solution that can bridge this gap by automatically gathering and analyzing the financial data that provides a solid foundation for investment analysis.

Recent advances in Artificial Intelligence (AI), especially in Large Language Models (LLMs), provide a significant opportunity to address this challenge. LLMs like OpenAI's ChatGPT and Google's Gemini have demonstrated an incredible ability to understand and generate human-like text, which means they can potentially read and summarize complex financial documents just as a human analyst would. Notably, early research by Lopez-Lira and Tang showed that even a general-purpose LLM (ChatGPT) can interpret financial news headlines and predict short-term stock price movements [1]. This suggests that LLMs can extract meaningful signals from unstructured financial text, motivating their use in aiding stock market analysis. Therefore, the motivation for this project is to leverage LLM's natural language understanding to perform fundamental analysis tasks at scale, which includes scanning through huge amounts of financial reports and news to identify the key insights for investors. By doing so, we aim to create a system that acts as a virtual stock analyst for retail investors, an AI-powered assistant that can quickly and efficiently digest large amounts of information and provide coherent, insightful analysis in a timely manner.

Furthermore, this project is driven by the observation that while quantitative trading algorithms and technical analysis tools are abundant, there is a lack of accessible tools focusing on qualitative analysis of fundamentals for retail investors. As Benjamin Graham famously stated, "In the short run, the market is a voting machine but in the long run, it is a weighing machine," highlighting the importance of fundamental analysis in evaluating a company's intrinsic value beyond short-term market fluctuations [2]. Given the proven importance of qualitative factors in determining a company's intrinsic value, an AI system that can handle such qualitative data could significantly enhance decision-making. In summary, this project exists to solve the problem of information overload in fundamental stock analysis, with the motivation to simplify complex analysis through advanced technology, allowing retail investors to make more informed and confident investment choices.

1.2 Project Scope

This project focuses on the development of a multi-agent AI system for fundamental stock analysis, emphasizing the integration of LLMs with qualitative financial data. This project utilizes fundamental approaches such as evaluating the intrinsic value and business health of companies based on the information about their financials and operating environment.

The system will digest publicly available textual data that are crucial to fundamental analysis, which include corporate disclosures such as annual reports, quarterly reports, company prospectuses, and relevant financial news articles and announcements. The core functionality is to collect, process, analyse qualitative information to produce useful insights about a company's stock. Specifically, the system will identify and extract key fundamental indicators and narratives like business descriptions, management commentary, risk factors, competitive landscape, significant partnerships or supply chain relationships. The analysis will yield outputs such as summarized reports on a company's financial health, identified strength and weaknesses, and potentially an evaluation or rating of the stock's performance. Meanwhile, the system may incorporate some basic quantitative analysis like reading financial statements from reports, its emphasis is on interpreting the textual information rather than performing deep numerical financial modelling from scratch.

This project will implement a multi-agent design, where each different components will handle different tasks in a coordinated way. For instance, one agent may be responsible for continuously fetching new financial documents and news (data collection agent), another may specialize in parsing and summarizing those documents using LLM capabilities (analysis agent) and another may construct a knowledge graph of entities (relationship-mapping agent). These different components will communicate and share information, in the end working together to emulate the comprehensive analysis that a human analyst or team of analyst might perform. By defining these agents and their interactions, the project scope includes designing the workflow for how raw data becomes a refined analysis through the system.

The end deliverable is expected to be a prototype "virtual stock analyst" tool. This includes developing a user interface which a retail investor can query the system to ask for an analysis of a particular company and receive an intelligible, well-structured response from the system. This response might be a written report or answers to specific questions about the company. The system will provide information and analytical opinions to support the user's decision. Therefore, this project will demonstrate the system's capabilities through case studies or examples, such as analysing a sample company and the reaction to a breaking news event that affecting that company.

1.3 Project Objectives

The focus of this project is on the module of developing a vector database for company reports, which plays a crucial role in enabling the system to provide accurate and timely insights about companies. The objectives of this module are divided into three sub-objectives:

- **Develop an Automated Multi-Agent Analysis System**
  Design and implement a multi-agent software system capable of autonomously gathering and processing financial information. This includes agents for

collecting financial reports and company prospectuses, an agent powered by LLMs to parse, analyze and summarize these qualitative data. This architecture should allow these components to work collaboratively, mimicking the workflow of a human analyst team.

- **Leverage LLMs for Qualitative Data Interpretation**
  Utilize state-of-the-art LLMs to perform tasks such as summarizing annual reports and extracting key facts from prospectuses. This LLM-driven components should be able to parse complex financial text which includes charts, tables, diagrams, infographics, and illustrations that cannot be converted directly into text and produce outputs that are accurate and useful.
- **Real-Time Information Integration**
  Enable the system to incorporate real-time or recent information, particularly from the financial reports and announcements. The objective is that the system should be able to update its analysis when new information appears. For instance, when a company releases a new quarterly or annual report at the end of its reporting period, the system's financial-reports agent should automatically fetch it and the LLM analysis agent should immediately parse and analyze the latest disclosures. This ensures that the output remains current and relevant, just like a human analyst would continuously update their view based on new information.

1.4 Impact, Significance and Contribution

This project's expected impact is enormous, both for individual investors and for the broader intersection of AI and finance. Firstly, for retail investors, which having access to this virtual analyst could be game-changing. It can effectively lower the barrier to performing comprehensive research on stocks. Individuals who lack the time or training to read dense financial reports would benefit from this concise AI-generated analyses, leading to a more informed investment decisions. In the long run, this could make investment research more accessible, by narrowing the gap between what institutional investors with large research departments and retail investors can know about a company. If successful, this tool could also empower more people to invest wisely, potentially improving financial outcomes for those who previously might have depended on inadequate and insufficient information.

Besides that, this project is important because it advances the application of LLMs in the financial analysis. While previous works have started to show that LLMs can interpret financial related text, this project takes a novel approach by integrating multiple components of fundamental analysis into one system. The system is no longer for a single-use case like just parsing news sentiment alone or reading one financial report itself. Instead, this system combines multiple sources and tasks into one, which is an ambitious initiative to imitate a comprehensive analyst. This integration is an important contribution because it explores how far we can utilize AI towards complex, multi-step reasoning in a critical fields. A successful implementation could inspire further research into multi-agent AI architectures for other knowledge-intensive fields as well.

1.5 Background Information

Fundamental analysis is the core concept in investing, which refers to the evaluation of a company's intrinsic value and financial health through deep analysis of its financial statements, operational data and the context in which it operates. In general, fundamental analysis involves examining a company's economic and financial reports, that includes all qualitative and quantitative information, to determine an estimate of its true value [3]. This approach looks at factors like revenues, earnings, profit margins, assets and liabilities, as well as the qualitative aspects such as the competence of management, company's competitive advantages and industry trends. The goal is to determine whether the current market price of a company's stock is consistent with its fundamentals or if there is a mispricing, either undervaluation or overvaluation that could be exploited by investors.

To perform fundamental analysis, traditional analysts rely on the financial statements such as income statements, balance sheets, cash flow statements and company disclosures. These provide the quantitative information and are often used to calculate ratios like Return on Equity (ROE), Price-to-Earnings (P/E) and debt-to-equity. This can help in comparing companies and track performance over time.

However, equally important is the qualitative information found in those reports. This includes the company's business model, list of risk factors, and the outlook given by executives. For example, the Management Discussion and Analysis (MD&A) section in every annual reports can give insight into the management's strategy and perspective on the year's results, which is some information not captured in numbers. Not only that, fundamental analysts also need to pay attention to some external factors like industry conditions, macroeconomic indicators and regulatory changes. All of these is very important in forming a complete picture of a company's prospects.

Fundamental analysis is always been a labour-intensive process which manually done by experts. Research analysts at investment firms might have to go through multiple companies, spending weeks or even months reading through each company's reports. Then, they would write research reports by summarizing their findings and providing some of their investment recommendations. As for retail investors, they usually have limited resources and often had to rely on these reports written by the experts or the simplified metrics available on financial websites. This is where advanced computing and artificial intelligence come in, to automate some of the work of these analysts.

In recent years, there were numerous attempts that have introduced automation and quantitative algorithms into stock analysis. Machine Learning (ML) alongside with statistical techniques have been applied to forecasting stock prices and predicting trends using historical numerical data. However, most of this work falls under the technical analysis part, also known as quantitative strategy, where it often ignores the importance of the rich qualitative information that used by fundamental analysts. We have seen more and more efforts that incorporate textual data (news or reports) into models, which through simple natural language processing (NLP) techniques initially. This involves sentiment analysis using dictionaries of positive/negative words and topic modelling of annual report text to identify themes.

Natural language processing models have been well developed and powerful enough to truly understand financial text. Early approaches included domain-specific models like FinBERT (a variant of BERT tuned to finance text) for sentiment classification of news or reports [4]. However, fully interpreting a document in context remained challenging. An important turning point, which is when the introduction of

LLMs, pre-trained on massive text datasets. LLMs were not only trained on financial data, but they appear to have absorbed a lot of financial knowledges from the internet. LLMs can also perform complex language tasks with minimal additional training required. Therefore, LLMs have shown the ability to summarize articles or answer questions about a given text with an understanding of nuance.

In addition to LLMs, the concept of a multi-agent system forms part of our project's background. Multi-agent systems in AI refer to a collection of autonomous agents that interact within an environment to achieve certain goals, collaborating and dividing tasks among themselves. This concept has been used in various fields and is increasingly seen in complex AI applications where single model is insufficient or inefficient. For instance, the process of fundamental analysis in our project can be considered as a pipeline of distinct tasks such as gather data, analyse text, compare information and output conclusions, where each can be handled by a different specialized agents. By having these as separate components, the system can be more modular and each agent can be optimized or fine-tuned for its specific function.

As conclusion, the background of this project sits between finance and AI innovation. AI, especially LLMs technology and multi-agent system design provides the tool that has only recently become powerful enough to tackle those challenges in a comprehensive approach.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

Fundamental stock analysis traditionally involves an in-depth review of financial statements and qualitative disclosures to assess a company's value [2]. In recent years, advances in LLMs have opened up new possibilities for automating this process using unstructured text data like news, reports and transcripts. Since 2023, there has been a significant increase in research applying this state-of-the-art LLMs to tasks ranging from sentiment analysis of news to automated reading of annual reports. This chapter surveys the latest academic and industry work on using LLMs and AI for fundamental stock analysis with qualitative data.

2.2 Market Prediction from News and Sentiment with LLMs

One of the earliest breakthroughs was demonstrating that general LLMs can interpret financial news for stock prediction. Lopez-Lira & Tang showed that ChatGPT's analysis of news headlines can successfully predict short-term stock price movements [1]. In their study, ChatGPT was prompted to evaluate whether a headline was good, bad or irrelevant for a firm. These LLM-generated sentiment scores had significant power to predict upcoming day returns, outperforming traditional sentiment analysis method. However, this effect was strongest for smaller stocks and after negative news, suggesting LLMs capture nuanced information that retail investors underreact to.

In 2024, Wu applied a similar approach in the Chinese market, where news were fed to both ChatGPT and domestic Chinese LLMs (Alibaba's Qwen), to score their impact on stocks [5]. The results obtained is consistent with Lopez-Lira & Tang, such that the daily sentiment scores from LLMs correlated with the next-day returns and enabled profitable backtested strategies. An interesting finding was that the market seems to be more sensitive to negative news, as short strategies using the LLM news score outperform long strategies. These works underscore that news sentiment analysis with LLMs has become a useful tool for return forecasting, often incorporating earlier sentiment methods.

In addition to news headlines, researchers have explored other textual sentiment signals. For example, social media and real-time feeds are a natural extension, although rigorous academic research (after 2023) is still rare. Overall, the general view after 2023 is that LLMs can read the news or reports like an analyst, extracting sentiment and signals that translate into measurable predictability of returns. However, these studies are often limited to short-term forecasts and rely on single data sources, without integrating a deeper fundamental analysis.

2.3 Financial Report and Disclosure Analysis with LLMs

Another additional research approach examines using LLMs to parse company disclosures like annual reports, regulatory filings, and earnings call transcripts, which are the core component to fundamental analysis. These documents are long and complex, making it an ideal candidate for LLM-driven summarization and insight extraction. Kim, Muhn & Nikolaev have lead multiple studies in this area, where one of the study investigated whether ChatGPT could help investors digest large amount of corporate filings [6]. They utilized GPT-3.5 to summarize SEC filings and earnings call transcripts, they found that AI-generated summaries distilled the most relevant information.

Notably, the sentiment of the ChatGPT's summary correlates more strongly with stock's market reaction than the sentiment of the original full document. In other words, LLMs was better at pinpointing the material content driving investor responses, effectively filtering out the "noise" in long reports [6]. The overall conclusion of the study is that generative AI adds considerable value in distilling financial disclosures, making it easier for retail investors to grasp essential information. This is a significant development in qualitative analysis where tasks like reading a 100+ pages reports and assessing its tone, once was the domain of experienced analysts, can now be partially automated with LLMs summarization.

Industry projects have similarly targeted financial statement analysis with LLMs. Gupta introduced GPT-InvestAR, which uses ChatGPT to read annual reports and generate quantitative features for stock picking [7]. In this approach, the LLM produces a summary and key points from each annual report, which are then encoded into a “quant-style” numerical dataset, which are then fed into a machine learning model to predict future stock returns. In forward testing on historical data, the LLM-augmented model achieved promising outperformance over the S&P 500 [7]. Essentially, this pipeline uses LLMS to do the tedious task of reading and distilling fundamental reports for possibly thousands of companies, then uses those findings in a quantitative strategy.

2.4 Earnings Forecasting and Financial Reasoning with LLMs

A key aspect of fundamental analysis is predicting a company’s future earnings or performance. Recent research suggests that LLMs, when prompted correctly, can perform this analytical reasoning at a high level. Another work by Kim, Muhn & Nikolaev, asked whether an LLM could analyse financial statements like a professional analyst to predict earnings changes [8]. In their experiment, they fed GPT-4 a standardized set of financial statements for a company, purposely without any textual narrative or management discussion. The LLM was able to predict the likely direction of future earnings even with just raw numbers. This outperforms human analyst in accuracy and matches the performance of state-of-the-art specialized machine learning models.

LLMs showed particular strength in cases where human analysts often made mistakes, suggesting that it finds subtle patterns or relationships in financial situations [8]. Notably, because the model was not given any data beyond the financial statements, its success suggests it is able to make immediate financial inferences rather than simply repeating learned answers. Besides that, this work is crucial because it shows that LLMs can handle numeric financial analysis tasks that traditionally require expert judgment. This also hints that LLMs, especially advanced ones like GPT-4o, Gemini 2.5 Pro, can integrate quantitative data with qualitative reasoning. One limitation worth noting is that adding too much narrative context to a single prompt increases the risk of model errors or “hallucinations.”

In summary, the application of LLMs to earnings forecasting and reasoning is new but growing. A major advantage observed is the explainability, where LLMs can output a human-readable explanation for its prediction, unlike a typical ML model.

However, challenges remain in ensuring factual accuracy when dealing with long text contexts and integrating real-time data. The key is to orchestrate these capabilities within a robust analytical pipeline, which motivates our proposed multi-agent system.

2.5 LLM-Driven Investment Report Generation and Decision Support

Beyond the specific prediction tasks, some projects have used LLMs to produce full investment analyses or recommendations. For instance, the FinSphere system, which is a conversational agent that answer user queries about stocks by combining real-time financial data with LLM analysis [9]. Their system can generate a detailed stock analysis report including assessments of financial ratios, recent news, and valuation metrics in a conversational format. It integrates an instruction-tuned LLM with quantitative tools and live data feeds. This comprehensive framework has achieved significant improvements in analysis quality and authenticity, outperforming both general models like GPT-4 when they were not similarly enhanced.

Another interesting example is an open-source project, AI Hedge Fund by the user Virattt on GitHub [10]. The project simulates a team of AI-driven investment agents working together, with each agent representing a different investment strategy or role. For instance, they implemented a "Ben Graham Agent" focusing on classic value investing, a "Cathie Wood Agent" seeking growth and innovation, and others modelled after famous investors. Not only that, they also have a "Valuation Agent" that computes intrinsic values, the "Sentiment Agent" that gauges market mood and a "Fundamental Agent" that parses financial metrics. This multi-agent architecture is essentially an AI investment committee that generates and executes investment thesis from multiple perspectives. It illustrates how to scale an LLM-centric system into a complete process, from data extraction to analysis to decision making. This approach addresses the limitations of a single LLM approach by introducing modularity and specialization, which forms like a team of human analysts with different specialties working together.

2.6 Limitations of Prior Work and Research Gaps

Although recent literature is promising, some limitations and gaps remain. Firstly, many studies only focus on isolated narrow sub-tasks. For instance, Lopez-Lira & Tang consider news sentiment for one-day return prediction, but they do not connect this to longer-term fundamental valuation or portfolio construction [1]. This narrow focus means previous solutions often lacked a complete fundamental analysis pipeline. In practice, analysts combine multiple sources of information like news, filings, industry reports, macro data, to form a comprehensive view. However, very few works so far have achieved this level of integration. Multi-agent system like the AI Hedge Fund project is an exception, but even here, this approach is in its early stages and mostly unvalidated with real financial results.

Evaluation and objectivity are also a focus. Some early works performed well and surpassed the market, but these need to be interpreted carefully. For instance, Erdem found a ChatGPT-chosen portfolios beat the S&P 500 over 6 months, but this was attributed to higher risk exposure and did not produce a statistically significant alpha value [11]. Furthermore, LLMs are prone to hallucinations, which involves making up facts, that can be dangerous in the financial world. While the summarization task has demonstrated accuracy, in some cases the LLM may infer incorrect relationships or overlook key nuances. Ensuring the factual correctness and stability of outputs remains a challenge, highlighting the need for interpretability frameworks.

A major limitation of previous studies is that they focus on single-company analysis and do not consider the interconnectedness between companies. Many existing studies treat each company in isolation, ignoring the upstream and downstream linkages that link companies together in practice. This independent-company approach ignores how a company's suppliers, customers, and strategic partners can significantly influence its performance. For example, a shock to one company can be transmitted to other companies through the customer-supplier channel, which means that isolated analysis will miss these ripple effects. Companies in reality are embedded in complex networks of business relationships, so ignoring such inter-company dependencies constitutes a significant research gap. By ignoring the broader corporate ecosystem, including supply chain connections and strategic alliances, prior LLM-driven fundamental analysis may fail to capture important contextual and risk factors that influence stock performance.

2.7 Suggested Improvements and Proposed Solutions

Given the above, several improvements and solutions are suggested. First, a common suggestion is to adopt a modular, multi-step approach to financial analysis with AI. Instead of one long prompt or model trying to do everything, breaking down the task into stages. For example, different stages for information retrieval, analysis and recommendation. This can improve overall accuracy and mange complexity. This also aligns with the multi-agent concept, where each agent can be optimized for its own task. Not only that, ensuring all the modules to share information effectively is an active area of development.

To address the lack of inter-company dependencies, this project proposes the development of a company relationship mapping system as a key advancement. The system will explicitly capture inter-company dependencies by collecting relationship information from financial reports and corporate disclosures, with a focus on identifying upstream suppliers, downstream customers, and strategic partnerships. Conceptually, the mapping will compile a network of companies where the edges represent significant business relationships based on qualitative disclosures. Next, by incorporating this relationship map into the analysis, LLMs can assess how developments in one company affect related companies.

Integrating awareness of inter-company relationships promises to produce a more comprehensive and realistic fundamental analysis than previous single-company approaches. In summary, the proposed relationship mapping technique will extend LLM-based stock analysis beyond isolated entities, enabling the model to consider supply chain and partnership dynamics as part of qualitative analysis.

CHAPTER 3 SYSTEM METHODOLOGY

This chapter details the technical approach adopted to transform raw company documents into a query-ready knowledge base that can be used for fundamental stock analysis. The overall system architecture and modular, pipeline-oriented design principles are outlined. This provides a comprehensive blueprint for how the proposed system will achieve the functional and performance goals.

3.1 System Model

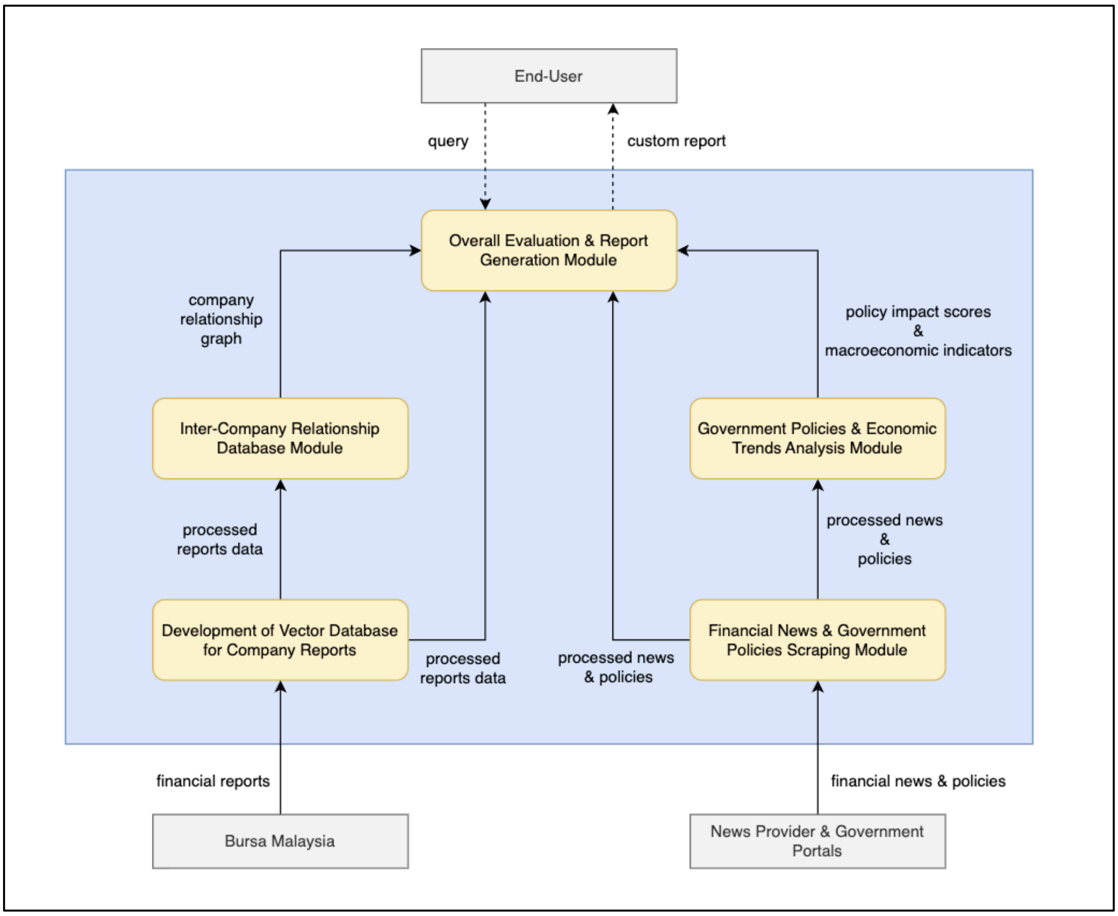


Figure 3.1.1 Overall Model of the System

The Development of a Vector Database for Company Reports is one part of the overall system for "Fundamental Stock Analysis with LLMs and Qualitative Data". It specifically handles the ingestion, processing, storage and retrieval of company reports. In the structure of the broader system, this module provides foundational data that other modules can utilize. For instance, a separate module will use these information from the reports to find relationships between companies, and an overall evaluation module might combine this data with news analysis.

This Development of a Vector Database for Company Reports consists of two tightly integrated components, a Company Report Database Module and a Vector RAG Module. Together, these components transform raw financial documents from Bursa Malaysia into a structured knowledge base and power an intelligent Q&A system. The Company Report Database Module handles data ingestion and preparation, such as making the data well-structured and ready for chunking and embedding, while the Vector RAG Module provides a chat-based interface that performs semantic search and retrieval augmented generation on that vectorized data.

3.1.1 Company Report Database Module

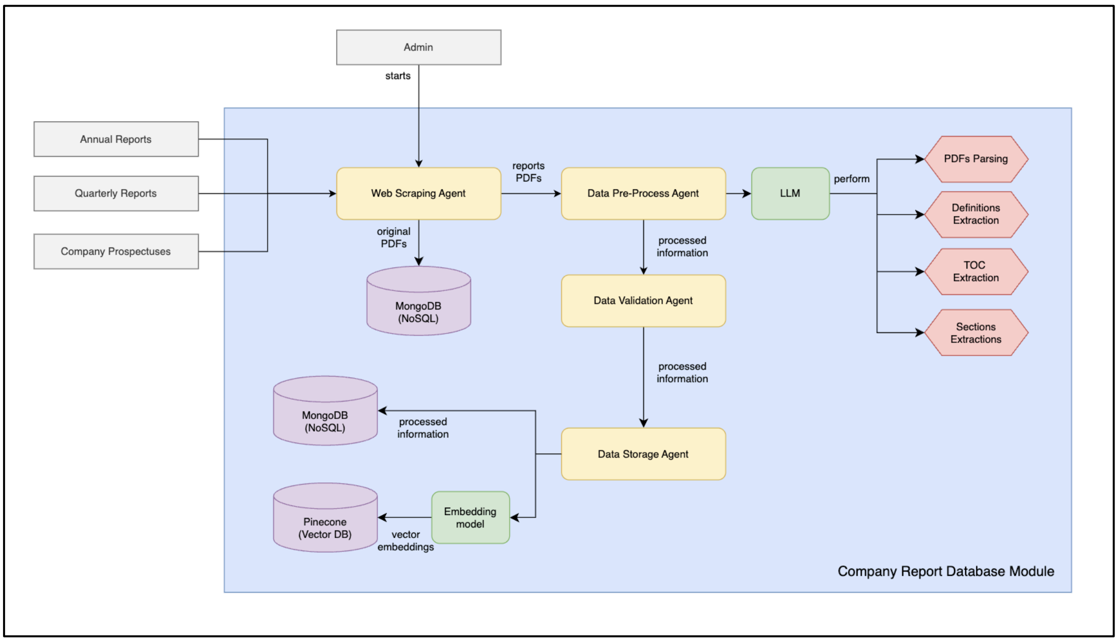


Figure 3.1.1.1 Company Report Database Module Block Diagram

The design of this Company Report Database Module is illustrated in a top-down system block diagram as shown in Figure 3.1.2, which shows all the major components and their interactions. Each component or agent in the diagram is responsible for a distinct functionality, and collaborate together to form a pipeline that ingests Bursa Malaysia reports, preprocess and validates them, and stores both structured data and text embeddings. The module creates a NoSQL document store for original and processed reports data, and a vector index of the textual content for semantic search.

This module is responsible for the collection, processing, storage and retrieval of company reports data. It takes raw corporate disclosures (annual reports, quarterly reports, prospectuses) from Bursa Malaysia and converts them into a query-ready database. The processed data is structured and cleaned to suit subsequent chunking and embedding steps, ensuring it can be effectively loaded into a vector database for semantic search.

3.1.2 Vector RAG Module

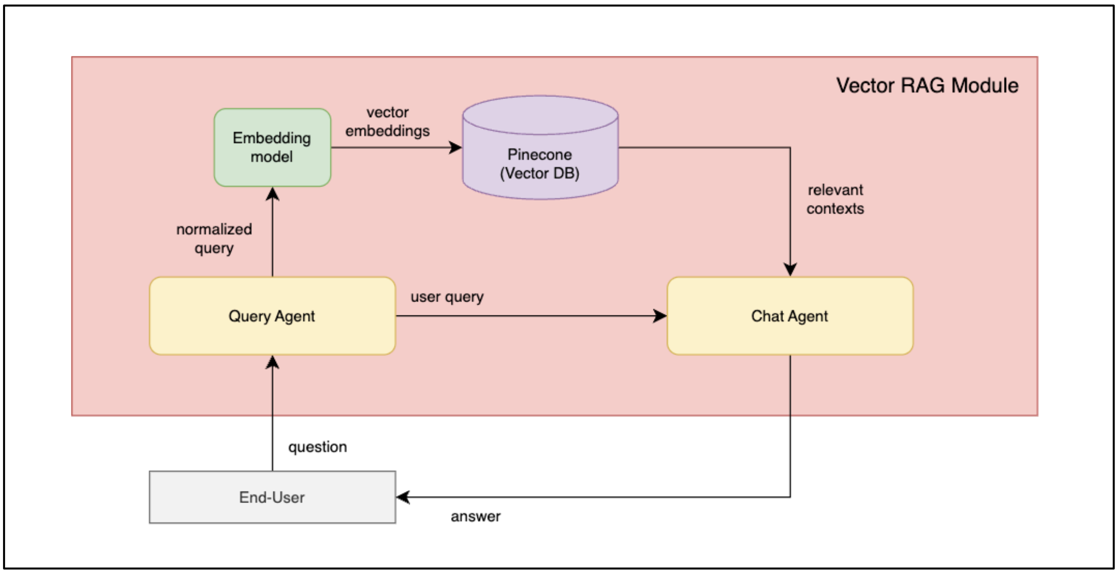


Figure 3.1.2.1 Vector RAG Module Block Diagram

The Vector RAG module is served as a chat-based Retrieval-Augmented Generation workflow to answer user questions by drawing on the vector database of company reports. In essence, this module allows a user to ask natural language questions about companies, and it will retrieve the most relevant information from the reports via the embeddings and generate a coherent answer. This approach extends a base large language model with the latest company-specific knowledge, avoiding the model's limitations, such as hallucinations and outdated training data by grounding answers in an external knowledge base.

In summary, the Vector RAG Module acts as a smart Q&A agent on top of the vectorized company report database. A user can query it in a chat interface, and behind the scenes the module will perform semantic search over the embedded knowledge base and then utilize an LLM to generate an answer that integrates the retrieved information. This design makes the system behave like a "virtual analyst," capable of synthesizing insights from across many documents and delivering answers to complex questions about a company's fundamentals.

3.1.3 Integration with Inter-Company Relationship Database

An important aspect of this project is how the above two components integrate with the Inter-Company Relationship Database Module (a separate but related subsystem). The Company Report Database's processed data provides a rich source of facts and disclosures that can be used to construct an ontology and knowledge graph of relationships between companies. For example, by scanning the reports' content, the system can identify mentions of partnerships, supplier-customer relationships, joint ventures, or other inter-company links. The relationship database module uses this information to build a network graph where companies are nodes and their significant business relationships form the edges. This means that from the raw textual data, such as "Company A's annual report mentions Company B as a major supplier", the system can formally represent that Company A → Company B has a supplier relationship in the graph.

The integrated system leverages both the structured relationship knowledge and the unstructured textual knowledge via RAG. The relationship mapping module can query the Company Report Database Module to get relevant data for populating the graph. Conversely, when answering a complex user query that involves multiple companies or dependencies, the relationship module's graph-based agent can call upon the Vector RAG Module to enrich the answer. For example, suppose a user asks, "How might Company A's recent earnings affect its major suppliers?" The relationship database can identify Company A's major suppliers from the ontology, and then for each supplier, use the vector database to retrieve contextual statements about the business relationship or any guidance given. The Chat Agent can then generate a nuanced answer that not only lists the related companies but also explains the inter-company impact with evidence from the reports, such as referring to Company A's outlook statements or the supplier's dependency noted in their own financial reports. This synergy between the graph and RAG components leads to more comprehensive and insightful answers than either approach alone.

3.2 Use Case Diagram

3.2.1 Module 1: Company Report Database Module

This module is responsible for ingesting, processing and storing company disclosure documents (annual report, quarterly reports, prospectuses). It provides a foundational repository of structured company information that other parts of the system can utilize. An Admin user oversees this module, primarily by triggering the retrieval and update of report data.

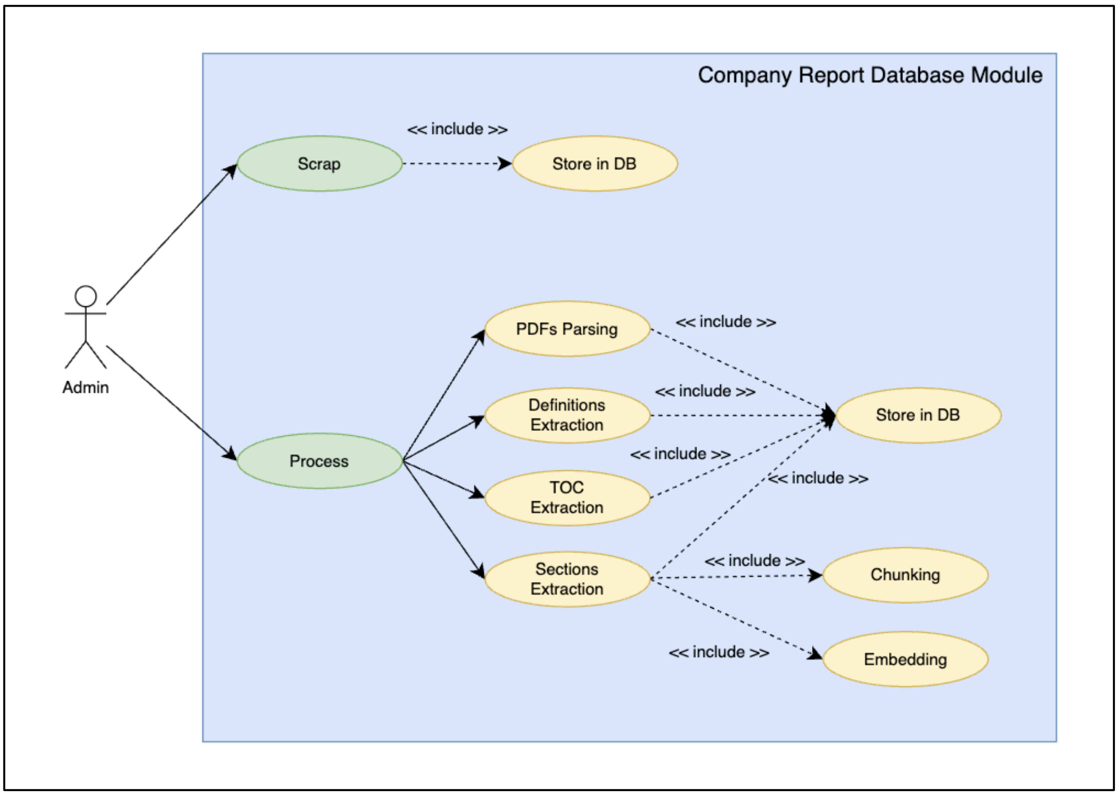


Figure 3.2.1.1 Company Report Database Module Use Case Diagram

The Admin initiates the process of scraping and processing company reports. The use case shown in Figure 3.2.1.1 represents the functionality for collecting new financial reports and transforming them into structured data.

The Admin triggers the system to fetch new company reports and incorporate them into the database. This includes scraping all the financial reports from Bursa Malaysia' website and then processing these documents using LLM techniques. The system extracts key information such as report sections and definitions and parsed content, ensuring the data is structured for analysis. Finally, it stores the processed results in persistent storage, including a document database and a vector database for downstream use.

3.2.2 Module 2: Vector RAG Module

The Vector RAG Module provides the user question-answering capability. It uses the processed data from the report database to answer users' questions. In this module, an end user asks questions in natural language, and the system employs vector semantic search and LLM-based generation to retrieve information and produce a helpful answer. This design allows the system to act as a "virtual analyst" that finds pertinent facts and delivers coherent responses to complex queries, effectively handling both straightforward data lookups and open-ended analytical questions.

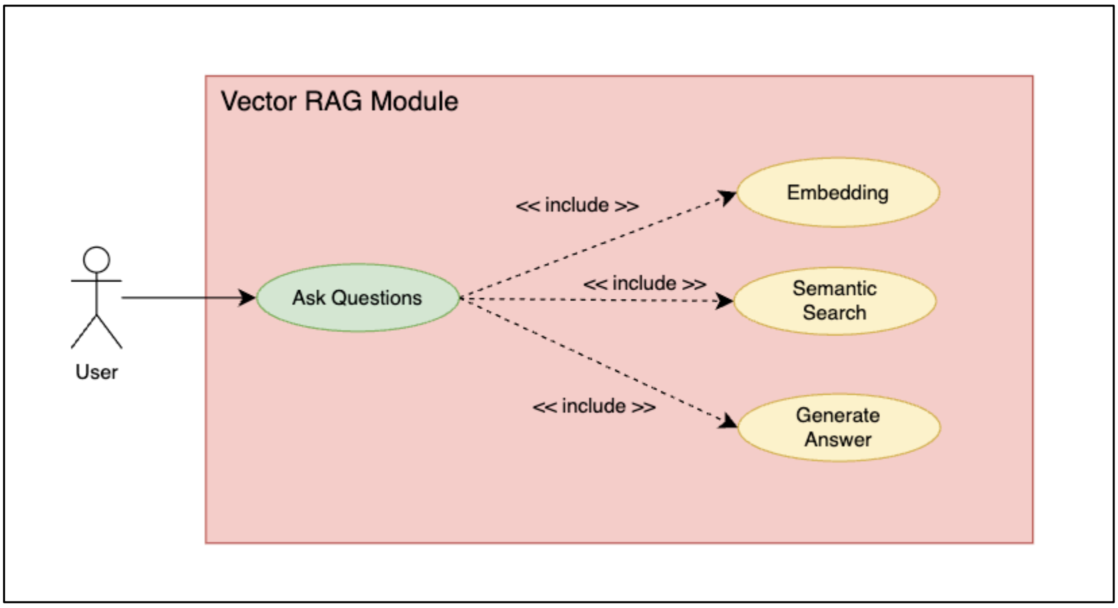


Figure 3.2.2.1 Vector RAG Module Use Case Diagram

The End-User interacts with this module by posing a question, such as, asking about a company's financials or management commentary. The use case represents the system's functionality to process the query and return an answer. The diagram in Figure 3.2.2.1 shows that the End-User initiates the Ask Questions use case, which the module handles internally by retrieving relevant knowledge through semantic searching and formulating a final response.

The End-User poses a natural-language question to the system about a company's fundamentals and receives a final answer. This covers the retrieval of relevant information from the vector report database and the generation of a response by the system's language model. The. Module utilizes retrieval-augmented generation where it finds semantically relevant pieces of stored text using vector embeddings, then a chat agent with LLM synthesizes those pieces into a coherent answer for the user. This allows the system to effectively address both direct factual queries and more open-ended analytical questions about the company.

3.3 Activity Diagram

3.3.1 Module 1: Company Report Database Module

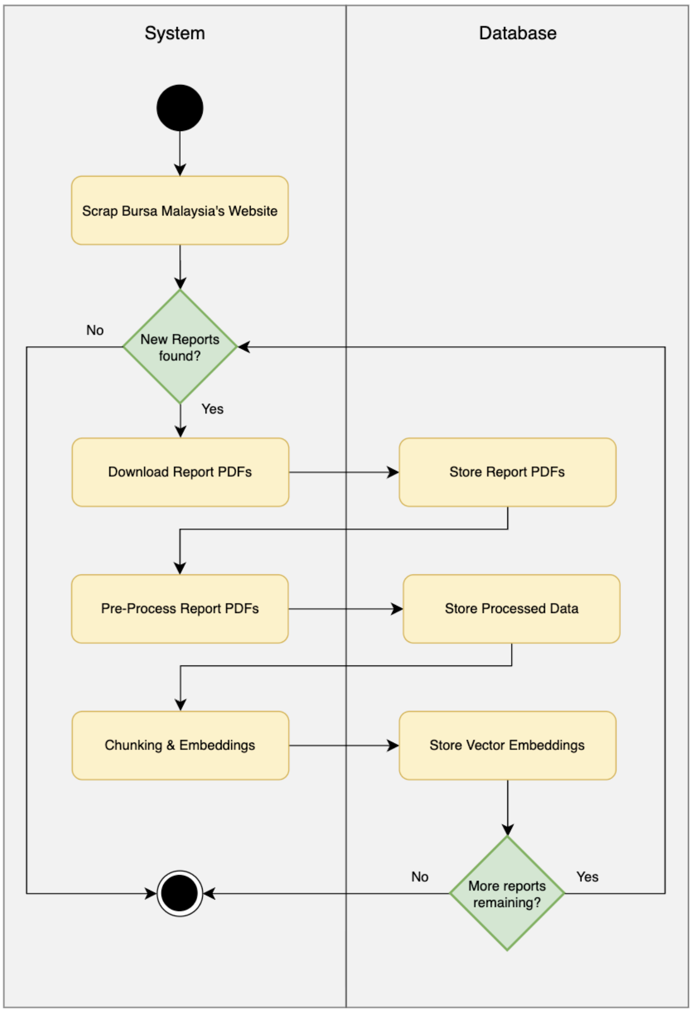


Figure 3.3.1.1 Company Report Database Module Activity Diagram

Figure 3.3.1.1 illustrates the step-by-step workflow of Company Report Database module from start to finish. The process begins when the Admin initiates the scraping operation. The system checks Bursa Malaysia for any new financial report documents. If new reports are found, it enters a loop to download each PDF, extract and parse its contents using LLMs, validate the extracted data, perform chunking and embedding, and then store the results in both the document database and the vector database. After each report is processed, the system checks if more reports remain to be handled. Once all new documents have been ingested, the flow terminates at the end node.

3.3.2 Module 2: Vector RAG Module

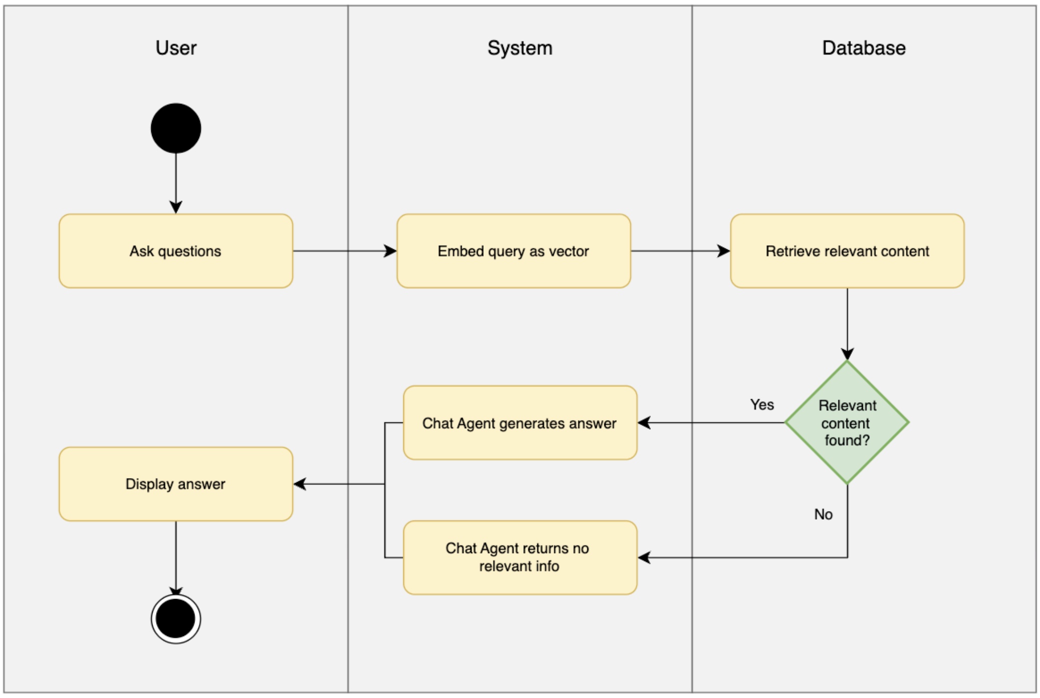


Figure 3.3.2.1 Vector RAG Module Activity Diagram

Figure 3.3.2.1 shows the flow of control when an End-User asks a question, and the system returns an answer. The process starts with the user's query. The question is first embedded into a vector representation, which is then used to query the vector database for relevant information. The decision node checks whether any relevant content was found in the knowledge base. If yes, the system invokes the LLM to generate an answer using those relevant report snippets as context. If no relevant content is available, the system hand this by preparing a response indicating it has no information. In either case, the final step is returning an answer to the End-User. The flow terminates. This activity diagram highlights how the module integrates RAG which involves both retrieval (vector similarity search) and generation (LLM-based answer formulation) to satisfy the user's query.

CHAPTER 4 SYSTEM DESIGN

4.1 System Block Diagram

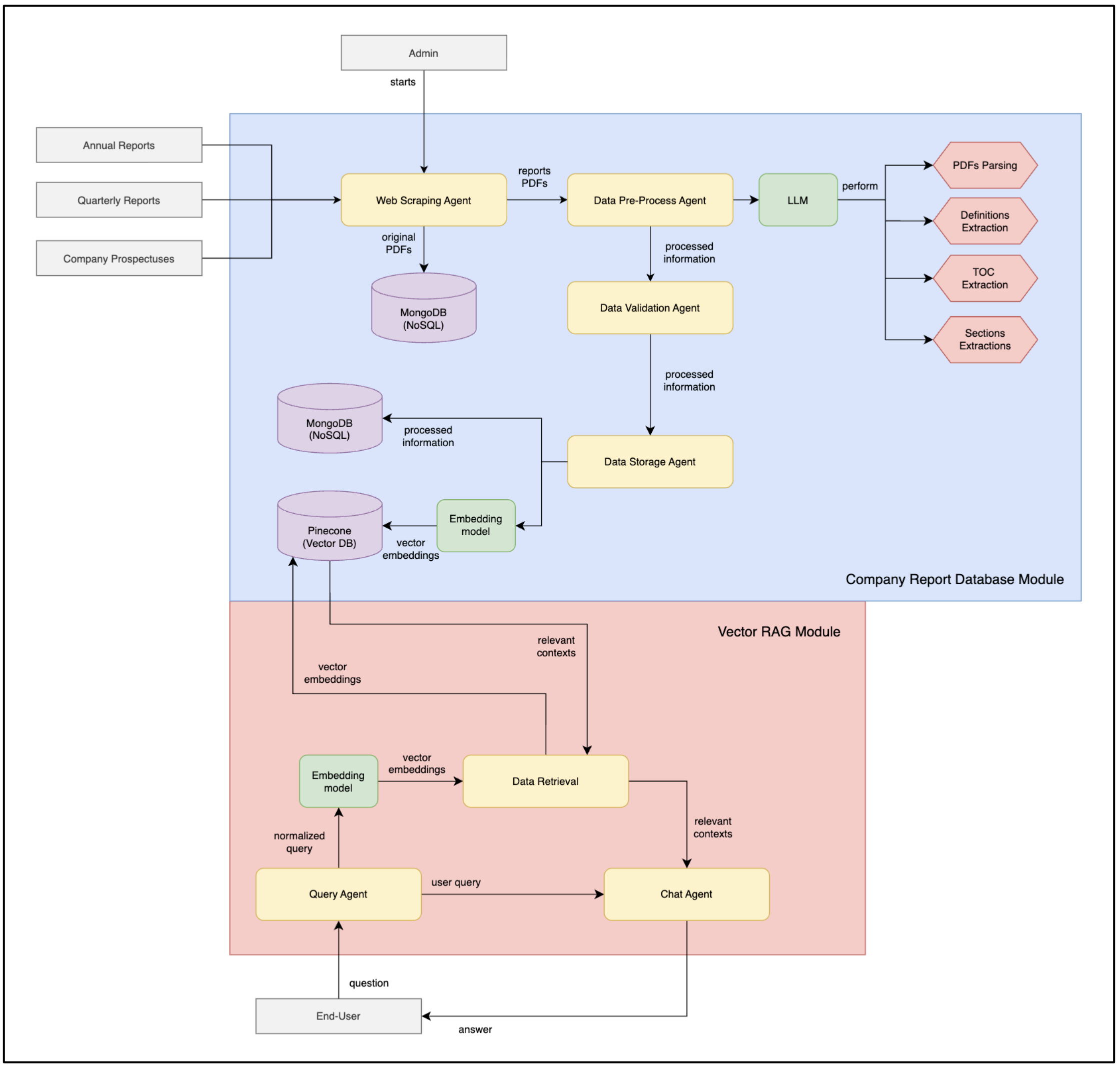


Figure 4.1.1 System Block Diagram

4.1.1 Web Scraping Agent

This report scraper component is the data ingestion point for the whole system. Its function is to retrieve financial reports (annual reports, company prospectuses) from Bursa Malaysia's official website. This agent periodically checks the target website for new financial documents and it can also be triggered when a user requests a report that is not yet in the database. It uses different web scraping tools to navigate through the site and download the report files (typically in PDF format). Besides that, this agent not only fetches the PDF documents, but it also collect additional data such as the report title, company name and code, financial period and the publication date. This metadata helps in organizing and referencing the documents later.

The agent automatically crawls the specified sources to retrieve the latest financial documents. It can handle both HTML webpages and PDF files. For HTML content, the agent employs web crawling and parsing techniques to extract relevant text and data fields. For PDF reports, it downloads the file and prepares them for extraction in subsequent steps. By automating the retrieval of these documents, the Web Scraping Agent ensures a comprehensive and timely data collection with minimal manual effort. The output of this stage is a collection of raw report data, which will be passed along to the next component for processing.

4.1.2 Data Pre-Process Agent

The Data Pre-Process Agent is responsible for parsing incoming company reports and extracting structured information from them. It uses LLMs to perform multiple sub-tasks in parallel, thus speeding up the processing of large documents. The agent executes three LLM-driven extraction tasks and then persists every result per section to MongoDB for reuse by other modules. These extraction tasks include prospectus definitions extraction for ontology, table of contents extraction and parallel section extraction from TOC.

Many financial reports especially prospectuses contain a "Definitions" or glossary section defining key terms and abbreviations. The agent uses an LLM prompt to identify if such a section exists and to extract the definitions of terms. This is done by instructing the LLM to scan the text for a definitions list and output a structured collection of term-definition pairs. By leveraging an LLM's understanding of language, the agent can accurately pull out the glossary of terms which can be important for interpreting the rest of the report.

Next, using the document's internal TOC page, the agent determines the major section headings of the report. It prompts an LLM with the full report text to output a list of the report's sections and sub-sections in order. The LLM is instructed via a specially crafter prompt, like "Extract the table of contents" and returns the section titles, which the agent parses as a JSON list of headings. This TOC extraction step yields an ordered outline of the report's structure, which is critical for guiding the next phase of extraction.

With the TOC in hand, the agent then extracts the content of each section one by one, often using LLMs. For each section title form the TOC, the agent can prompt the LLM to retrieve the text of that section from the report. Essentially, the LLM acts as a reader that, given the section name and the full report text, will output the exact content of that section. This divides a 100+ page report into logical chunks, each labelled by section. By performing these LLM extraction calls asynchronously, the agent can handle multiple sections in parallel, significantly accelerating processing. The result is a collection of section texts categorized by section name. Each section's content is now available as a clean text that can further analysed.

Traditional PDF pipelines frequently lose semantic content carried by visuals like charts, infographics and figures that management uses to communicate trends. By contrast, the LLM is able to interprets visual structure and language jointly, generates searchable captions and structured descriptors for each figure. This boosts retrieval recall for queries that would otherwise miss non-textual evidence. By the end of preprocessing, the originally unstructured report is converted into a structured dataset. It contains cleaned and standardized financial data as well as condensed textual insights.

4.1.3 Data Validation Agent

After preprocessing the reports, the Data Validation Agent checks for the integrity and quality of the processed data. Its main role is to validate and clean the data before it is stored in database. This agent verifies that all essential information from the original financial documents is present and correctly captured in the processed data. It checks for any missing key values and also confirms that the data are in the correct formats. Not only that, it makes sure that there are no duplicates or inconsistencies introduced during the scraping or preprocessing stages. If any errors are detected such as incomplete data sections, misparsed values or contradictory information, the agent will flag them for further review or attempt to correct them. For instance, it may attempt to re-process a particular missing section of a report. Therefore, the output of the Data Validation Agent is a clean, validated dataset of company reports and summarized texts.

This helps to ensure only reliable and high quality data proceeds to the storage stage, which is important for maintaining trustworthiness of the system's outputs.

4.1.4 Data Storage Agent

The Data Storage Agent is responsible for persistently storing both the original documents, processed data and the vector embeddings of the textual data. The system uses a dual storage approach to handle different data types optimally, a NoSQL document database for flexible storage of documents and metadata, and a vector database for efficient similarity search on text embeddings.

NoSQL Document Database is used to store all the original financial documents and the processed company's data. This includes the company name and code, report metadata, the extracted structured financial metrics and the textual content of each section. A document store is chosen for its flexibility and scalability in handling semi-structured data. For instance, financial reports can greatly vary in sections and length, which a NoSQL database can accommodate without a fixed schema. Each report is stored as a document, making it easy to retrieve the full report details or update them.

In parallel, the qualitative text extracted from each report is converted into a high-dimensional numeric vector using an embedding model. An embedding is a dense vector representation of the text such that semantically similar texts have similar vector values. These vectors are stored in a specialized vector database that supports fast nearest-neighbour searches.

Before any text is embedded, the system splits documents into chunks, because embedding whole report or a very long sections dilutes semantic signal and harms retrieval quality. A chunk is a contiguous span of text that is small enough for the embedding model to represent coherently, yet large enough to preserve local context. In practice, we first segment by logical section, then sub-segment each section into a fixed-length windows of 1200 characters with a sliding overlap of 150 characters.

When the chunk size is too large, the vector blends multiple topics together, causing its similarity scores become noisy. Whereas, if the chunk size is too small, the sentences will lose its context, leading to false positives and brittle answers. Therefore, overlapping preserves cross-boundary meaning and improves recall during retrieval.

Each chunk is cleaned, normalised and for tables, accompanied by a caption, header-aware textual linearization so its quantitative context is not lost. The system converts every chunk into a dense embedding vector using OpenAI's embedding model and stores it in the vector database with rich metadata. These metadata enables filtered search and precise source attribution.

Using both database in series forms a robust knowledge base where this approach is similar to an integrated vector database concept where embeddings are stored alongside original data for consistency, but here it is achieved by coordination between two systems. The NoSQL store and vector store are kept in sync through unique IDs, ensuring that a query result from the vector DB can always fetch the corresponding document from the document DB. The end result is a comprehensive, easily queryable repository of financial reports, where one that can be queried by specific fields, and by semantic content.

4.1.5 Query Agent

The Query Agent is the entry point for user questions. Its job is to understand the user's query and prepare it for effective retrieval. This involves potentially decomposing complex queries, extracting key parameters, and formulating a normalized query representation.

Users may ask questions in natural language, possible referencing companies and years in various ways. The Query Agent uses an LLM to parse and interpret these questions. Specifically, it can employ an LLM prompt designed for query analysis to identify any company names mentioned in the question. The system's design anticipates a future where the user might not explicitly provide structured inputs like a separate field for company name, so the Query Agent must infer those from the query text.

Using the LLM prompting, the agent detects which company the question is about. The prompt to the LLM could be something like "Extract the company names and years from this questions." The LLM's response can then be parsed to get a list of relevant company names. This is crucial for filtering the search later. If multiple companies are mentioned, the agent knows the query might require retrieving information for each and comparing.

At the same time, if the query is complex or multi-part, the Query Agent may break it down. For example, “How do the 2022 financial results of Company X compare to 2021?” might be split into two sub-queries. One to retrieve information about 2021 results and another for 2022, with a plan to compare them. In the architecture, this decomposition can be handled by sequential retrievals or by formulating a single query that covers both aspects.

Other than that, the Query Agent also normalizes the query into a form suitable for the retrieval steps. This could mean simplifying the language or ensuring the terminology matchers what is in the database. For example, if the user uses an informal term, the agent might rewrite it in terms of report language, using an LLM to paraphrase the question more formally and explicitly. It might also append or restructure the query with the identified parameters. This normalized query ensures that the subsequent embedding step focuses on the right aspects, like the core question text, possibly augmented with detected company metadata.

In summary, the Query Agent serves as the “brain” that interpret user intent. It heavily leverages LLM capabilities to parse free-form text and can use prompt-based logic to extract entities like company names. By the time the Query Agent has done its work, the system has a well-defined query or set of sub-queries along with the target company metadata, ready to feed into the retrieval system.

4.1.6 Data Retrieval Agent

Once the query is understood and refined, the next step is to convert that query into a form that can be used to search the semantic index. This is where the embedding process comes in. The core idea is to generate a dense vector representation of the user’s question, using the same embedding space that the documents were stored in, so that to find semantically similar contents.

The normalized query text is passed through a text embedding model to produce a numerical vector. The system then uses this vector for similarity search. It is important that the same embedding technique was applied to the documents, since the system stored report sections as vectors using the identical model. The user’s question is now in a vector form that captures its essence, which then be used to probe the vector database for relevant information.

The Data Retrieval Agent is responsible for searching the knowledge base and fetching the pieces of data that can answer the query. Using the query vector from previous step, it performs a similarity search in the vector database to retrieve the most relevant chunks of text from company reports. The query vector is sent to vector database with a request for the top K nearest neighbour vectors. Vector database which indexes all the embedded report sections, computes the similarity (via consine similarity) between the query vector and every stored vector, efficiently returning the closest matches. Each match returned includes the stored vector's metadata and the original text chunk. This retrieval's speed and optimization for this task allow these nearest-neighbour lookups to be very fast, even with thousands of embedded sections.

Not only that, the retrieval can be scoped using metadata filters if the Query Agent identified any specific company in the query. The search can be restricted to only the company's data. This prevents irrelevant results from other companies from showing up. So the Data Retrieval Agent might search with a filter like "{"company": "ACME CORP"}", if the query is specifically about Acme Corp's report. By applying these filters, the system increase precision, where it will not retrieve a chunk that is totally irrelevant to the question.

The most semantically relevant chunks of text were returned after each search, where each usually a paragraph or section excerpt that potentially contains the answer. Each chunk comes with its source metadata and also store a snippet of text in metadata.

The Data Retrieval Agent thus bridges the gap between the user's question and the knowledge base content. After this step, the system has a collection of relevant report chunks that likely contain the answer to the question, plus context around it. These chunks are now handed off to the Chat Agent for final answer synthesis. It is important to note that by retrieving actual text segments, it maintains a grounding for the answer. The next stage will use these exact segments to generate the answer.

4.1.7 Chat Agent

The Chat Agent is the component that generates the final answer to the user question. It takes the retrieved chunks of report text and the original query, and uses an LLM to compose a coherent answer strictly grounded in the provided content.

The Chat Agent builds a prompt for the LLM that includes the context and clear instructions. After listing the relevant context, the prompt then appends the user's actual question at the end. The prompt also includes a system message instructing the LLM on how to answer. This instruction would tell the model to use only the given context to answer the question and not to rely on any outside knowledge. It also enforces strict grounding rules through the prompt. Specifically, it multiple companies are in context, it ensures the LLM distinguishes them and does not mix information from different companies. The context chunks themselves are labelled by company, and the question usually makes it clear which company's information is needed, so the prompt will remind the model to keep data separated by company. To prevent hallucination, the instruction explicitly forbids the LLM from introducing any information that is not present in the context.

Once the prompt is assembled, the Chat Agent invokes the LLM to generate the answer. Because the prompt includes the relevant report text, the LLM can draw the answer from those chunks. The use of a state-of-the-art LLM ensures the answer is fluent and well-structured, while the grounding ensures accuracy. The result delivered by the Chat Agent is a well-formed answer to the user's query, with each factual claim grounded in the retrieved source material.

Through the Chat Agent's careful orchestration of the LLM, the system adheres to a retrieval-augmented generation paradigm, where the generation is augmented and constrained by real data from the reports. This design means the answer will remain accurate even when the LLM's own training knowledge is outdated or if the question is very specific, because the model is not generating from scratch but rather synthesizing given facts.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 Hardware Setup

The hardware involved in this project is only a computer. This computer is responsible for handling full range of system operations, including web scraping, data processing, data storing. No dedicated GPU is required since heavy LLM computations such as embeddings and contents generation are offloaded to cloud APIs through OpenAI and Google. Server with network access to MongoDB Atlas and Pinecone cloud services were configured to ensure scalability and low-latency vector search. Here are the specifications and usage details for the computer.

Description	Specifications
Model	Apple Mac Mini
Processor	M4
Operating System	MacOS Sequoia
Memory	16GB RAM
Storage	256GB SSD

Table 5.1.1 Specifications of Computer

5.2 Software Setup

Category	Component	Description
Development	Prototyping	Python
	Implementation	Python
	Libraries	Requests, BeautifulSoup4, OpenAI, Pinecone, Google-GenerativeAI
	IDE	Visual Studio Code
Data Management	Data Storage	MongoDB, Pinecone
	Data Transfer	JSON
Core Technology	LLMs	OpenAI GPT, Google Gemini
Communication	Protocol	HTTP/HTTPS, RESTful APIs
Version Control	Platform	GitHub

Table 5.2.1 Overview of Software Tools

5.3 Design Specifications

The development of the Company Report Database Module follows a modular and iterative approach. The system is designed as a pipeline consisting of different components, each responsible for a specific task in the data flow from raw input to final output. The general workflow starts with data collection, then data processing, followed by storage, and finally information retrieval. This step-by-step approach ensures that each stage is functional and validated before being integrated into the larger system.

5.3.1 Tools and Technologies Used

To implement the above methodology, the project leverages a range of tools and technologies, where each is chosen because of its suitability to meet the specific requirements of the system.

- **Web Scraping**
  The system attempts to download financial documents and PDF links with the lightweight Python request library, which issues direct HTTP GET/HEAD calls and follows redirects [12]. This requests library is faster and consumes far fewer resources than a full browser session, making it the preferred path for the large majority pages that return static HTML. Only when a target page employs a CAPTCHA or anti-bot mechanisms, alternative solutions such as Selenium or Playwright become useful. These uses browser automation framework to navigate and interact with web pages when scraping data. Selenium is a widely used tool that automates web browsers [13]. It was originally developed for automated web testing but is equally useful for web scraping dynamic sites. Playwright which is a newer automation library, is designed for fast, reliable web automation across different browsers [14]. Both Selenium and Playwright allow web scraper to mimic real user behavior, which is essential for accessing websites with dynamic content loading or anti-scraping measures. By using these automation tools, the Web Scraping Agent can handle complex, JavaScript-driven content on Bursa Malaysia website.
- **HTML Parsing**
  After fetching HTML pages, the project employs BeautifulSoup (a Python library) to parse and extract data from the raw HTML content [15].

BeautifulSoup provides a convenient way to navigate the HTML DOM, find specific tags, and retrieve text. This allows the agent to find tables of company announcements by tag/attribute patterns and traverse through all the anchor tags inside those tables to capture the PDF download URLs for the full reports. The library's tolerant parser handles imperfect or nested HTML, and its CSS-selector-like queries make the workflow concise and maintainable. Therefore, BeautifulSoup allows system to systematically pulls out the needed pieces of information from each page.

- **NoSQL Database**

  All the financial reports gathered and processed data is stored in MongoDB database. MongoDB is a document-oriented NoSQL database that stores information in flexible, JSON-like documents rather than rigid tables [16]. This schemeless design fits our needs perfectly, since different reports may have different sets of fields or sections. MongoDB ensures scalability and allows for easy addition of new data points without changing the fixed schema. By using MongoDB, the module benefits from high-performance CRUD operations on large numbers of documents and can be easily scaled horizontally if the data volume grows.

- **Vector Database**

  In addition to MongoDB, the system uses Pinecone as a vector database to store and search high-dimensional embeddings of textual data. Pinecone is a cloud database optimized for vector similarity search, commonly used in AI applications [17]. It enables the system to perform semantic queries, instead of doing keyword matching, the query is converted into a vector and compared with the stored vectors to find semantically similar content. Pinecone is known for its speed and scalability for processing large vectors, as well as millisecond query latencies, making it an ideal choice for our use case of searching large amounts of document text. In our design, textual portions of reports are embedded into vector form using a pre-trained language model for embeddings and stored in Pinecone. By leveraging Pinecone, the module enables intelligent search capabilities beyond exact matches, which is critical for gaining qualitative insights from reports.

- **Large Language Models (LLMs)**

The system integrates LLMs to enhance its understanding and generation of natural language content. In this project, LLMs (such as OpenAI's GPT-5 mini, Google's Gemini 2.5 Pro) is utilized in two ways. First, during data processing, it helps to process and extract insights from lengthy reports. This helps transform unstructured text into more structured insights that can be stored or presented directly. Second, at query time, LLM helps interpret complex user questions and formulate responses. The system can use LLMs to understand the intent of the question and compose an answer based on the retrieved data. The use of LLMs elevates the system from a basic data retrieval service to an intelligent assistant capable of understanding nuance in questions and answers.

- **Embedding & Chunking**

  Before embedding, reports are segmented into semantically coherent chunks to preserve meaning while fitting model limits. The agent first uses the extract TOC to split by logical sections, then sub-segments long sections into ~1200 characters with 150 characters overlap. Overlap protects context across boundaries. For instance, excessively large chunks will blur out topics, while tiny chunks lose context. Cleaned chunks are converted into dense vectors using an OpenAI embedding model (text-embedding-3-small). The same model is used later for query embeddings to ensure vector space alignment.

5.4 System Operation

5.4.1 Web Scraping Agent

Before starting to develop web scraping agent, the structure of the "Company Announcement" page of Bursa Malaysia's website was explored and studied.

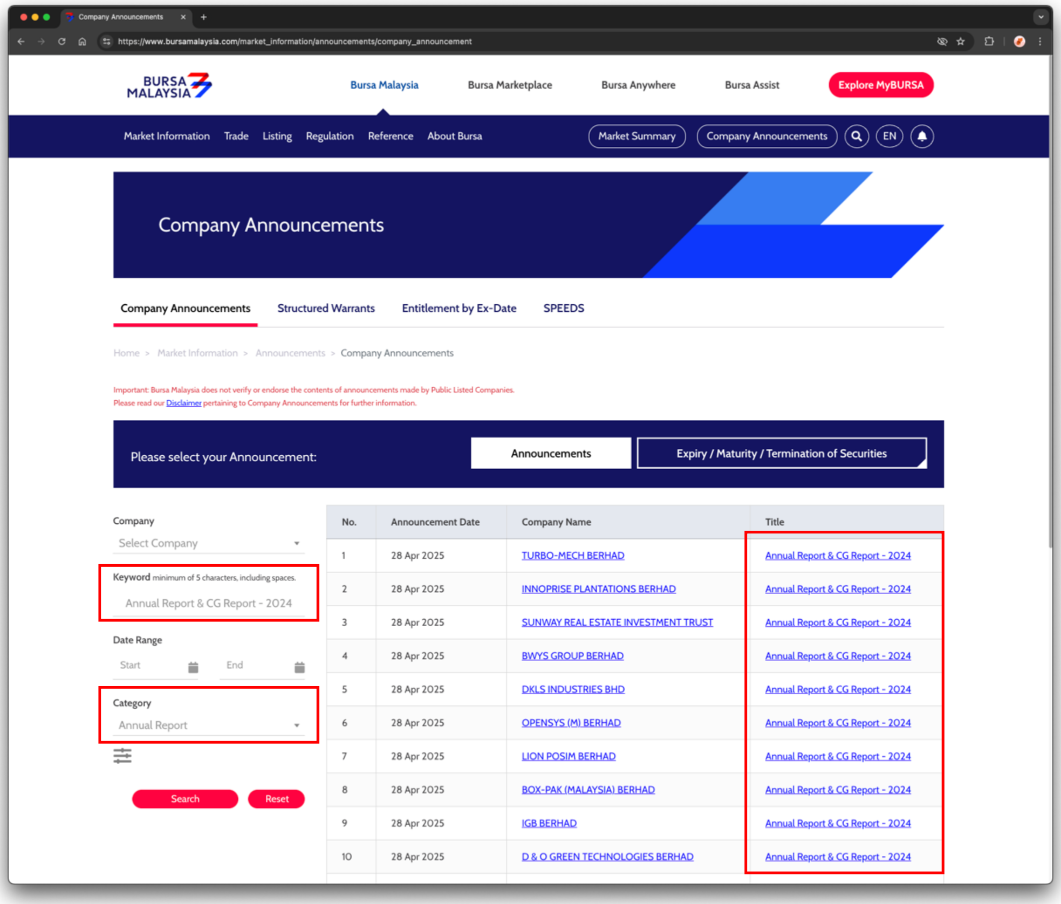


Figure 5.4.1.1 Company Announcement Page

First of all, Figure 5.4.1.1 shows the "Annual Report" category is selected in the filtering section to show only company's' annual reports. It also can be searched by entering the keyword, such as "Annual Report & CG Report – 2024" to filter out the annual report of the year 2024. The page will then loads all the results found in a tabular form. It contains all the corresponding URLs to the announcement page for each result found. These announcement URLs will bring us to the page that shows all the details about the specific announcement as shown in Figure 5.4.1.2 below.

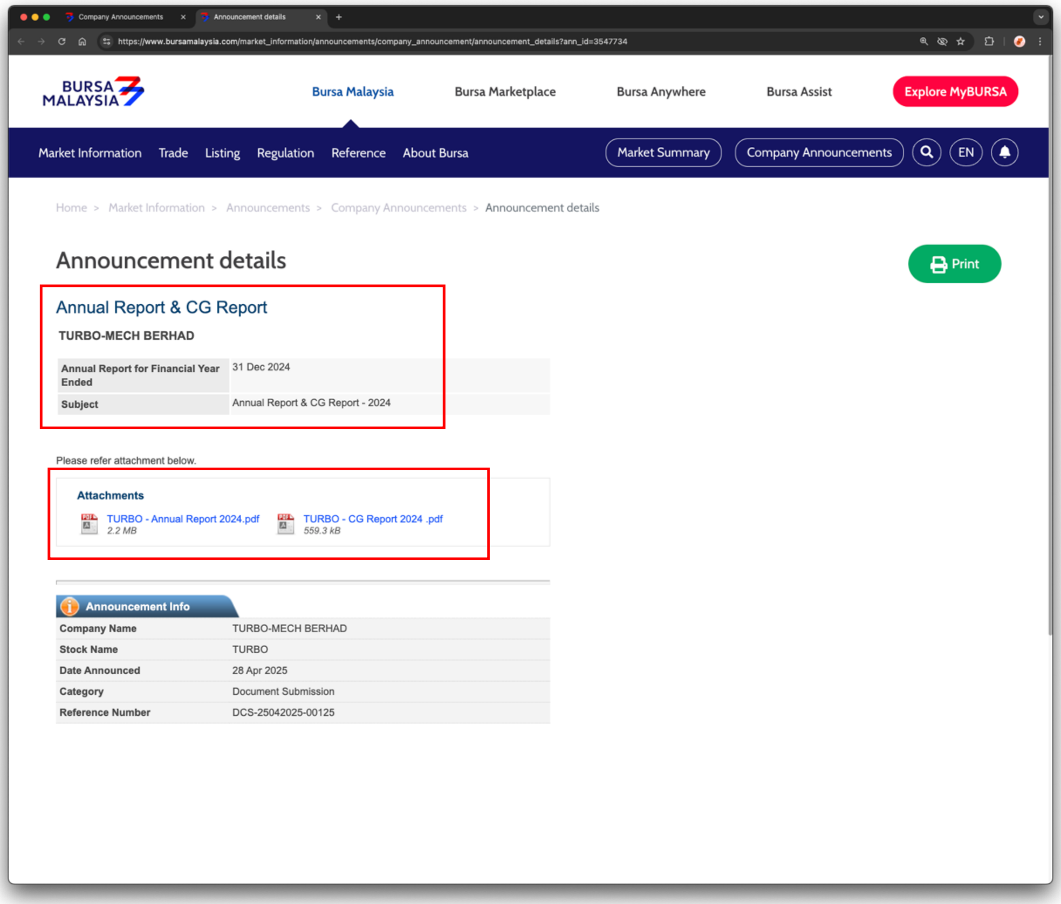


Figure 5.4.1.2 Announcement Details Page

This page will show all the details about the announcement including the subject title, company name and the date of announcement. It also contains an attachments section that contains all the URLs to access the financial documents attached. These attachment URLs are the main object the web scraping agent is looking for. The agent will then access and download all the financial documents using these URLs. These steps are repeated to scrape for all financial documents from Bursa Malaysia's official website.

Two approaches were prototyped for web scraping, one using direct HTTP requests in Python, and another using browser automation tools (Selenium and Playwright). In the direct HTTP requests method, Python *requests* and *BeautifulSoup* libraries were used together to fetch HTML content and parse links to PDF files.

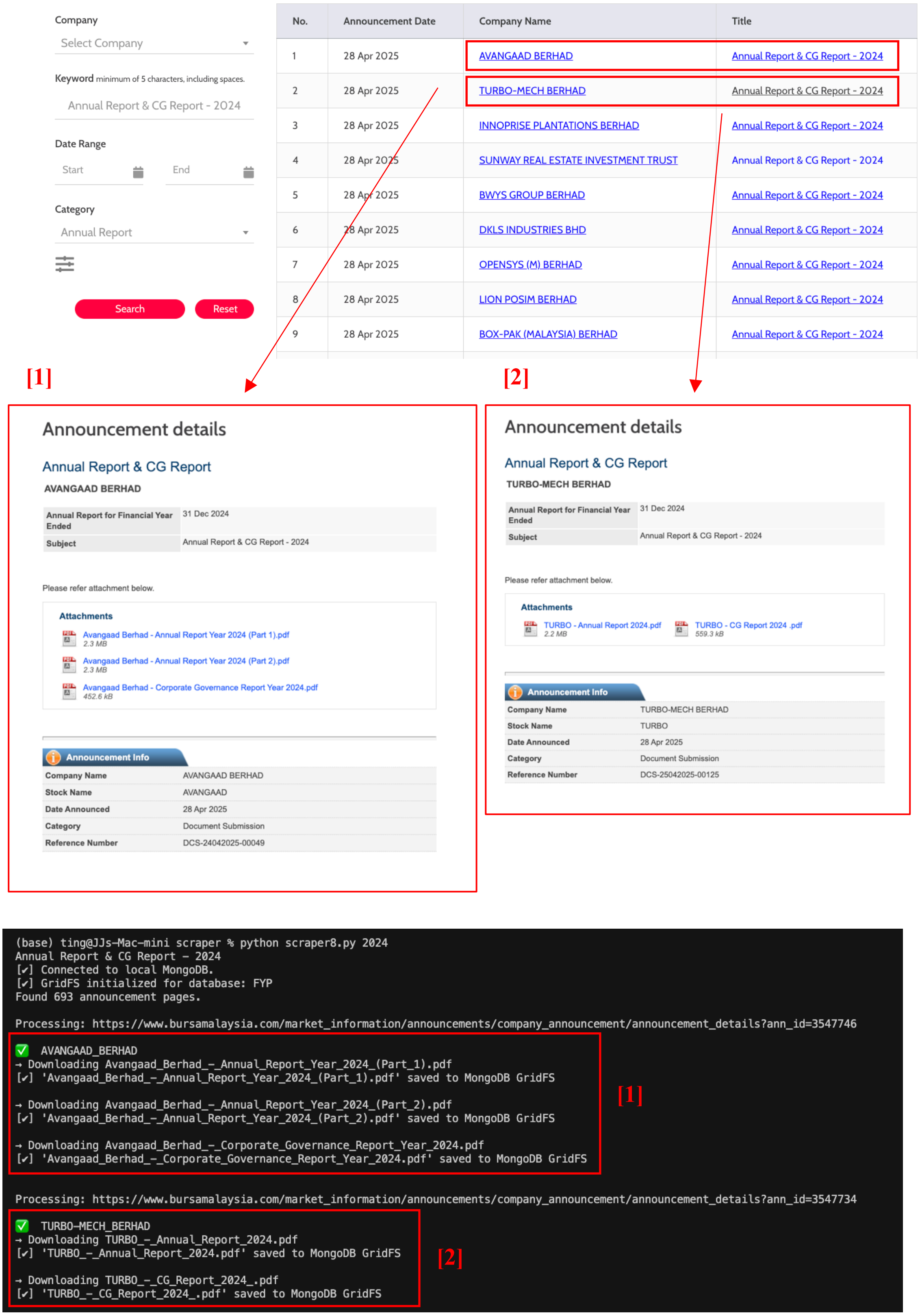


Figure 5.4.1.3 Web Scraping Agent

Figure 5.4.1.3 shows that the agent successfully scraped a set of annual report PDFs from Bursa Malaysia. The scraper query the Bursa Malaysia announcements page for a given year and visit each announcement's detail page to find the actual PDF download links. The PDF files for each company were downloaded and stored in a MongoDB database as shown in Figure 5.4.1.4 and Figure 5.4.1.5 below. A GridFS file storage was used to accommodate the binary PDF data, and a separate metadata collection recorded the company name, report year, and source URL for each file. This confirmed that the relevant reports can be automatically collected and persisted for further processing.

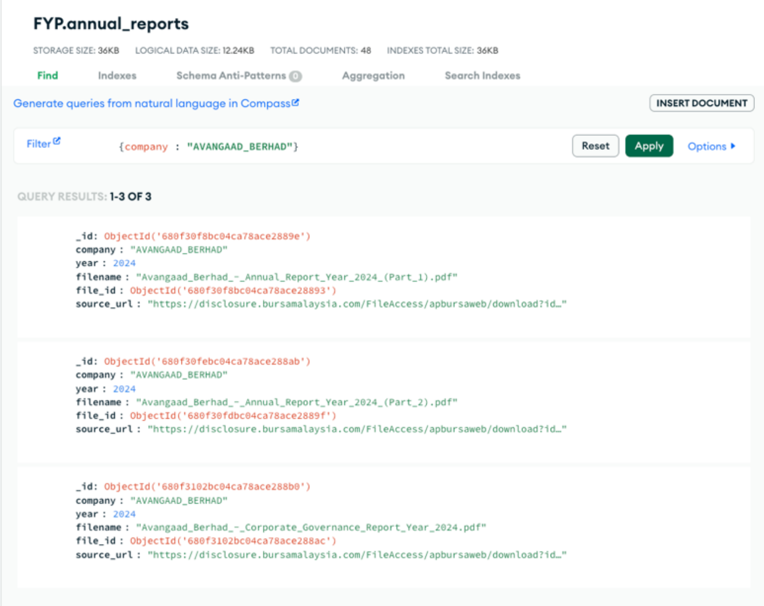


Figure 5.4.1.4 AVANGAAD BERHAD in MongoDB

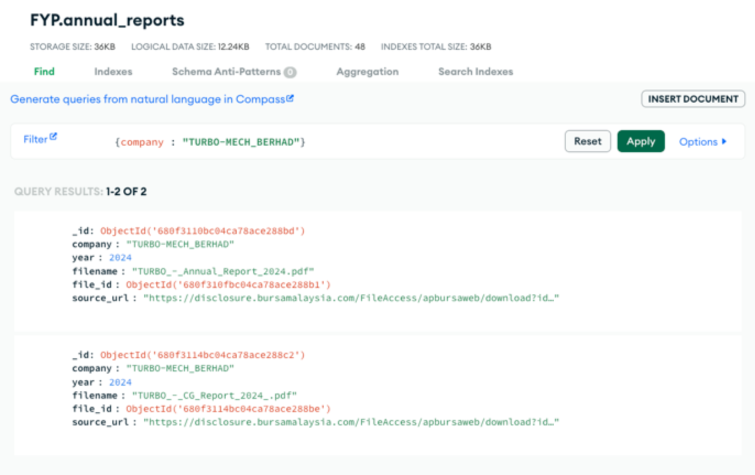


Figure 5.4.1.5 TURBO-MECH BERHAD in MongoDB

However, one observation during scraping was the potential risk of being blocked due to high-frequency access. Rapid and repeated requests to the Bursa website could result in HTTP errors or temporary bans if the site detected a bot. Going forward, rotating IP proxies or a more distributed scraping schedule may be considered to further reduce the chance of being blocked. Nevertheless, in the preliminary experiments the scraper was able to retrieve the required documents without encountering a permanent block.

To increase throughput while keeping the crawler responsive, the Web Scraping Agent runs the listing crawl, announcement-page parsing and PDF downloading concurrently in a three stage pipeline. Announcement listings, announcement detail pages and attachment (PDF) downloads are processed in parallel and decoupled with in-memory task queues. An asynchronous HTTP client handles I/O-bound requests with a shared keep-alive session, while a small thread pool performs CPU-bound HTML parsing and MongoDB GridFS writes.

5.4.2 Data Preprocess Agent

After obtaining the PDF reports, the next step was to preprocess these documents for analysis by language models. Preliminary testing was conducted with open-source PDF parsing techniques to extract text from the annual reports. Libraries such as PyMuPDF were explored to programmatically convert PDF content into plain text. However, a key challenge noted was that while these libraries can extract textual content, they do not capture the information in charts, tables, or images, which are abundant in annual reports. Simply extracting all text can also produce a very large string, often tens of thousands of words for a single report, which will increase the input tokens and cost of the analysis by LLMs.

Therefore, a direct PDF-to-LLM approach was tried and the results outperform the previous method. In this approach, raw PDF files were fed into a LLMs to let the model parse and extract the content. Specifically, we utilized Google's generative AI model (Gemini 2.5 Pro) via Google's File API that allowed file upload as context. The model was prompted to read the entire annual report PDF and extract key information to produce a structured data in markdown format. The model will perform three tasks, including extracting the definitions, TOC, and each TOC sections using the detailed prompt as shown in Figure 5.4.2.1, Figure 5.4.2.2 and Figure 5.4.2.3 respectively. The prompt explicitly directed the model to not omit any details and to interpret non-textual content like charts or infographics in words. Surprisingly, the token usage for a direct PDF-to-LLM approach is lesser than the traditional PDF pure text extraction method due to many repeated and unwanted sections were extracted as well.

```
PROMPT[
    "DEFINITION PARSING"
] = """
            You are an information extraction system. The provided PDF contains two relevant sections: “DEFINITION” and “GLOSSARY OF TECHNICAL TERMS.” Extract only the entries under these two sections, mapping each term to its definition as key-value pairs in JSON. Return only the JSON—no explanations, headers, or additional text.
            Output format example:
            {
            "Term": "Definition"
            }
        """
```

Figure 5.4.2.1 Detailed Prompt for Definitions Extraction

DEFINITIONS

The following definitions shall apply throughout this Prospectus unless the term is defined otherwise or the context otherwise requires:

Companies within our Group

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))

General

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 5.4.2.2 Example of Definitions in Company Prospectus

```
_id: ObjectId('68937e3398cecf4069228444')
type : "CONSTRAINTS"
name : "AUTOCOUNT_DOTCOM_BERHAD_IPO_WORD_DEFINITION"
from_company : "AUTOCOUNT_DOTCOM_BERHAD"
content : "{                                                                  "
            "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 (14
            "ADB Group or Group": "ADB and its subsidiaries, namely ACSB, ACSPL, AOTGSB ar
            "AOTGSB": "Autocount On The Go Sdn Bhd (Registration No.: 201601008185 (11791
            "ASSB": "Autocount Software Sdn Bhd (Registration No.: 202001018079 (1374399-\
            "ACE Market": "ACE Market of Bursa Securities",
            "Acquisition of ACSB": "Acquisition by ADB of the entire equity interest of AC
            "Acquisition of AOTGSB": "Acquisition by ACSB of the entire equity interest of
            "Acquisition of ASSB": "Acquisition by ADB of the entire equity interest of AS
            "Acquisitions": "Collectively, the Acquisition of ACSB, the Acquisition of AOT
            "Act": "Companies Act 2016",
            "ADA": "Authorised Depository Agent, a person appointed by Bursa Depository ur
            "ADB Shares or Shares": "Ordinary shares in ADB",
            "AMCL": "Autocount (Myanmar) Company Limited (Registration No.: 111547688)",
            "Application": "Application for our IPO Shares by way of Application Form, Ele
            "Application Form(s)": "Printed application form(s) for the application of our
            "ASEAN": "Association of Southeast Asian Nations",
            "ATM": "Automated teller machine",
            "Authorised Dealer(s)": "Any person or entity who has a valid dealer agreement
            "Authorised Financial Institution(s)": "Authorised financial institution(s) pa
            "Board": "Board of Directors of ADB",
            "Bursa Depository": "Bursa Malaysia Depository Sdn Bhd (Registration No.: 198
created_at : 2025-08-06T16:09:23.466+00:00
published_at : "14 Apr 2023"
```


Figure 5.4.2.3 Extracted Definitions Saved in MongoDB

```
PROMPT[
    "TABLE OF CONTENT EXTRACTION"
] = """
    Please extract only the top-level section headings listed under the Table of Contents of the cached PDF.

    WHAT TO CAPTURE
    - Top-level items = first-level sections in the TOC (not sub-sections).
    - Titles may appear next to page numbers; treat those numbers as page numbers, not section numbers.

    PAGE-NUMBER VS. SECTION-NUMBER RULES
    - If a line begins with a bare number (e.g., "02", "4", "156") followed by spaces/dot leaders and then text, that number is a PAGE NUMBER – ignore it.
    - If a line ends with a number after dot leaders (e.g., "Title .... 35"), that number is a PAGE NUMBER – ignore it.
    - If the text explicitly contains a section marker ("Section N", "Chapter N", "Part N", "N." before the title), you may use it to confirm the item is top-level, but
    **do not keep that original number** in output. We will renumber all items sequentially.
    - Numbers inside the title that are part of the wording (years, amounts, model names) must be preserved.

    NORMALIZE EACH CAPTURED ITEM TO THIS EXACT FORM
    - Output as: "N. Title"
    - N = sequential Arabic numeral starting at 1 based on top-to-bottom order in the TOC (1, 2, 3, …). Ignore any page numbers or original section numbers.
    - Title = the heading text as it appears (preserve casing and words).
    - Remove dot leaders and any page numbers.
    - Collapse internal whitespace to single spaces; trim leading/trailing spaces.
    - Ignore sub-sections like "1.1 …", lettered items ("A.", "B."), roman-numeral lists, bullet lists, or unnumbered minor headings.

    ORDER
    - Keep the original TOC order from top to bottom.

    OUTPUT
    - Return a JSON array of strings only (no extra keys, no commentary, no page numbers).
    - Example (input lines like "02 Global Presence", "04 Financial Highlights", ...):
    ["1. Global Presence", "2. Financial Highlights", "3. Corporate Structure", "4. Corporate Information"]
"""
```

Figure 5.4.2.4 Detailed Prompt for Table of Contents Extraction

TABLE OF CONTENTS

Figure 5.4.2.5 Example of Table of Contents in Company Reports


```
_id: ObjectId('6893913b98cecf406922889c')
from_company : "AUTOCOUNT_DOTCOM_BERHAD"
name : "AUTOCOUNT_DOTCOM_BERHAD_IPO_TOC"
type : "TOC  "
content : "[                                                                    "
           "1. CORPORATE DIRECTORY",
           "2. APPROVALS AND CONDITIONS",
           "3. PROSPECTUS SUMMARY",
           "4. DETAILS OF OUR IPO",
           "5. RISK FACTORS",
           "6. INFORMATION ON OUR GROUP",
           "7. BUSINESS OVERVIEW",
           "8. INDEPENDENT MARKET RESEARCH REPORT",
           "9. INFORMATION ON PROMOTERS, SUBSTANTIAL SHAREHOLDERS, DIRECTO
           "10. RELATED PARTY TRANSACTIONS",
           "11. CONFLICT OF INTEREST",
           "12. FINANCIAL INFORMATION",
           "13. ACCOUNTANTS' REPORT",
           "14. STATUTORY AND OTHER INFORMATION",
           "15. SUMMARISED PROCEDURES FOR APPLICATION AND ACCEPTANCE"
         ]
created_at : 2025-08-06T17:30:35.021+00:00
published_at : "14 Apr 2023"
```


Figure 5.4.2.6 Extracted Table of Contents Saved in MongoDB

```
PROMPT[
    "REPORTS PARSING SYSTEM INSTRUCTION"
] = """
    You are a PDF-to-text converter and interpreter. A financial report PDF has been loaded into context and cached.
    When I say:
    Section: "<Section Name>"
    you must extract only that section (matching the Table of Contents).

    - Extract and convert only the content under the given section heading, preserving 100% of the original meaning.
    - For any charts, diagrams, tables, or illustrations within that section, provide a comprehensive textual interpretation that fully and accurately conveys their
    content, including a reference to the section title and page number.
    - Exclude all content outside the specified section.
    - Remove any headers, footers, or page numbers.
    - Return only the plain text of that section—no explanations, metadata, headers, or additional commentary.
"""

PROMPT[
    "IPO SECTION PROMPT FRESH"
] = """
    You are extracting authoritative content from one or more PDF filings.

    SECTION: "{section}"

    OBJECTIVE
    Return STANDARDIZED MARKDOWN optimized for atomic chunking. Do not add or remove meaning.

    CORE RULES
    1) Fidelity: Preserve 100% of content; no summaries or commentary.
    2) Scope: Include only text truly under this heading/subheadings (ignore headers/footers/margins).
    3) Pagination: Add page refs as “(p. X)” where applicable.
    4) Headings: Use ATX Markdown.
      - First line: # {section}
      - Use ## for major subheads and ### for nested subheads you observe in the PDF. Include page refs in the heading when helpful: e.g., "## Risks (p. 14)".
    5) Paragraphs: Separate with one blank line. No hard wraps inside a paragraph.
    6) Lists:
      - Use "- " for bullets; "1." for numbered lists.
      - Preserve roman enumerations like "(i)", "(ii)" at the start of items.
      - One list item per line (no wrapping).
    7) Tables (ROW-AS-BULLETS for chunking):
      - First add a label line: "Table: <Exact Title or [no title]> (p. X)".
      - Then output each table row as a single bullet on one line:
      - "Col A: Val; Col B: Val; Col C: Val"
      - Do not include a markdown grid table. If the table spans pages, show both pages in the label (e.g., "(p. 121–122)").
    8) Figures/Diagrams:
      - Label: "Figure: <Title/description> (p. X)"
      - Follow with one paragraph describing the figure (no image).
    9) Numbers: Keep all numeric formats exactly (commas, decimals, signs, currencies).
    10) Output: Markdown only. Do NOT use code fences.

    OPTIONAL (use only if present in the source)
    - If the section has a short preface or bulletable outcomes, add:
    ## Key Points
    - <verbatim point or heading stub from the source>

    RETURN SKELETON (adapt to the actual content)
    # {section}

    ## Key Points
    - …

    ## <Subheading A> (p. X)
    <paragraphs>

    Table: <Title or [no title]> (p. X)
    - Col 1: …; Col 2: …; Col 3: …
    - Col 1: …; Col 2: …; Col 3: …

    Figure: <Title> (p. X)
    <one-paragraph description>

    ### <Nested Subheading> (p. X)
    - (i) …
    - (ii) …
"""
```

Figure 5.4.2.7 Detailed Prompt for Sections Extraction

```
# AUTOCOUNT_DOTCOM_BERHAD IPO

# 1. CORPORATE DIRECTORY

## BOARD OF DIRECTORS (p. 19)

Table: BOARD OF DIRECTORS (p. 19)
- Name: Choo Chin Peng; Designation: Executive Director / Chairman; Gender: Male; Nationality: Malaysian; Address: 23, Jalan Bayu Laut 8, D'Laman Greenville, 41200 Klang, Selangor
- Name: Choo Yan Tiee; Designation: Executive Director / Managing Director; Gender: Male; Nationality: Malaysian; Address: 26, Jalan Bayu Laut 12, D'Laman Greenville, 41200 Klang, Selangor
- Name: Dato' Ng Wan Peng; Designation: Independent Non-Executive Director; Gender: Female; Nationality: Malaysian; Address: 16-8, One Central Park, Jalan Residen 2, Desa Parkcity, 52200, Kuala Lumpur Wilayah Persekutuan
- Name: Dr. Liew Soung Yue; Designation: Independent Non-Executive Director; Gender: Male; Nationality: Malaysian; Address: 220, Jalan Mahsuri 6, Taman Mahsuri, 31900 Kampar, Perak
- Name: Chin Chee Seng; Designation: Independent Non-Executive Director; Gender: Male; Nationality: Malaysian; Address: No. 9, Jalan 8/149L, Bandar Baru Seri Petaling, 57000 Kuala Lumpur Wilayah Persekutuan

## AUDIT AND RISK MANAGEMENT COMMITTEE (p. 19)

Table: AUDIT AND RISK MANAGEMENT COMMITTEE (p. 19)
- Name: Chin Chee Seng; Designation: Chairperson; Directorship: Independent Non-Executive Director
- Name: Dato' Ng Wan Peng; Designation: Member; Directorship: Independent Non-Executive Director
- Name: Dr. Liew Soung Yue; Designation: Member; Directorship: Independent Non-Executive Director

## REMUNERATION COMMITTEE (p. 19)

Table: REMUNERATION COMMITTEE (p. 19)
- Name: Dato' Ng Wan Peng; Designation: Chairperson; Directorship: Independent Non-Executive Director
- Name: Dr. Liew Soung Yue; Designation: Member; Directorship: Independent Non-Executive Director
- Name: Chin Chee Seng; Designation: Member; Directorship: Independent Non-Executive Director

## NOMINATION COMMITTEE (p. 19)

Table: NOMINATION COMMITTEE (p. 19)
- Name: Dr. Liew Soung Yue; Designation: Chairperson; Directorship: Independent Non-Executive Director
- Name: Dato' Ng Wan Peng; Designation: Member; Directorship: Independent Non-Executive Director
- Name: Chin Chee Seng; Designation: Member; Directorship: Independent Non-Executive Director

## COMPANY SECRETARY (p. 20)

- Wong Youn Kim (MAICSA 7018778)
- CCM Practicing Certificate No. 201908000410
- (Chartered Secretary, Associate Member of the Malaysian Institute of Chartered Secretaries & Administrators)
- Acclime Corporate Services Sdn Bhd (199901021060 (495960-D))
- Level 5, Tower 8, Avenue 5, Horizon 2, Bangsar South City, 59200 Kuala Lumpur, Wilayah Persekutuan (KL), Malaysia
- Tel No.: +603 2636 4628
- Fax No.: +603 2280 6399

## REGISTERED OFFICE (p. 20)

- Level 5, Tower 8, Avenue 5, Horizon 2, Bangsar South City, 59200 Kuala Lumpur, Wilayah Persekutuan (KL), Malaysia
- Tel No.: +603 2280 6388
- Fax No.: +603 2280 6399
```

Figure 5.4.2.8 Example of Extracted Section 1 of Company Reports

Figures 5.4.2.9 and 5.4.2.10 below illustrate how the module converts non-textual report content (tables, charts and other visuals) into RAG-ready artifacts. The pipeline detects each table, reads headers, legends and then linearizes the content into header-aware records with preserved units and footnotes, alongside a concise LLM-generated caption and description. The result is stored as Markdown formation with rich metadata, making it easy to chunk and embed for semantic retrieval while retaining exact values and provenance for grounded answers.

2. APPROVALS AND CONDITIONS *(cont'd)*

Details of our Shares held by the Specified Shareholders which will be subject to moratorium are as follows:-

	Moratorium shares during the First 6-Month Moratorium		Moratorium shares during the Second 6-Month Moratorium		Moratorium shares after the Second 6-Month Moratorium	
Specified Shareholders	**No. of Shares**	**(1) %**	**No. of Shares**	**(1) %**	**No. of Shares**	**(1) %**
CYT	169,357,196	30.76	123,862,500	22.50	82,575,000	15.00
CCP	169,357,196	30.76	123,862,500	22.50	82,575,000	15.00
	338,714,392	**61.52**	**247,725,000**	**45.00**	**165,150,000**	**30.00**

Figure 5.4.2.9 Original Table from Company Report

Details of our Shares held by the Specified Shareholders which will be subject to moratorium are as follows:- (p. 24)

Table: [no title] (p. 24)

- Specified Shareholders: CYT; Moratorium shares during the First 6-Month Moratorium No. of Shares: 169,357,196; Moratorium shares during the First 6-Month Moratorium (1) %: 30.76; Moratorium shares during the Second 6-Month Moratorium No. of Shares: 123,862,500; Moratorium shares during the Second 6-Month Moratorium (1) %: 22.50; Moratorium shares after the Second 6-Month Moratorium No. of Shares: 82,575,000; Moratorium shares after the Second 6-Month Moratorium (1) %: 15.00
- Specified Shareholders: CCP; Moratorium shares during the First 6-Month Moratorium No. of Shares: 169,357,196; Moratorium shares during the First 6-Month Moratorium (1) %: 30.76; Moratorium shares during the Second 6-Month Moratorium No. of Shares: 123,862,500; Moratorium shares during the Second 6-Month Moratorium (1) %: 22.50; Moratorium shares after the Second 6-Month Moratorium No. of Shares: 82,575,000; Moratorium shares after the Second 6-Month Moratorium (1) %: 15.00
- Specified Shareholders: [Total]; Moratorium shares during the First 6-Month Moratorium No. of Shares: 338,714,392; Moratorium shares during the First 6-Month Moratorium (1) %: 61.52; Moratorium shares during the Second 6-Month Moratorium No. of Shares: 247,725,000; Moratorium shares during the Second 6-Month Moratorium (1) %: 45.00; Moratorium shares after the Second 6-Month Moratorium No. of Shares: 165,150,000; Moratorium shares after the Second 6-Month Moratorium (1) %: 30.00

Figure 5.4.2.10 Linearized Table Output with Caption & Metadata

This experiment showed that the LLM can indeed ingest a full financial report and return an organized text version covering all the key points, including descriptions of images and tables based on context. However, it also highlighted a significant issue, which it token consumption and performance. Processing an entire PDF of 100+ pages in one go is extremely token-intensive for the LLM. In our tests, a single report could consume many thousands of input tokens and a similarly large number of output tokens in the summary, pushing the limits of the model's context window and incurring substantial API usage cost. The generation process was also time-consuming for very large inputs. Therefore, after extracting the TOC, each section is processed independently and simultaneously. Additionally, per-section processing helps keep generations within the model's output token limit, which often smaller than the input context limit.

In summary, the observations suggest that while directly using an LLM on raw PDFs is feasible and yield good results, it is still costly. Therefore, a hybrid approach with better parsing techniques to trim down the input, then using the LLM in a more targeted way could be more efficient. These learnings will inform the next iteration of the preprocessing pipeline to optimize token usage and runtime.

5.4.3 Chunking & Vector Embedding

Next, once the financial report content was converted into structured data, the next step was to create vector embeddings of this text for semantic search. We used Pinecone as the vector database to store and index the embeddings, and OpenAI's text-embedding-3-small model to generate the embeddings. The text-embedding-3-small model produces a 1536-dimensional numerical representation for each input text, capturing its semantic meaning in vector form. This model is part of OpenAI's third-generation embedding models, chosen for its balance of efficiency (lower cost, smaller size) and good accuracy for general text similarity tasks.

In our implementation, each company report is split into smaller chunks before embedding. Typically, we split the processed data into sections or paragraphs, with each chunk containing a few hundred words. This is showed in the Figure 4.3.1 below. By chunking, we ensure that the embeddings correspond to focused pieces of content, which improves the relevance of search results. Each chunk of text is passed to the OpenAI embedding API, returning a 1536-dimension vector. These vectors are then upserted into the Pinecone index as shown in Figure 5.4.3.

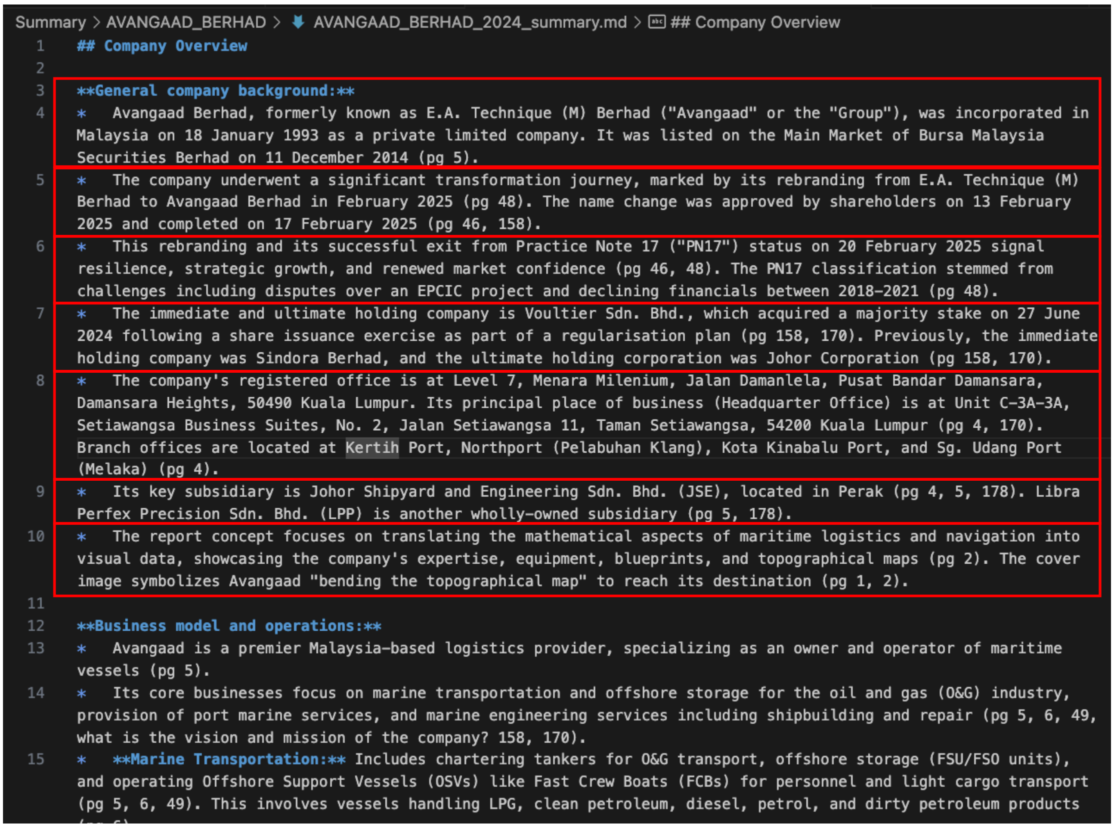


Figure 5.4.3.1 Chunking Example of Extracted Data

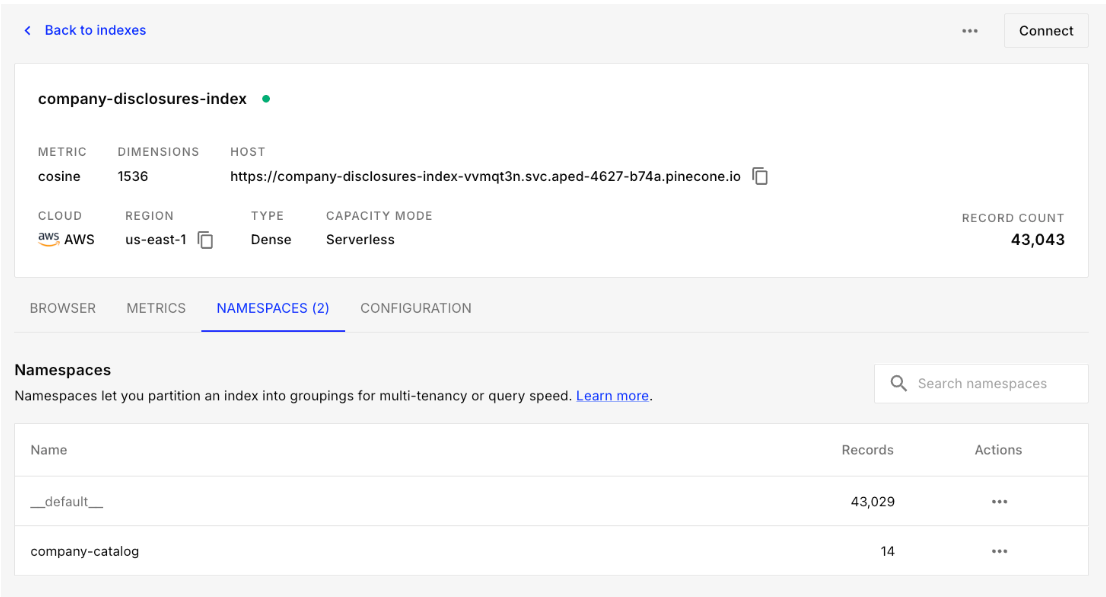


Figure 5.4.3.2 Pinecone Dashboard

Storing the embeddings in a vector database allows us to perform efficient similarity searches. Pinecone is optimized for nearest-neighbour search on vectors, meaning given a new vector like an embedded query, it can quickly return the most similar vectors in the database by using cosine similarity in our case.

5.4.4 Vector RAG

This RAG runs as a lightweight chat app using Gradio ChatInterface as shown in Figure 5.4.4.1 below.

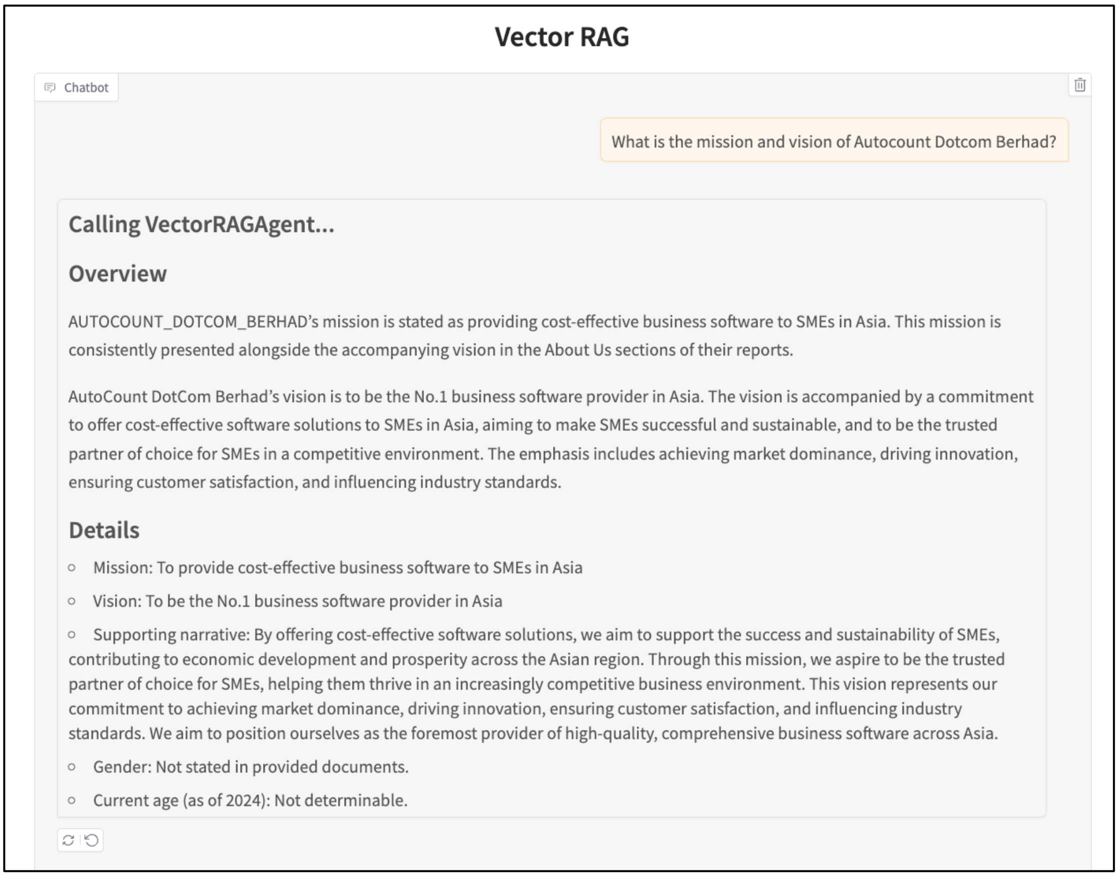
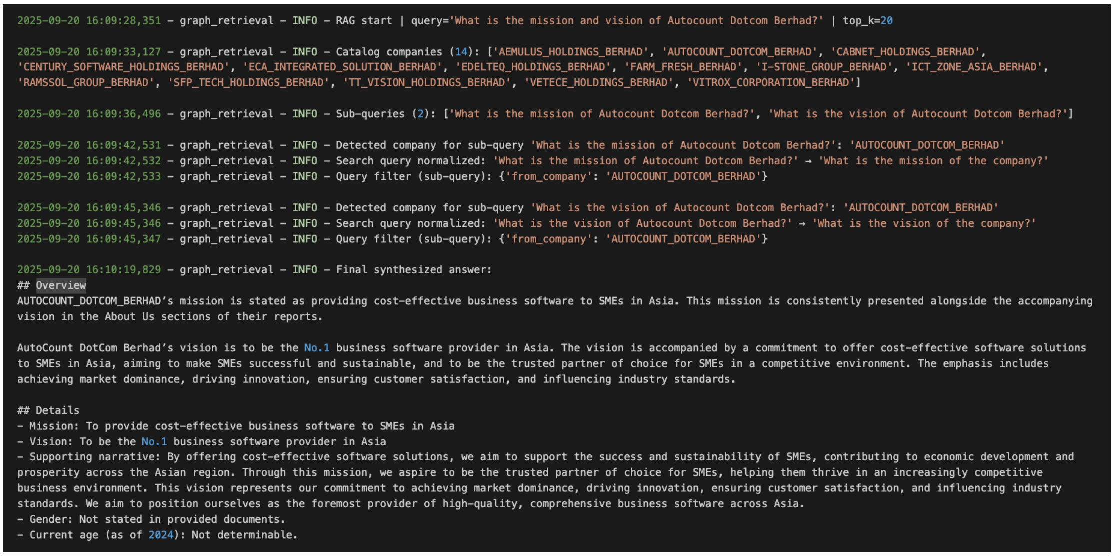


Figure 5.4.4.1 Vector RAG Chat App

In the end-to-end flow, a new RAG request is logged with the raw question ("What is the mission and vision of Autocount Dotcom Berhad"), after which the system loads the catalog of company names to support robust matching. The Query Agent then runs the decompose prompt to split the query into atomic sub-queries ("What is the mission …?", "What is the vision …?"), followed by company detection and normalization where each sub-query is mapped to detected company and rewritten as a company-agnostic, grammatical search string ("What is the mission of the company?") to minimize lexical mismatch while preserving constraints. Each normalized sub-query is embedded with the same model used for the corpus and issued to Pinecone with top_k=20, metric="consine" and a metadata filter such as {"from_company": "AUTOCOUNT_DOTCOM_BERHAD"}. Thus, only that company's chunks are considered. Finally, the Chat Agent receives the original question plus these company-scoped contexts and applies a strict system prompt, that use only provided chunks, group by company, never mix entities and if evidence is insufficient, output "Not found in provided documents". Figure 5.4.4.2 shows the log of this end-to-end flow, that shows each stage including RAG start, sub-queries, detected company, normalized query, Pinecone filter used, final synthesized answer.



```
2025-09-20 16:09:28,351 - graph_retrieval - INFO - RAG start | query='What is the mission and vision of Autocount Dotcom Berhad?' | top_k=20

2025-09-20 16:09:33,127 - graph_retrieval - INFO - Catalog companies (14): ['AEMULUS_HOLDINGS_BERHAD', 'AUTOCOUNT_DOTCOM_BERHAD', 'CABNET_HOLDINGS_BERHAD',
'CENTURY_SOFTWARE_HOLDINGS_BERHAD', 'ECA_INTEGRATED_SOLUTION_BERHAD', 'EDELTEQ_HOLDINGS_BERHAD', 'FARM_FRESH_BERHAD', 'I-STONE_GROUP_BERHAD', 'ICT_ZONE_ASIA_BERHAD',
'RAMSSOL_GROUP_BERHAD', 'SFP_TECH_HOLDINGS_BERHAD', 'TT_VISION_HOLDINGS_BERHAD', 'VETECE_HOLDINGS_BERHAD', 'VITROX_CORPORATION_BERHAD']

2025-09-20 16:09:36,496 - graph_retrieval - INFO - Sub-queries (2): ['What is the mission of Autocount Dotcom Berhad?', 'What is the vision of Autocount Dotcom Berhad?']

2025-09-20 16:09:42,531 - graph_retrieval - INFO - Detected company for sub-query 'What is the mission of Autocount Dotcom Berhad?': 'AUTOCOUNT_DOTCOM_BERHAD'
2025-09-20 16:09:42,532 - graph_retrieval - INFO - Search query normalized: 'What is the mission of Autocount Dotcom Berhad?' → 'What is the mission of the company?'
2025-09-20 16:09:42,533 - graph_retrieval - INFO - Query filter (sub-query): {'from_company': 'AUTOCOUNT_DOTCOM_BERHAD'}

2025-09-20 16:09:45,346 - graph_retrieval - INFO - Detected company for sub-query 'What is the vision of Autocount Dotcom Berhad?': 'AUTOCOUNT_DOTCOM_BERHAD'
2025-09-20 16:09:45,346 - graph_retrieval - INFO - Search query normalized: 'What is the vision of Autocount Dotcom Berhad?' → 'What is the vision of the company?'
2025-09-20 16:09:45,347 - graph_retrieval - INFO - Query filter (sub-query): {'from_company': 'AUTOCOUNT_DOTCOM_BERHAD'}

2025-09-20 16:10:19,829 - graph_retrieval - INFO - Final synthesized answer:
## Overview
AUTOCOUNT_DOTCOM_BERHAD's mission is stated as providing cost-effective business software to SMEs in Asia. This mission is consistently presented alongside the accompanying
vision in the About Us sections of their reports.

AutoCount DotCom Berhad's vision is to be the No.1 business software provider in Asia. The vision is accompanied by a commitment to offer cost-effective software solutions
to SMEs in Asia, aiming to make SMEs successful and sustainable, and to be the trusted partner of choice for SMEs in a competitive environment. The emphasis includes
achieving market dominance, driving innovation, ensuring customer satisfaction, and influencing industry standards.

## Details
- Mission: To provide cost-effective business software to SMEs in Asia
- Vision: To be the No.1 business software provider in Asia
- Supporting narrative: By offering cost-effective software solutions, we aim to support the success and sustainability of SMEs, contributing to economic development and
prosperity across the Asian region. Through this mission, we aspire to be the trusted partner of choice for SMEs, helping them thrive in an increasingly competitive
business environment. This vision represents our commitment to achieving market dominance, driving innovation, ensuring customer satisfaction, and influencing industry
standards. We aim to position ourselves as the foremost provider of high-quality, comprehensive business software across Asia.
- Gender: Not stated in provided documents.
- Current age (as of 2024): Not determinable.
```

Figure 5.4.4.2 Vector RAG Log

5.5 Implementation Issues and Challenges

One of the primary implementation challenges would be the web scraping of a very large number of financial documents. Sequentially fetching such a volume would result in a severe performance bottleneck, as fetching thousands of pages one by one is very time consuming. Not only that, sending a high frequency of requests may raise the risk of triggering anti-scraping mechanisms on target websites. For instance, Cloudflare's bot protection in Bursa Malaysia could result in the scraper being blocked. To address these issues, the system's web scraping agent was designed to use multithreading for parallel downloads. This can improve the overall throughput and reduce the total crawl time. At the same time, careful backoff strategies such as inserting delays and monitoring for throttling responses were implemented to avoid the target servers from overloading. Therefore, the agent will be under the detection thresholds and minimizes the likelihood of being blocked.

Processing lengthy financial documents with LLMs introduced significant computational complexity. Feeding such large texts into LLMs would also incur a huge computational cost. To overcome these limitations, the implementation must rely on token reduction techniques and intelligent chunking strategies. For instance, before summarization, unnecessary or repetitive parts of the text are removed to reduce the number of input's token. Each report is then split into smaller chunks that fit within the LLM context window while preserving the logical structure. Determining the optimal chunk size requires careful tuning. Chunks must be large enough to preserve important context, but small enough to stay within the capacity of the model.

Other than that, LLMs also poses a challenge in terms of its output reliability, as they can sometimes produce information that does not present in the source context, a phenomenon known as "hallucination". This hallucination is particularly problematic in the context of summary financial reporting, as any fabricated facts or figures can mislead the analysis and undermine the credibility of the system. This risk can be mitigated. By tuning the LLMs' generation parameters and incorporating with some additional verification steps.

A further challenge encountered during implementation was provider-side safety enforcement. When processing some company reports, the hosted LLM intermittently returned RECITATION errors and terminated the process, likely due to the model's policy guardrails against long verbatim reproduction from copyrighted material. The only deterministic workaround would be to self-host an open-source LLM and fine-tune it with domain-specific data and adjusted safety settings. However, this requires substantial GPU resources, expertise, and ongoing maintenance, which is costly and beyond the scope of this project.

Lastly, as the reports grew, ensuring reliable storage and efficient data retrieval becomes a critical issue. The system needs to handle not only the original financial documents themselves, but also derived data such as summaries and their embeddings, which represent a large amount of information.

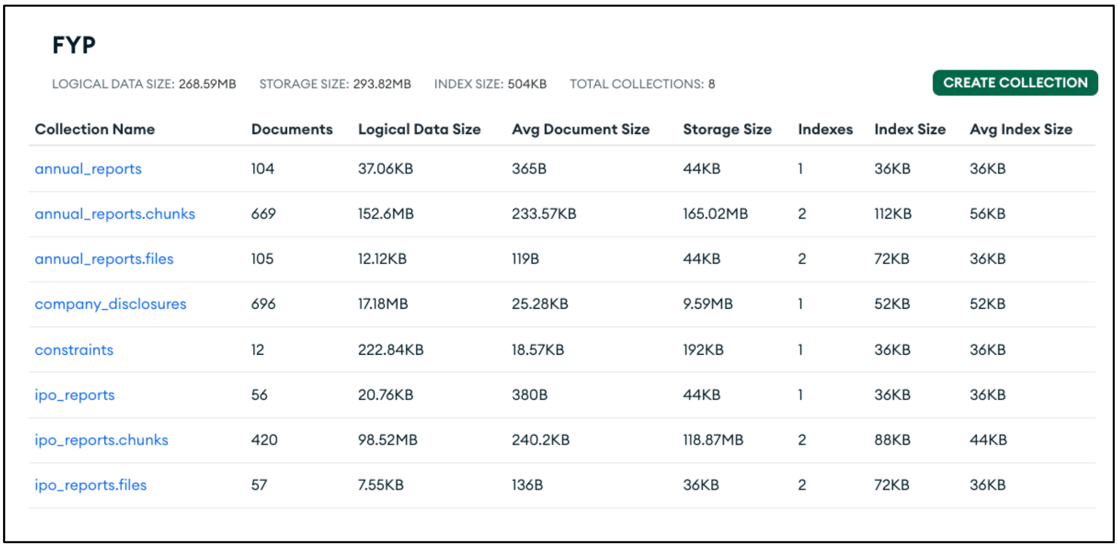
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing and Result

The system's web scraping agent collected company filings by crawling the Bursa Malaysia announcements pages and downloading linked PDFs. In practice, it targeted about 30 technology-sector companies, fetching roughly 5-10 financial documents each. For each company, the scraper ran listing crawls and parsed announcement pages in loop, then downloaded every new PDF it found. The downloaded files were stored in database along with metadata like company name and report year. Most pages on Bursa Malaysia are static HTML, so the scraper primarily used direct HTTP requests and HTML parsing (with BeautifulSoup) to extract PDF links. After fetching each announcement page, BeautifulSoup or similar parsing logic finds the table of attachments and extracts all PDF URLs in it.

In initial tests the agent successfully scraped and saved a batch of annual reports from multiple companies as proof of concept. It stored each PDF as a binary file in GridFS, indexed by company and year. The agent's pipelined design lets it handle many documents in parallel queues.

Heavy scraping risks triggering anti-bot defences. Indeed, one observation was that high-frequency requests can lead to HTTP errors or temporary bans on the site. To mitigate anti-bot blocking from high-frequency requests, use "curl_cffi" for HTTP fetching. It is a Python binding to curl-impersonate that can impersonate real browsers' TLS and HTTP fingerprints, which helps bypass fingerprinting-based defences that often flag requests. Overall, the scraping module proved robust in collecting the needed reports across the target companies.



FYP

LOGICAL DATA SIZE: 268.59MB STORAGE SIZE: 293.82MB INDEX SIZE: 504KB TOTAL COLLECTIONS: 8

CREATE COLLECTION

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
annual_reports	104	37.06KB	365B	44KB	1	36KB	36KB
annual_reports.chunks	669	152.6MB	233.57KB	165.02MB	2	112KB	56KB
annual_reports.files	105	12.12KB	119B	44KB	2	72KB	36KB
company_disclosures	696	17.18MB	25.28KB	9.59MB	1	52KB	52KB
constraints	12	222.84KB	18.57KB	192KB	1	36KB	36KB
ipo_reports	56	20.76KB	380B	44KB	1	36KB	36KB
ipo_reports.chunks	420	98.52MB	240.2KB	118.87MB	2	88KB	44KB
ipo_reports.files	57	7.55KB	136B	36KB	2	72KB	36KB

Figure 6.1.1 MongoDB Dashboard

Once documents are scraped, they are converted into machine-readable text and structured data by LLMs. A key challenge is token usage. Naïve text extraction from a large PDF can produce extremely long strings. In testing, a single annual report yielded "tens of thousands of words," translating to a very large token count for an LLM. This inflates API costs and may exceed context limits. To address this, the project experimented with a "PDF-to-LLM" method where an LLM (Google's Gemini 2.5 Pro via the file-upload API) is prompted to read the entire PDF and output structured content. Interestingly, the direct PDF-to-LLM method used fewer tokens overall than the raw-text approach. This is because the LLM can ignore repeated boilerplate when generating structured content, whereas a blind text dump contains all content verbatim. In practice, the system feeds each report to the LLM to extract key sections, definitions, and figures of interest in markdown/JSON format.

Even so, processing dozens of long reports can approach rate or usage quotas. In our experiments on a free-tier API account, we often hit error 429 ("quota exceeded") once the free credit was consumed. This meant only about 10 companies' reports could be fully processed before hitting the limit. Occasional 500 Internal Server Errors also appeared, indicating transient issues on the service side, requiring simple retries.

The solution was to upgrade to a paid plan. In fact, the documentation notes that 429 errors mean the monthly or usage quota has been reached. After adding billing credentials, the errors ceased as the account moved to pay-as-you-go. Therefore, the system could run the TOC-guided sections extraction pipeline concurrently, processing multiple sections in parallel without tripping quota errors. Concretely, Gemini 2.5 Pro's rate limits jumped from the Free tier (5 RPM, 250000 TPM, 100 RPD) to the default Tier 1 paid limits (150 RPM, 2000000 TPM, 10000 RPD), which removes the earlier 429 bottleneck and comfortably supports simultaneous section extraction.

Overall, while the model generally produced useful structured information of each report, outputs were always reviewed for accuracy. The logs confirmed that careful prompting and validation are necessary to prevent or catch hallucinations.

As for the Vector RAG module, which is the system's Q&A engine, it was evaluated through response correctness by comparing RAG outputs to known facts. In tests, straightforward factual questions were answered correctly by the RAG system, matching the values in the actual reports. For more open-ended queries, the answers synthesized multiple relevant excerpts. This result was also cross-checked with the external sources, such as a Google search or even a plain ChatGPT query sometimes yielded a conflicting or made-up answer, whereas the system's RAG stuck to the source content. Though our domain is financial text, this suggests RAG markedly improves factual accuracy and source citation compared to unguided generation.

In evaluating the retriever component, precision measures how many of the top-k retrieved chunks are relevant, while Recall measures how many of the truly relevant chunks the retriever found out of all relevant chunks. In our case, high recall is crucial. Therefore, the retriever should fetch all pieces needed to answer the questions. Any missed chunk could leave out key facts. In practice, the system used a relatively large k to ensure recall, trusting that the LLM can ignore any extra noise. Indeed, if only 1 of 10 chunks has the answer, precision is low but recall is 100%, which is acceptable here because the LLM then filters out itself.

Overall, the retrieval step worked well on our company corpus, and the generated answers generally remained faithful to the evidence. Any incorrect or incomplete answer could be traced to either a missing snippet or a generation slip, guiding us to iteratively improve chunk coverage and prompts.

The following screenshots shows some results of the Vector RAG module comparing with answer generated using ChatGPT.

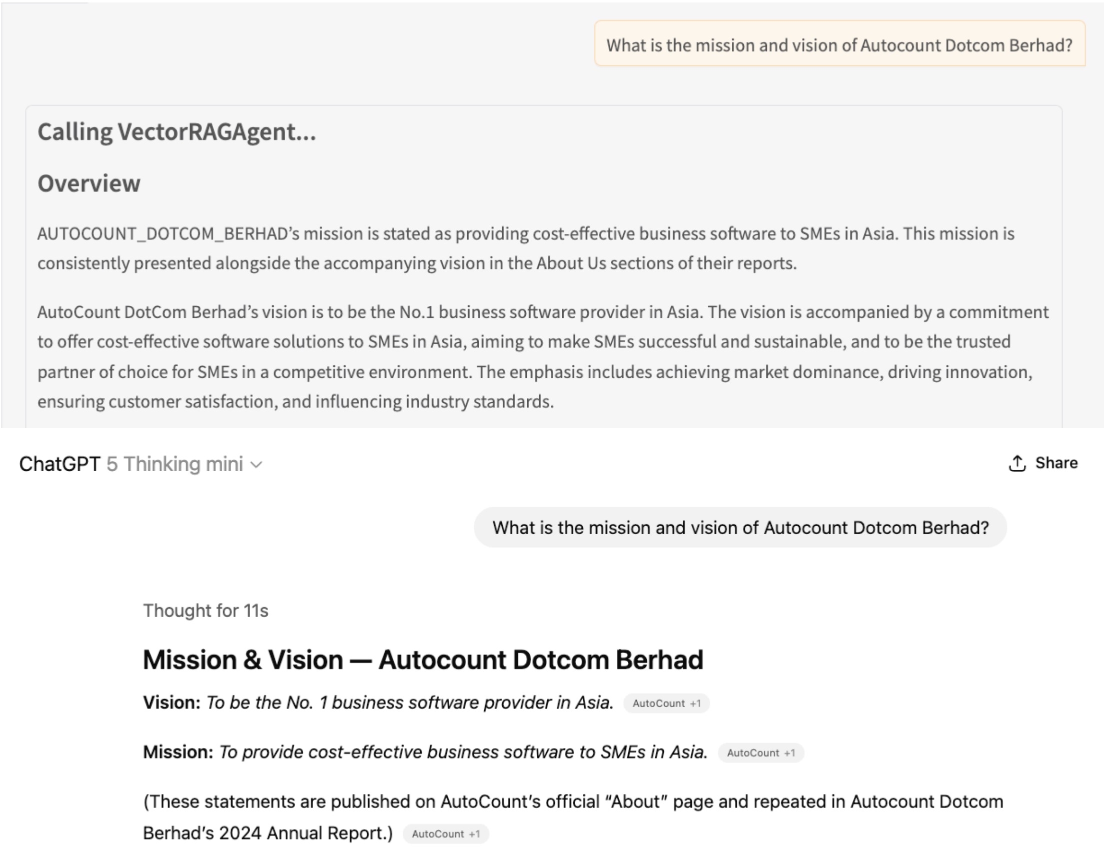


Figure 6.1.2 Mission & Vision Answers (Vector RAG vs ChatGPT)

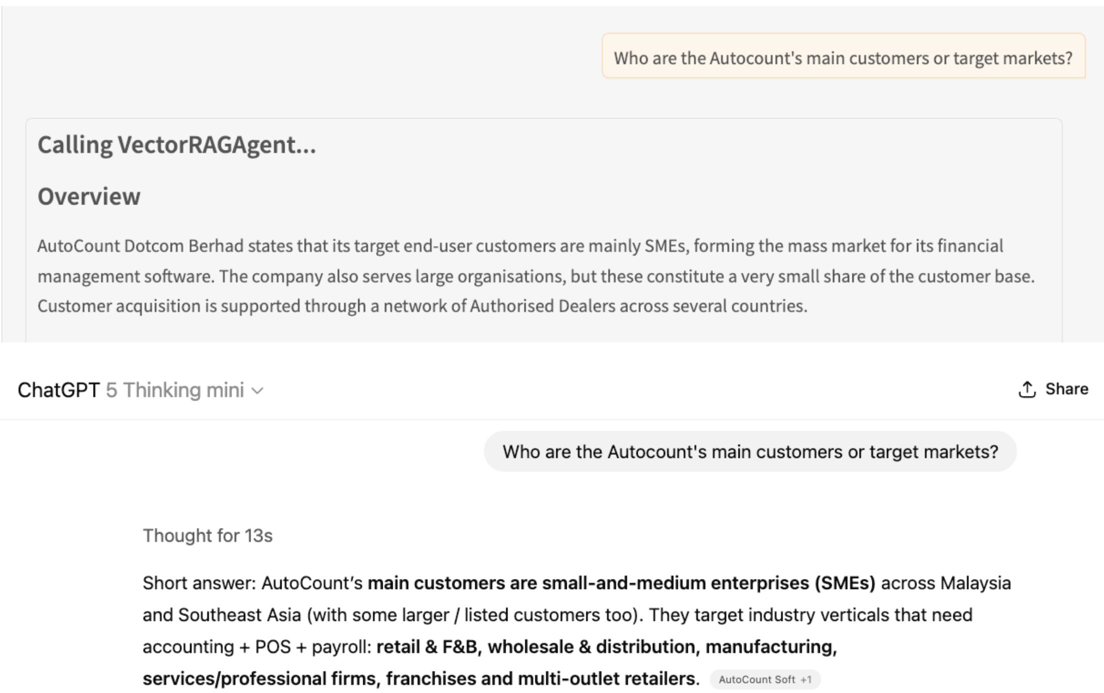
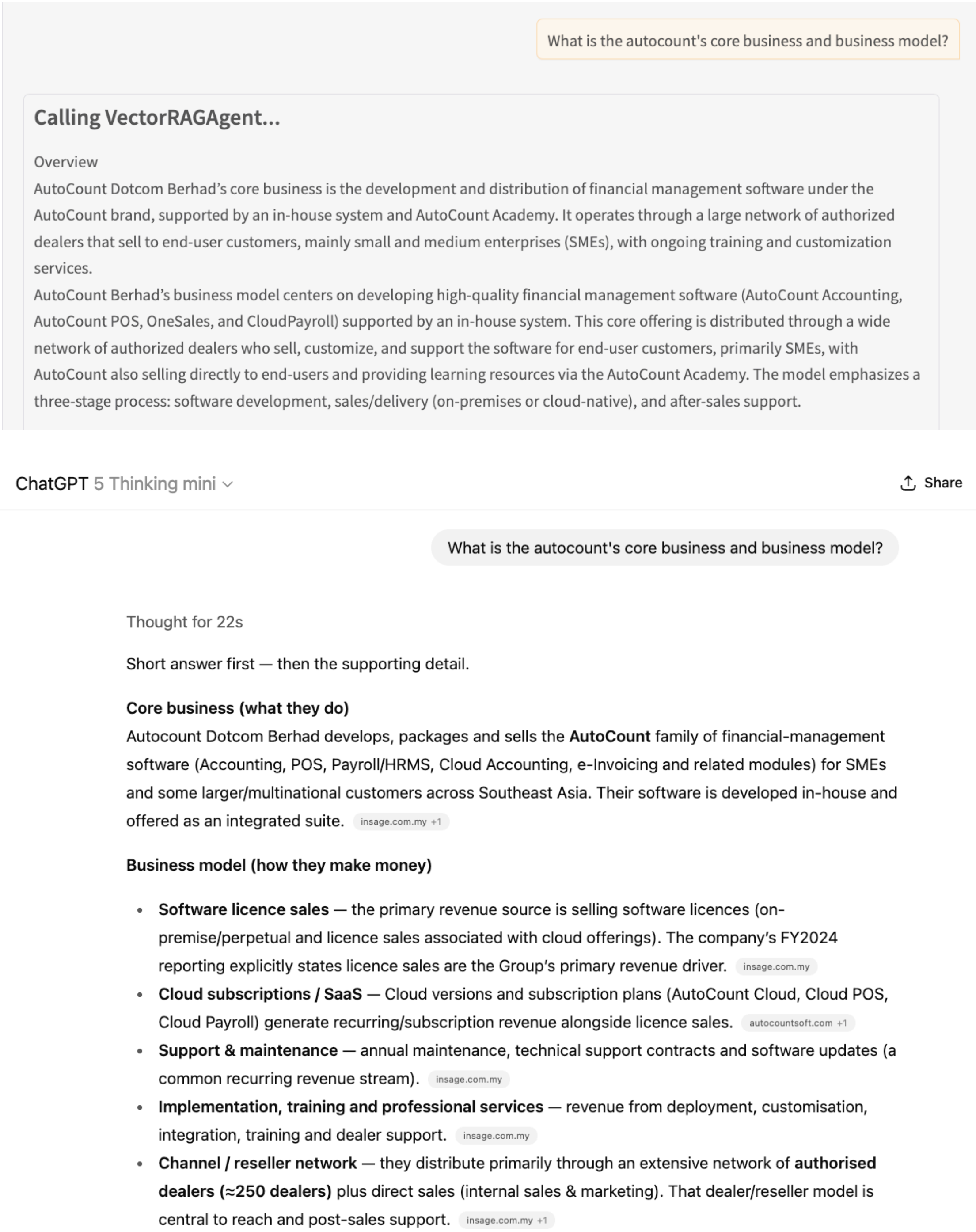


Figure 6.1.3 Main Customers & Target Markets Answers (Vector RAG vs ChatGPT)



What is the autocount's core business and business model?

Calling VectorRAGAgent...

Overview

AutoCount Dotcom Berhad's core business is the development and distribution of financial management software under the AutoCount brand, supported by an in-house system and AutoCount Academy. It operates through a large network of authorized dealers that sell to end-user customers, mainly small and medium enterprises (SMEs), with ongoing training and customization services.

AutoCount Berhad's business model centers on developing high-quality financial management software (AutoCount Accounting, AutoCount POS, OneSales, and CloudPayroll) supported by an in-house system. This core offering is distributed through a wide network of authorized dealers who sell, customize, and support the software for end-user customers, primarily SMEs, with AutoCount also selling directly to end-users and providing learning resources via the AutoCount Academy. The model emphasizes a three-stage process: software development, sales/delivery (on-premises or cloud-native), and after-sales support.

ChatGPT 5 Thinking mini Share

What is the autocount's core business and business model?

Thought for 22s

Short answer first — then the supporting detail.

Core business (what they do)

Autocount Dotcom Berhad develops, packages and sells the **AutoCount** family of financial-management software (Accounting, POS, Payroll/HRMS, Cloud Accounting, e-Invoicing and related modules) for SMEs and some larger/multinational customers across Southeast Asia. Their software is developed in-house and offered as an integrated suite. insage.com.my +1

Business model (how they make money)

- **Software licence sales** — the primary revenue source is selling software licences (on-premise/perpetual and licence sales associated with cloud offerings). The company's FY2024 reporting explicitly states licence sales are the Group's primary revenue driver. insage.com.my
- **Cloud subscriptions / SaaS** — Cloud versions and subscription plans (AutoCount Cloud, Cloud POS, Cloud Payroll) generate recurring/subscription revenue alongside licence sales. autocountsoft.com +1
- **Support & maintenance** — annual maintenance, technical support contracts and software updates (a common recurring revenue stream). insage.com.my
- **Implementation, training and professional services** — revenue from deployment, customisation, integration, training and dealer support. insage.com.my
- **Channel / reseller network** — they distribute primarily through an extensive network of **authorised dealers (≈250 dealers)** plus direct sales (internal sales & marketing). That dealer/reseller model is central to reach and post-sales support. insage.com.my +1

Figure 6.1.4 Core Business & Business Model Answers (Vector RAG vs ChatGPT)

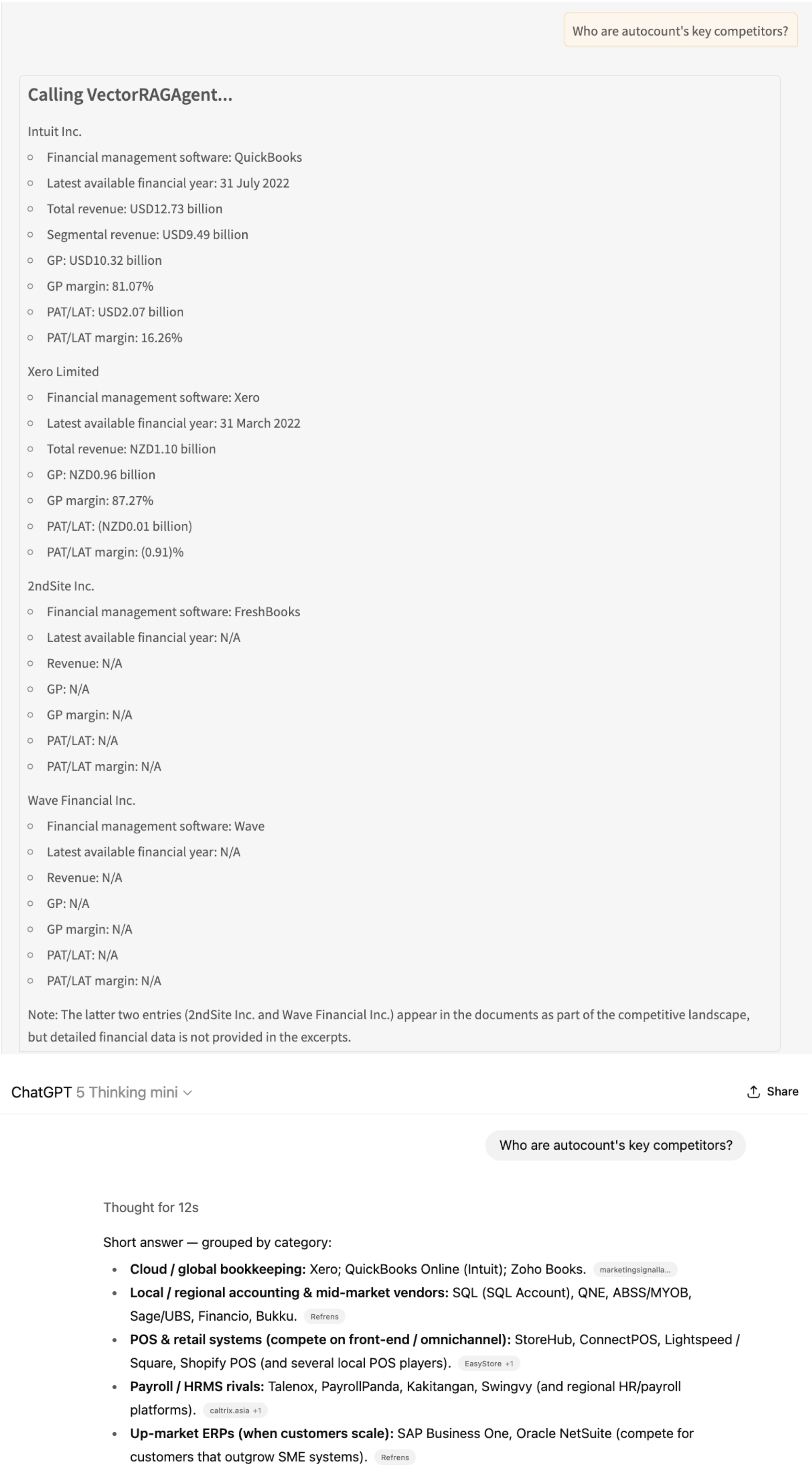


Figure 6.1.5 Key Competitors Answers (Vector RAG vs ChatGPT)

6.2 Objectives Evaluation

The implemented system successfully illustrates a multi-agent architecture. We have separate components for crawling listing pages, parsing announcement pages, downloading PDFs, embedding text, and querying the LLM. These components operate in parallel and communicate through shared resources like the database and queues, effectively mimicking a team of analyst working simultaneously. In testing, this autonomy was demonstrated, where the system collected and processed new reports without further human intervention. This meets the first objective of an end-to-end automated workflow.

The second objective was to use LLMs to interpret qualitative financial text. Our system meets this by employing state-of-the-art LLMs (Gemini 2.5 Pro) to digest and extract information from each report's narrative sections. The generated outputs include key facts presented in structured markdown/JSON. In practice, the LLM extracts were generally accurate. For example, descriptions of the company's business model, management commentary, and strategy were correctly captured. The LLM successfully handled complex language and jargon in the reports, subject to the grounding provided by retrieval. Importantly, the evidence shows the combination of retrieval and generation led to reliable answers. A RAG approach yields higher factual accuracy than generation alone, and the system's results reflect this. Thus, the system largely achieves the second objective of leveraging LLMs for deep qualitative interpretation of reports.

In conclusion, the system successfully achieved its project objectives. The vector database of embedded report text was built as the core module enabling semantic search. The system's pipeline autonomously gathers and processes reports, the LLM component effectively interprets qualitative content and the retrieval mechanism ensures that answers are grounded in the latest information. Overall, the evaluation shows that the system meets its design goals, providing accurate and timely company insights.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

7.1 Conclusion

This project set out to address the challenge of fundamental stock analysis using qualitative data by leveraging LLMs in an innovative, data-driven system. The core problem to be solved is that crucial information on stock valuation is often hidden in lengthy company reports and other unstructured documents, making it difficult for investors, especially retail investors to effectively extract insights. By integrating cutting-edge AI technology, the project aims to bridge this gap and provide non-expert investors with a “virtual analyst” who can read, understand and summarize complex financial disclosures.

To achieve this goal, the project implemented a comprehensive multi-agent system consisting of several key components. The architectural innovation introduced by the project is that it uses multiple specialized agents to coordinate the execution of a single analytical task. Each agent is optimized for a different function (data collection, preprocessing, storage/retrieval, and analysis), which makes the entire system modular and robust. Essentially, this project demonstrates a cutting-edge approach where LLMs are augmented with external memory (via vector embeddings), enabling them to generate sensible analytics on large-scale financial texts.

The successful implementation of this approach marks a significant step forward in the application of AI in fundamental analysis. It demonstrates that generative AI can create tremendous value in distilling complex financial disclosures into easily digestible intelligence. Tasks such as reading an annual report of more than 100 pages and assessing its tone and key points, which once performed only by experienced analysts can now be partially automated through LLM-based summarization and Q&A capabilities. This capability is particularly meaningful for retail investors, where the system lowers the barrier to thorough stock research by automatically synthesizing information that would require hours of manual reading.

The project helps to narrow the information asymmetry between retail investors and institutional investors with large research teams. In short, the system can serve as a proof-of-concept "AI analyst" to enhance decision support capabilities and make stock fundamental analysis more accessible to investors with limited time or financial expertise.

7.2 Recommendation

For scraping, adopt an on-demand acquisition policy rather than continuously crawling everything. Concretely, trigger the scraper only when a user query or downstream task requires a report that is missing. This event-driven approach reduces bandwidth, compute and storage, while keeping the essential documents up to date. Pair it with a small cache and a recency index so repeated requests do not re-download the same files and keep polite, bounded concurrency with retries for resilience.

For LLM-based report processing, restrict inference to section-scoped chunks before calling the model. Use the extracted table of contents to isolate only the sections required for the task and then subdivide those sections so they fit comfortably within the model's context window. This targeted, TOC-guided chunking reduces token usage, prevents context overflows and lowers latency, while still preserving coherence because the boundaries follow the document's own structure. Enforce a strict max tokens limit for each call and apply iterative or hierarchical summarization only when a section still exceeds those limits.

For retrieval, prioritize a systematic study of pre-embedding chunking to preserve semantic meaning. Future work should compare approaches such as TOC-aligned chunks versus sentence or paragraph window, fixed window sizes versus adaptive windows and different overlaps to maintain cross-boundary context. Each configuration should be evaluated with retrieval and answer-quality metrics as well as operational metrics. The objective is to identify a chunking strategy that maximizes semantic recall and downstream answer fidelity while minimizing cost and storage overhead.

REFERENCES

[1] A. Lopez-Lira and Y. Tang, ‘Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models’, Apr. 2023, [Online]. Available: https://doi.org/10.48550/arxiv.2304.07619.

[2] B. Graham and D. Dodd, Security Analysis, 6th ed. Whittlesey House, McGraw-Hill Book Co., 2009.

[3] S. R. Bentes and R. Navas, “The Fundamental Analysis: An Overview,”, Mar. 2013, [Online]. Available: https://www.researchgate.net/publication/340511850_The_Fundamental_Analysis_An_Overview

[4] D. Araci, “FinBERT: Financial Sentiment Analysis with Pre-trained Language Models,” arXiv (Cornell University), Aug. 2019, doi: https://doi.org/10.48550/arxiv.1908.10063.

[5] R. Wu, “Portfolio Performance Based on LLM News Scores and Related Economical Analysis,” SSRN Electronic Journal, Jan. 2024, doi: https://doi.org/10.2139/ssrn.4709617.

[6] A. G. Kim, M. Muhn, and V. V. Nikolaev, “Bloated Disclosures: Can ChatGPT Help Investors Process Information?” Jan. 2023, doi: https://doi.org/10.2139/ssrn.4425527.

[7] U. Gupta, “GPT-InvestAR: Enhancing Stock Investment Strategies through Annual Report Analysis with Large Language Models,” SSRN Electronic Journal, 2023, doi: https://doi.org/10.2139/ssrn.4568964.

[8] “Financial Statement Analysis with Large Language Models,” 2024, [Online]. Available: https://arxiv.org/html/2407.17866v1.

[9] “FinSphere: A Conversational Stock Analysis Agent Equipped with Quantitative Tools based on Real-Time Database,” Arxiv.org, 2023. https://arxiv.org/html/2501.12399v1.

[10] virattt, “GitHub - virattt/ai-hedge-fund: An AI Hedge Fund Team,” GitHub, 2024. https://github.com/virattt/ai-hedge-fund

[11] O. Erdem, “Can Chatgpt Outsmart the Market?,” 2025, doi: https://doi.org/10.2139/ssrn.5106339.

[12] K. Reitz, “requests: Python HTTP for Humans.,” PyPI, May 22, 2023. https://pypi.org/project/requests/

[13] “The Selenium Browser Automation Project,” Selenium. https://www.selenium.dev/documentation/

[14] “ Playwright,” GitHub, Jul. 03, 2022. https://github.com/microsoft/playwright

[15] L. Richardson, “beautifulsoup4,” PyPI, Oct. 06, 2019. https://pypi.org/project/beautifulsoup4/

[16] MongoDB, “The most popular database for modern apps,” MongoDB, 2024. https://www.mongodb.com/

[17] “Vector Database for Vector Search | Pinecone,” www.pinecone.io. https://www.pinecone.io/

APPENDIX

Poster