Team 86

Pathway

End Term Submission

Table of Contents

Introduction

Our project is centered on developing a transformative Agentic RAG (Retrieval-Augmented Generation) system leveraging Pathway's advanced dynamic capabilities. Our primary aim was to design a solution that autonomously retrieves, analyzes, and synthesizes information from diverse documents within a novel context, effectively addressing a real-world need.

The key objectives of our project include:

- **Identifying a Relevant Problem:** Discovering a real-world problem where an Agentic RAG system with Pathway's capabilities could provide significant benefits.
- **Ensuring Accurate Responses:** Delivering precise and reliable answers to complex and varied queries.
- **Enhancing Autonomy and Agency:** Developing an autonomous system equipped with capable agents that can better solve identified challenges.
- **Showcasing Value and Innovation:** Ensuring each element of our project demonstrates clear value, rather than existing merely to fulfill a checklist.

Use-Case Selection and Novelty

Space:

Financial due diligence represents a highly resource-intensive, time-consuming and painstakingly manual process, often extending over several weeks or months. This complexity arises from the need for exhaustive, detailed analysis and reasoning across financial datasets, which often require frequent updates.

Use-Case:

Our solution, designed around the core objectives outlined earlier, directly addresses the key pain points of financial due diligence (FDD). It is tailored to serve all of the stakeholders of FDD, from investors to analysts and lawyers, by automating document analysis, accurately responding to queries, and dynamically adapting to changing document datasets. Additionally, it generates summarized, concise FDD reports and quick-look dashboards for targeted firms, offering a strong starting point in the due diligence process. It can also be used independently to perform Q&A on financial documents.

Novelty:

While there are generic RAG-based systems available for Q&A on documents, these have moderate accuracy and often suffer from high scope for error. Given the high stakes involved in FDD, these systems cannot be relied upon. Few firms are attempting to automate this process by offering tools for various aspects of FDD, but not addressing the core issues we focus on. As such, the competitive landscape is sparse, making our selected use case particularly novel and impactful.

Uniqueness

- **Real-World Application:** Our solution is grounded in a robust, real-world use case that capitalizes on Pathway's capabilities, specifically targeting the complexities of financial due diligence (FDD).
- **Versatile Functionality:** Beyond streamlining the FDD process, our solution can also act as an agentic RAG-based chatbot for financial documents, capable of answering complex queries related to financial documents.
- **Innovative Architecture:** The novel architecture of our implementation is a major strength, characterized by its simplicity, elegance, and effectiveness. This is validated by both theoretical intuitions and empirical results.

- **Unmatched Offering:** There are no existing closed or open-source solutions that parallel the comprehensive and innovative capabilities we deliver for our specific use case of enhancing the FDD process.

Solution Overview

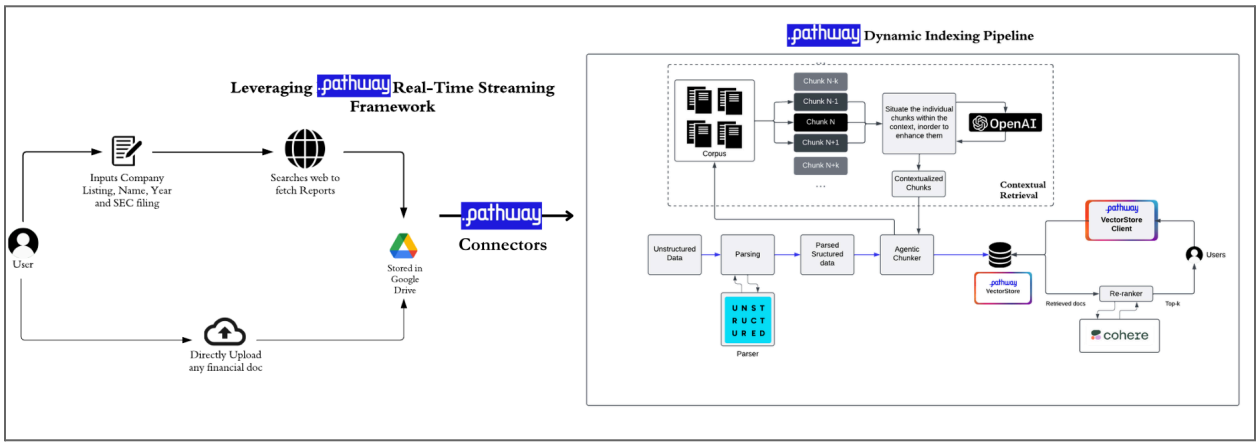
The aim is to streamline financial due diligence into a less manual, more efficient, elegant, and significantly faster process. Hence our solution, **F**inancial **A**gentic **A**utonomous and **A**ccurate **S**ystem **T**hrough **E**volving (Dynamic) **R**etrieval Augmented Generation, or **FA^{3}STER**, is designed to address the challenges of FDD. Here's an outline of FA^{3}STER's design: At the heart of FA^{3}STER lies a sophisticated four-component architecture:

- **The Pre-Retrieval and Retrieval Phase** introduces an Agentic Chunker that enhances document parsing by implementing contextual retrieval[1]. This approach allows for more nuanced and context-aware information extraction, improving the accuracy of subsequent analysis. This entire phase is built using Pathway's real-time streaming framework coupled with its dynamic indexing pipeline, enhanced with custom classes we've developed for the Agentic Chunker and Parser. (This addresses the Extra/Bonus section of the PS)
- **The Post-Retrieval Phase** focused on building a strong, autonomous agentic framework capable of making intelligent decisions and selecting optimal workflow paths. We created a suite of specialized agents such as finance, document grading, SQL, charting agents, etc—each designed to handle specific analytical tasks with precision. We have implemented robust error handling, resilience strategies and fallback strategies to support our system in case of failures. (This addresses Level 2 of the PS)
- **The Vertical Autonomous Layer** is our most innovative approach to accelerating financial due diligence. Instead of simply adding more tools and agents to the post-retrieval architecture without much value addition, we created an intelligent layer of inter-networked agents that leverage our RAG system as a powerful foundation. These agents can query the system, discuss findings amongst themselves, exchange knowledge, and provide cross-agent suggestions through multiple iterative rounds. The result is a truly unique and technically exciting approach that transforms FA^{3}STER from a passive tool into an active intelligence platform, ultimately generating a concise, summarized FDD report and an actionable dashboard for stakeholders!
- **The UI** is designed to provide both simplicity and transparency, offering users a clear window into the complex processes occurring behind the scenes. The UI features a dynamic graph that visualizes the entire agent workflow in real time, highlighting the current nodes and operational states.

System Architecture

Our multi-agent RAG system for financial analysis prioritizes robust performance, seamless integration, and accuracy. The system leverages Pathway's Dymnamic Indexing Pipeline including the Vector Store and Google Drive connector for dynamic data management. An Agentic Chunker, powered by GPT-4o-mini, enriches document fragments with contextual information for enhanced retrieval precision. Unstructured data is processed via Unstructured's Parser, while OpenAI's text-embedding-3-small model generates embeddings for hybrid indexing, combining semantic search with metadata filtering. Complex queries are decomposed and ranked using Cohere's Re-ranker. Post-retrieval processing employs LangGraph and specialized agents within a modified SELF-RAG architecture. Tavily Search integrates external data, and a reasoning agent generates insightful visualizations. A detailed examination of these components follows.

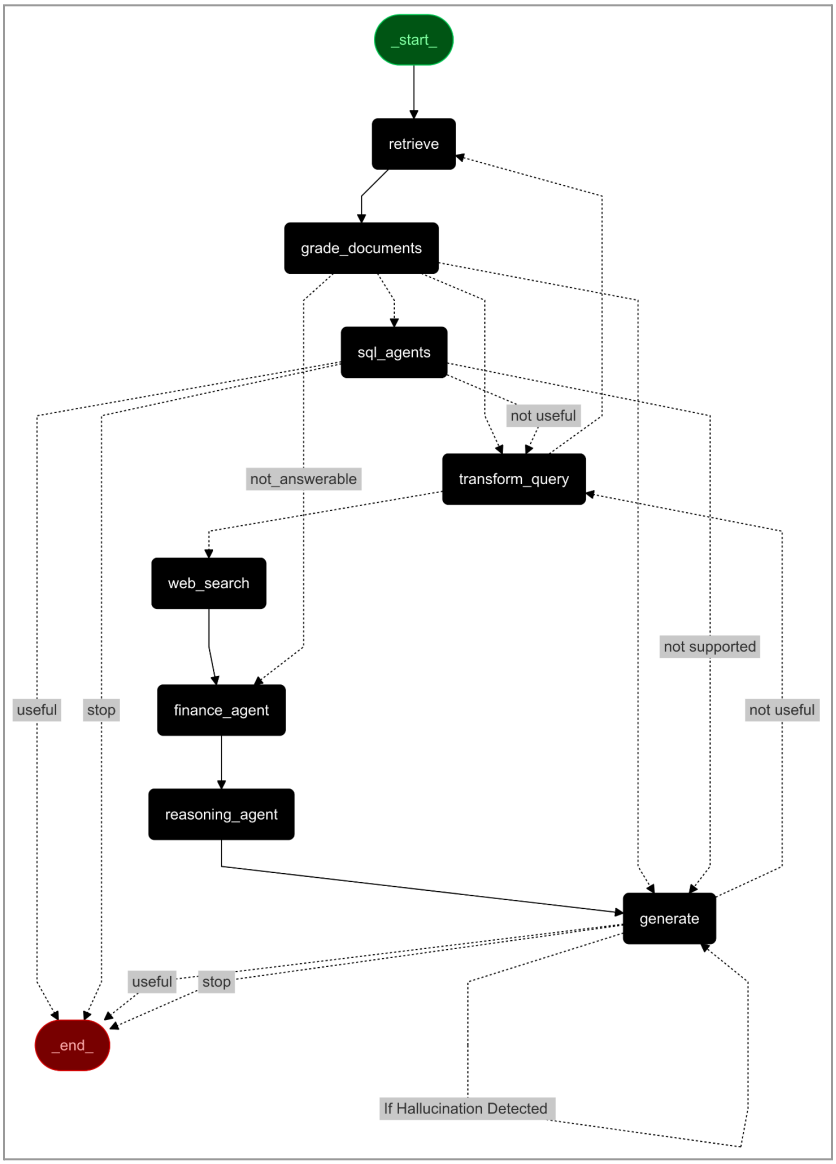
Pre-Retrieval and Retrieval



1. **Pathway's Real-Time Streaming Framework**
   Pathway's Google Drive connector dynamically streamlines data ingestion into the RAG system, supporting both real-time updates and one-time imports. Its configurable parameters and UNIX-like path syntax provide efficient access to files and folders within Google Drive.
2. **Pre-Retrieval Workflow and Data Processing**
   Since the documents are mainly in pdf format, we need to have them converted to embeddings before using them for retrieval. For that we have to parse the document, chunk and embed it and finally store it in the vector store.
   - **Unstructured's Parser**: Unstructured.io's parsing tools excel at extracting structured data from various financial document formats, including tables, lists, and key-value pairs.By preserving document structure, Unstructured's parser facilitates more accurate and context-aware analysis of financial reports.
   - **Agentic Chunker**: Leveraging contextual retrieval, the Agentic Chunker utilizes GPT-4o-mini to enrich document chunks with relevant context beyond simple text segmentation. This method enhances the accuracy of information retrieval by providing richer contextual information for each chunk, enabling more nuanced understanding and relevance ranking compared to traditional chunking methods. This approach addresses the limitations of standard retrieval methods by incorporating semantic understanding and relationships between different parts of the document.
   - **OpenAI's text-embedding-3-small**: OpenAI's text-embedding-3-small model generates vector representations of text, capturing semantic meaning. These embeddings are used for similarity comparisons and retrieval within the vector database.
   - **Pathway's Vector Store:** Pathway's Vector Store is a specialized database optimized for storing and retrieving vector embeddings, enabling efficient similarity search. It supports various indexing methods and offers a flexible API for integration with other components of the RAG system. Furthermore, its dynamic update capabilities ensure the vector store remains current as new data is ingested.
3. We adopted a **Hybrid Indexing** strategy that combines semantic relevance through vector embeddings with metadata-based filtering, ensuring efficient and accurate data retrieval while maintaining scalability for larger datasets.
4. **Query Decomposer**: Breaks down complex queries into smaller, meaningful subqueries to retrieve relevant documents from multiple perspectives that collectively address the main query.
5. **Cohere Re-ranker**: Assigns a relevance score to each document by understanding the query's intent, ensuring that the most relevant knowledge base entries are prioritized in responses.

Post-Retrieval

The post-retrieval stage is implemented using LangGraph, which enables seamless communication between agents. We extended the SELF-RAG architecture as the foundation for adding various agents, routers, and tools, making it a hybrid and more flexible Agentic-RAG system.



1. **Retrieve**: The process starts by retrieving relevant documents from the Pathway vector store, ensuring the system collects the most contextually appropriate information to address the query effectively. Additionally, it includes metadata not only about the document content but also about the node from which the information was retrieved, enhancing the system's trustworthiness, transparency, and accountability.

2. **Grade Documents**: Retrieved documents are evaluated for relevance and quality, with subqueries routed as follows: To the **Transform Query** node if the documents are irrelevant, ensuring a refined query retrieves a more relevant set of documents; To the **SQL Agent** if metadata indicates the presence of tables for more precise information extraction; To the **Finance Agent** for queries requiring specific financial data; or directly to the **Generate** node if the documents sufficiently address the query, producing the final output.

- **Transform Query Agent:** If the query is deemed not useful in its current form, it is rephrased to provide a clearer direction for the response.
- **SQL Agent**: Converts the query into an SQL statement to extract data from the database tables and transforms the results into a contextually relevant document using the LLM.
- **Finance Agent:** The financial multi-agent RAG system leverages various tools to address diverse finance-related queries. These include finding **similar companies, tracking stock performance (gainers, losers, undervalued, technology, large-cap, small-cap, penny stocks), and providing trending news, Google search trends, and stock price updates**. It also offers tools for retrieving **key metrics, and earnings data, and performing calculations**. The system dynamically selects the appropriate interdependent tools based on the query, ensuring efficient and accurate responses to queries requiring specific financial information.

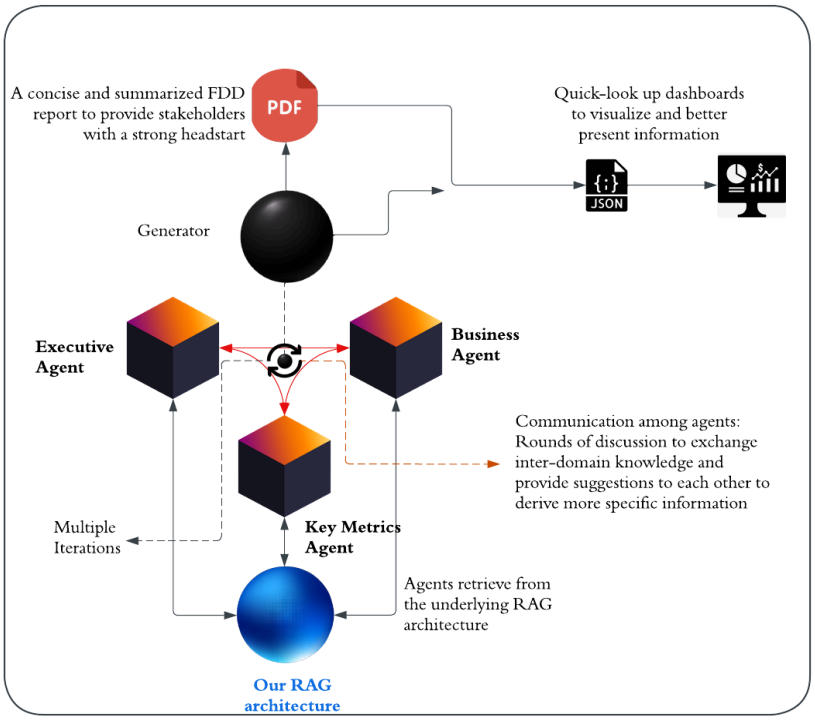
3. **Web Search**: Incorporating web search through **Tavily Search** allows the system to fetch real-time external data, addressing a broader range of queries, beyond the information available in the vector database. It is

activated when the documents retrieved from the database fail to provide a relevant answer to the query, even after multiple retrieval attempts. The results retrieved occasionally provide generic answers. This issue is resolved by the financial agent, which uses the retrieved documents and the query as context to determine the appropriate tools to invoke.

4. **Reasoning Agent**: It processes retrieved financial data to generate statistical graphs adding in interpretability and depth to the raw financial data retrieved, making it more actionable and user-friendly.
5. **Generate**: The final response for each subquery is reviewed by the **Fact Verification Agent**, ensuring coherence and grounding in the documents retrieved from multiple agents. While we implemented and tested a chain-of-verification workflow, where responses are validated by generating new queries based on the original response, answering them, and comparing the results to eliminate hallucinations—it has been excluded in the final version to minimize latency and token costs. The responses for the subqueries are then seamlessly combined into a unified output by the **Aggregator Agent**.

Vertical Autonomous Layer:

Intuition and Conceptual Overview:



The proposed system introduces a vertically scaled autonomous layer within the RAG (Retrieval-Augmented Generation) architecture to streamline Financial Due Diligence (FDD). Recognizing the time-intensive nature of FDD, this layer generates a quick and concise starting point report by leveraging agents collaboratively atop the existing RAG framework. Instead of further scaling our system horizontally, which was resulting in diminishing returns, we decided to innovate and scale the system vertically!

The agentic layer produces a concise FDD report in around 10–12 minutes, offering stakeholders a focused overview for informed decision-making. The report, along with an Overview Panel, highlights critical areas, saving significant analysis time. Our system efficiently handles the With a minimal cost of ₹15–₹20 (approximately $0.20–$0.25) per report, the system ensures both affordability and operational efficiency.

Components of the Autonomous Layer:

The autonomous layer comprises three specialized agents:

- **Key Metrics Agent:** Analyzes key financial metrics like revenue, profit margins, and growth rates, asking targeted questions to assess financial health.

- **Business Agent:** Examines broader business factors such as market conditions, competitive positioning, and risks impacting financial performance.
- **Executive Agent:** Evaluates overall performance, governance, risks, and strategic alignment to ensure financials meet industry standards and long-term objectives.

These agents are internetworked and collaborate to provide a financial assessment.

Operational Mechanism:
Each agent operates in two modes, and, the system switches between them iteratively:

- **QnA Mode:** Agents generate domain-specific questions, which the agentic RAG system answers using integrated data sources. These questions probe the company's financial and operational aspects and conditions.
- **Discussion Mode:** After the Q&A session concludes, the agents transition into Discussion Mode, where they collaboratively suggest changes, raise concerns, or propose insights. These suggestions are processed by a dedicated agent, which formulates new queries and re-injects them into the RAG system.

This iterative loop continues until the majority of agents agree that the insights are accurate, comprehensive, and require no further refinement. This process ensures the generation of high-quality, precise, and actionable financial insights. Additional details are provided in the appendix.

Autonomy and Workflow:
The autonomous layer operates intelligently, handling all decision-making from question generation to report finalization. Agents generate questions, evaluate answers, and either refine them or pass information to the generator agent which appends it into a comprehensive report. The process iterates until a high-quality final report is produced. The information extracted from the report is displayed in a **Quick Overview Panel** that facilitates a quick and interactive way of engaging with the data. Further details on the workflow and agent behaviour are discussed in the appendix.

Results and Metrics

For evaluation, we use 2 datasets:

- **FinQABench:** This dataset was created using Apple's 10K SEC filing from 2022. It has 100 test cases, each with a query and a corresponding response. The dataset is ideal for evaluating financial AI chatbots for hallucinations and accuracy. The dataset is available on HuggingFace.
- A subset of the SEC 10-Q dataset, selecting 4 AAPL documents and 39 questions designed to test the robustness and accuracy of RAG systems over queries which require multiple reasoning steps. The dataset includes single-chunk, multi-chunk, and multi-document queries. The dataset is provided in the zip file.

The following metrics from **RAGAS**[2] were used for evaluation:

1. **Context Precision:** It measures the proportion of relevant chunks in the retrieved_contexts. A higher value indicates that the retrieved documents are better.
2. **Context Recall:** It measures how many of the relevant documents were successfully retrieved. As we do not have the ground truth retrieved contexts, it uses the reference answers as a proxy. Hence, it might not be the best metric.
3. **Response Relevancy**: It focuses on assessing how pertinent the generated answer is to the given prompt. A higher score indicates a more relevant answer.

4. **Faithfulness**: It measures the factual consistency of the generated answer against the retrieved context.
5. **Factual Correctness**: It compares and evaluates the factual accuracy of the generated response with the ground truth answer.

FinQABench

We compared the performance of the following workflows on this dataset:

- Naive RAG without Agentic Contextual Chunking
- Naive RAG with Agentic Contextual Chunking

For both the workflows, we used Unstructured's Parser

Workflow	Context Precision	Context Recall	Response Relevancy	Faithfulness	Factual Correctness
Naive RAG with Agentic Chunking	**0.904**	**0.849**	**0.928**	**0.901**	**0.518**
Naive RAG without Agentic Chunking	0.801	0.782	0.883	0.828	0.449

It is clear from the results that the performance is enhanced by Agentic Chunking. Both the models performed well on this dataset as the questions were simple and did not involve reasoning over multiple chunks or documents.

SEC-10Q Dataset

Here, we compared the following workflows:

- OpenParser with Self-RAG[3]
- OpenParser with Naive-RAG
- Our Post-Retrieval Agentic Workflow with Contextual Chunking
- Unstructured Parser with Naive-RAG
- Unstructured Parser with Self-RAG

It can be observed that Our Post-Retrieval Agentic Workflow with Contextual Chunking outperforms other methods by a significant margin. This shows the efficiency and enhanced performance of our Post-Retrieval Agentic Workflow on datasets which require reasoning over multiple documents in order to answer queries. In particular, improved **Factual Correctness** and **Faithfulness** are much needed in Financial Due Diligence, as it indicates that hallucination is minimized and information is preserved.

Workflow	Context Precision	Context Recall	Response Relevancy	Faithfulness	Factual Correctness
OpenParser with Self-RAG	0.524	0.272	0.364	0.659	0.280
OpenParser with Naive-RAG	0.521	0.253	0.374	0.623	0.273
Unstructured Parser with Self-RAG	0.499	0.259	0.269	0.596	0.269
Unstructured Parser with Naive-RAG	0.524	0.313	0.319	0.503	0.276
Our Workflow	**0.690**	**0.396**	**0.701**	**0.788**	**0.384**

Challenges Faced and Solutions

- **Promoting Transparency**: To enhance transparency, we implemented socket-based communication between the frontend and backend servers. Our architecture integrates multiple agents and tools, enabling precise tracing of each query's path to enhance explainability. During query processing, the backend emits socket messages as the query traverses agents or tools, which the frontend captures and visualizes as a sequence of tags. Each subquery's completion is marked with an "end" emit, displayed as a green tick on the frontend, indicating successful processing. This approach improves model transparency and reliability, crucial for Financial Due Diligence (FDD) in high-stakes corporate settings.
- **Speeding up processes and parallelization:** We tackled numerous processes asynchronously using libraries such as `asyncio`, which introduced the typical challenges associated with asynchronous programming, including maintaining a consistent state, error handling and debugging. In our architecture, all subqueries of a single query are processed in parallel through our agentic flow. This design ensures that, regardless of complexity, any query is completed within 3 minutes.
- **Prioritized Stability:** In some cases, functionality was prioritized over parallelization. For instance, during agentic chunking, initial asynchronous calls caused crashes due to OpenAI's token limits. To ensure stability within constrained resources, these calls were switched to synchronous, adding some time overhead but maintaining reliability. The codebase remains scalable and can be adapted for asynchronous operations when resources allow.
- **Integrating and building the vertical autonomous layer**: The agents generate 15 queries each, decomposed into ~40 sub-queries, which are processed parallely by the RAG. The results are re-verified by the agents for correctness and further refinement. Once the majority of agents are satisfied, a comprehensive report is sent to the **Generator Agent** for final compilation. To ensure accuracy, vague or unreliable answers are rejected, avoiding hallucination and delivering an error-free report. Despite the complexity of multi-agent coordination and randomness, a simple and elegant solution was implemented.

Resilience to Error Handling:

Our Financial Agentic RAG system incorporates robust error management, ensuring uninterrupted functionality. This enhances resilience by enabling quick recovery from failures, rapid error identification, and timely debugging and resolution.

Tool-Specific Fallbacks: Key tools have designated backups, such as web search stepping in for API failures or alternative tools in the Finance Agent.
For example: When the relevant data is not retrieved from the primary data source, Finance Agent falls back to tools like Tavily and if that too fails, it falls back to *open source web searches* like duck duck go, further fallback details have been mentioned in the appendix.
Error Handling: Nested *try* and *except* blocks to manage failures effectively.
Fallback Mechanisms, extensive logging is implemented to ensure traceability. When a fallback is triggered, the callback function raises a warning to facilitate tracking and debugging.
Avoiding Exponential Back-Off: Excluded exponential back-off to maintain query speed, as most tool failures were observed to be binary.

User Interface

The application offers two core modes that users can effortlessly switch between **Chat Mode** and **Report Generation Mode**.

1. Chat Mode:
In this mode, users can ask questions related to a company's financials, and the system generates responses using the underlying architecture built on the **Pathway** infrastructure. The key feature here is the transparency provided by the system—users not only receive answers but also get a visual representation of the exact solution structure used to derive the response. This allows users to clearly understand how their query is processed and addressed, making the system's approach more intuitive and insightful.

2. Report Generation Mode:
This mode enables the generation of light and concise FDD reports using the **Vertical Autonomous Layer**. By entering a company's name, collaborative agents powered by an **Agentic RAG** architecture collect, process, and analyze financial data. The **Next.js** and **TypeScript**-based user interface provides real-time visibility into each agent's actions and contributions, ensuring transparency in the report generation process. The final report is saved locally in the working directory, and a dashboard page is generated to present Revenue Shares, Global Market Penetration among other key metrics. This **Dashboard** features an immersive and interactive frontend interface, providing a comprehensive visualization of the company's performance and insights.

Technology Highlights:
This dual-mode functionality provides a powerful tool for both quick queries and detailed financial analysis, tailored to meet the needs of users looking for flexibility and depth in financial due diligence.

Responsible AI Practices

Our AI architecture incorporates robust guardrails to ensure secure, ethical, and compliant interactions. These safeguards block irrelevant queries, including defamation, privacy violations, hate speech, and intellectual property concerns, upholding privacy and ethical standards.

Llamaguard's Safety guardrail:[4]

Llamaguard, built on the Llama-3.1 8b model, classifies input content as "safe" or "unsafe." Unsafe queries are further categorized into 14 predefined types, enabling risk mitigation. This ensures secure, compliance-driven user interactions with the system.

PII Guardrail[5]**:**

We explored using Guardrail.ai with Presidio_Analyzer and Presidio_Anonymizer for PII detection and anonymization. However, we found that it added unnecessary complexity and overhead to our workflow without providing sufficient value, leading us to discontinue this approach.

Transparency through Socket Communication:
To ensure clarity in the query processing flow, we integrated socket communication between the frontend and backend to ensure clarity in query processing. With multiple agents and tools involved, the backend sends real-time socket messages to track each query's journey. The frontend captures these messages and displays them as a sequence of tags, offering users a transparent, visual representation of the query's progress through the system.

Lessons Learned

Prioritize Simplicity: Usability increases with intuitive design and modular architecture, which led us to create a streamlined interface for seamless navigation and a modular backend that simplifies feature integration and maintenance.
Challenges Foster Creativity: Limited resources and the need to balance latency with accuracy pushed us to innovate. For instance, Hybrid Indexing and the Agentic Chunker emerged as solutions to enhance data retrieval efficiency.
Resilience Through Error Handling: Implementing fallback strategies, error handling, and resilience mechanisms in RAG helped mitigate the impact of agent failures and ensured uninterrupted functionality.
User Feedback is Vital: Early testing with financial reports revealed gaps in our design, allowing us to tailor features like the **Quick Overview Panel** and **Vertical Autonomous Layer** to better meet investors' needs.

Conclusion

In conclusion, this project demonstrates the transformative potential of an agentic RAG system applied to the complex domain of financial due diligence. FA^{3}STER, our proposed solution, leverages Pathway's dynamic capabilities to autonomously analyze financial documents, respond accurately to complex queries, and generate concise, actionable reports and dashboards. Through innovative architectural design, including agentic contextual chunking, a sophisticated post-retrieval agent framework, and a novel vertical autonomous layer, FA^{3}STER significantly streamlines the FDD process, enhancing both efficiency and accuracy. Rigorous evaluation using established metrics and datasets demonstrates the system's superior performance compared to existing RAG implementations. While challenges in transparency, parallelization, and multi-agent coordination were addressed effectively, future work could explore further enhancements in explainability, scalability, and the integration of real-time market data through Pathway's ETL and streaming frameworks for even more comprehensive financial analysis. This project contributes a valuable framework for developing and deploying agentic RAG systems in demanding real-world applications, paving the way for more intelligent and automated solutions in the financial domain.

----------END----------