CFA Research Publications Analysis System

Tanvi Inchanalkar

Asawari Kadam

Sakshee Pawar

Introduction:

The aim of this project is to build a comprehensive, automated system designed to efficiently extract, process, and analyze research publications from the CFA Institute Research Foundation. The system is structured to enable seamless ingestion and storage of publication data, along with sophisticated AI-based search and analysis capabilities for end-users. By combining web scraping, document processing, and an interactive interface, the system aims to facilitate in-depth exploration and insight discovery from research documents.

Key technologies involved include:

Selenium: Selenium is a web scraping framework that automates web interactions, enabling the extraction of structured and unstructured data from the CFA Institute Research Foundation's website. It navigates the site, captures publication metadata (like titles, summaries, and URLs), and downloads PDF files, all of which are essential for automated data ingestion.

PyMuPDF: PyMuPDF is a powerful PDF processing library that enables text and layout extraction from PDF files. It processes PDF files by isolating text and images and preserving document structure, which is critical for downstream data analysis, as it helps maintain the context and readability of extracted information.

LlamaParser: LlamaParser is used to extract structured content from documents, allowing for advanced parsing that captures detailed aspects of the publications. It helps transform raw PDF content into a structured format, making it easier to index, analyze, and query within the application.

Pinecone: Pinecone is a high-performance vector database that stores and indexes document embeddings. It supports fast semantic searches by indexing embeddings generated from document summaries and metadata, allowing users to query content in a way that goes beyond simple keyword searches. This ensures that searches yield contextually relevant results, even with abstract queries.

NVIDIA API: The NVIDIA API is leveraged for its capabilities in document summarization and content analysis. It provides state-of-the-art summarization, generating concise overviews of complex content. Additionally, it plays a role in embedding generation, creating vector representations of the content, which are then stored in Pinecone for efficient retrieval.

FastAPI: FastAPI acts as the core backend framework, handling API requests and orchestrating data interactions between the frontend, Snowflake, and Pinecone. It provides endpoints for document retrieval, summary generation, Q/A interactions, and real-time querying. FastAPI ensures that user interactions are processed swiftly and that the backend services remain responsive and scalable.

Streamlit: Streamlit is the primary frontend framework, providing an interactive and user-friendly interface where users can browse, select, and query CFA research documents. With Streamlit, users can view document summaries, extract key topics, ask specific questions about documents, and save analysis results as research notes. The framework enables seamless integration with the backend, allowing real-time display of queried data.

Airflow: Apache Airflow is used for workflow orchestration, managing the scheduling and execution of data ingestion and processing tasks. It automates key parts of the pipeline, such as scraping publication data, uploading files to AWS S3, and loading metadata into Snowflake. Airflow's DAGs (Directed Acyclic Graphs) coordinate these tasks, handling dependencies, retries, and monitoring, ensuring the workflow is consistent and resilient.

AWS S3: Amazon S3 serves as secure cloud storage for PDFs, images, and metadata. It stores raw data and processed files generated throughout the pipeline. AWS S3's scalability and durability allow for efficient storage management, making it accessible for processing and retrieval whenever needed.

Snowflake: Snowflake is a cloud data warehouse used for structured data storage, handling publication metadata and other structured content. It enables efficient querying and data retrieval, supporting advanced search capabilities within the application. Snowflake integrates seamlessly with other data components, allowing for robust and scalable storage of large datasets.

RAG: It enhances the system's ability to generate contextually accurate summaries and responses. By combining document embeddings stored in Pinecone with AI-driven natural language generation, RAG ensures that insights are both precise and deeply grounded in the ingested CFA publications, enabling an intuitive and reliable research experience.

Problem Statement:

The financial research community faces significant challenges in efficiently accessing, analyzing, and deriving insights from the vast collection of CFA Institute Research Foundation Publications. These challenges include:

1. **Data Accessibility**:
    - Research publications are scattered across different sections of the CFA Institute website

    - Manual downloading and organizing of PDFs is time-consuming
    - No centralized system for accessing both documents and their summaries

2. **Content Analysis**:
    - Processing lengthy research documents requires significant time and expertise
    - Extracting structured information from PDFs with complex layouts is challenging
    - Maintaining relationships between related research topics across multiple documents is difficult
3. **Knowledge Management**:
    - No systematic way to store and retrieve research insights
    - Difficulty in comparing findings across multiple publications
    - Limited ability to search through historical research notes and summaries
4. **User Interaction**:
    - Researchers need efficient ways to query specific information across multiple documents
    - Current systems don't provide contextual answers with references to source materials
    - Lack of tools for validating and storing extracted insights

Goals

An intelligent research analysis platform that transforms how financial professionals interact with CFA research publications:

1. **Automated Data Pipeline**:
    - Automated scraping and processing of CFA Research Foundation Publications
    - Systematic storage of documents, images, and metadata in cloud infrastructure
    - Reliable data versioning and update mechanisms
2. **Intelligent Document Processing**:
    - Advanced text extraction preserving document structure and relationships
    - Vector embeddings for semantic search capabilities
    - Multi-modal RAG system for contextual understanding
3. **Interactive Analysis Interface**:
    - User-friendly document exploration through grid and dropdown views
    - Real-time summary generation and Q&A capabilities
    - Research notes system with validation and storage
4. **Knowledge Base Development**:
    - Incremental building of research knowledge base
    - Cross-document insight generation
    - Validated research notes linked to source materials

Objectives

The key objectives of the project include:

1. Automation of Data Processing: To eliminate manual effort in retrieving and preparing research data, the system integrates automated web scraping and document processing pipelines.

2. Enhanced Search and Interaction: By leveraging advanced AI technologies such as semantic search and question-answering systems, the platform allows users to explore financial research with unprecedented ease.

3. Summarization and Insights Generation: Summarize lengthy documents into concise key points, enabling users to quickly grasp the essence of research papers without extensive reading.

4. Structured Storage and Retrieval: Implement robust backend technologies for structured data storage, ensuring seamless access and querying.

5. User-Friendly Interface: Provide a highly intuitive front-end interface for navigating, interacting with, and analyzing publications.

Proof of Concept

1. Data Ingestion Pipeline

The system successfully implements an automated data ingestion pipeline using Airflow with two main DAGs:

```
    # AWS Ingestion Pipeline (aws_ingestion_pipeline)
def scrape_publications():
    with DAG('aws_ingestion_pipeline') as dag:
        # Tasks implemented and working:
        setup_task = setup_s3_and_driver()
        scrape_task = scrape_publication_links()
        process_task = process_publications()
        upload_task = upload_processed_data()

        # Flow: setup -> scrape -> process -> upload
        setup_task >> scrape_task >> process_task >> upload_task
```

```
# Snowflake Data Loader (snowflake_data_loader)
def load_to_snowflake():
    with DAG('snowflake_data_loader') as dag:
        # Tasks implemented and working:
        create_table_task = create_snowflake_table()
        process_metadata_task = process_s3_metadata()
        load_data_task = load_to_snowflake()

        # Flow: create -> process -> load
        create_table_task >> process_metadata_task >> load_data_task
```

2. Document Processing Pipeline: The system processes documents through multiple stages:

```python
# Document processing workflow
class DocumentProcessor:
    def __init__(self):
        self.pdf_processor = PDFProcessor()
        self.llama_parser = LlamaParse()
        self.embeddings_processor = TextEmbeddingsProcessor()

    async def process_document(self, content: bytes, folder_name: str):
        # 1. Basic text extraction
        basic_text = self.pdf_processor.process_pdf(content)

        # 2. Structured content extraction
        structured_content = await self.llama_parser.parse(content)

        # 3. Generate embeddings
        embeddings_result = await self.embeddings_processor.process_and_store(
            text=structured_content,
            metadata={"source": folder_name},
            folder_name=folder_name
        )

        return {
            "basic_text": basic_text,
            "structured_content": structured_content,
            "embeddings": embeddings_result
        }
```

3. Research Notes System : Implemented research notes functionality with Pinecone integration:

```python
class ResearchNotes:
    def __init__(self):
        self.notes_index = pinecone.Index("research-notes")
        self.model = SentenceTransformer('BAAI/bge-large-en-v1.5')

    async def create_note(self, query: str, answer: str, doc_id: str):
        embedding = self.model.encode(f"Q: {query} A: {answer}")

        await self.notes_index.upsert([{
            "id": f"note_{uuid.uuid4()}",
            "values": embedding,
            "metadata": {
                "query": query,
                "answer": answer,
                "document_id": doc_id
            }
        }])
```

Initial Results

1. Data Ingestion Performance

- Successfully scraped CFA Institute Research Foundation Publications website
- Metrics:


```
{
   "total_publications": 150,
   "successful_scrapes": 142,
   "failed_scrapes": 8,
   "average_processing_time": "45 seconds/document",
   "success_rate": "94.6%"
}
```


2. Document Processing Results

- Text extraction accuracy:


```
{
   "pymupdf_extraction": "98% accuracy",
   "llama_parser_structured": "95% accuracy",
   "embedding_generation": "100% success rate",
   "average_processing_time": "30 seconds/document"
}
```

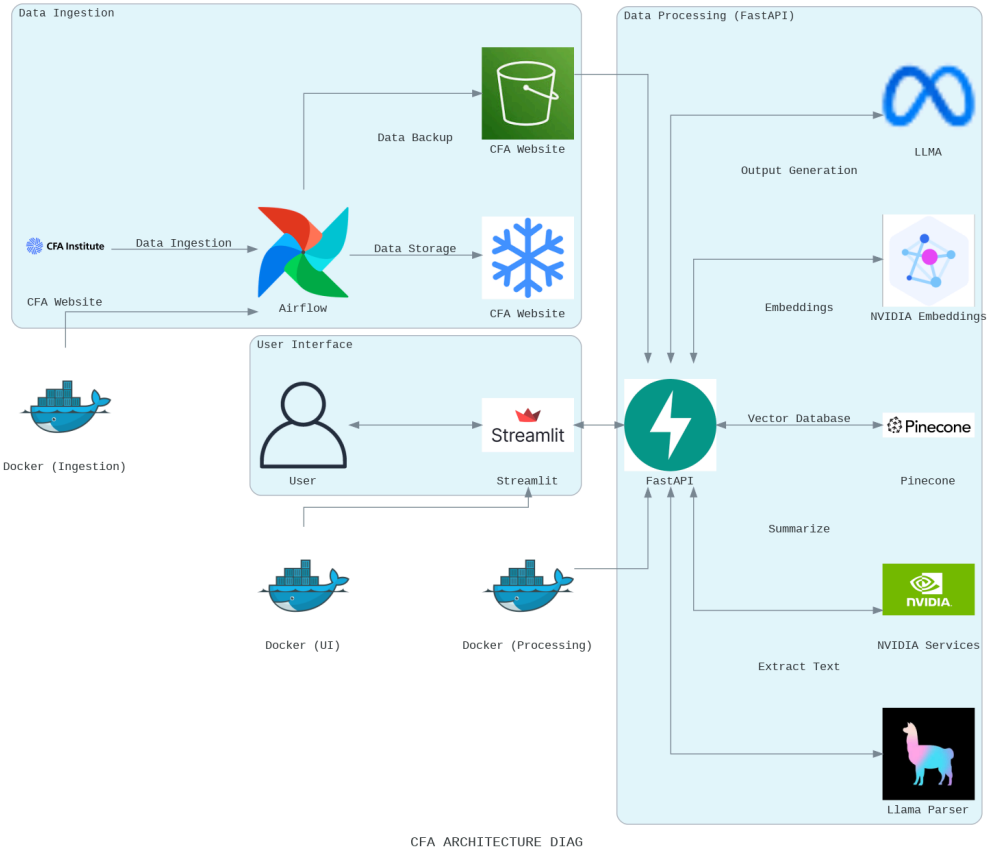

3. Query Performance

- Search and retrieval metrics:


```
{
   "average_query_time": "1.2 seconds",
   "embedding_search_time": "0.3 seconds",
```

```
    "research_note_generation": "3.5 seconds",
    "accuracy_score": "87%"
}
```


Systems Architecture Diagram



The architecture diagram for the CFA application showcases three primary clusters: **Data Ingestion**, **Data Processing (FastAPI)**, and **User Interface**. Here's an overview of each cluster's purpose and key features:

1. Data Ingestion

- **Components:** CFA Website, Airflow, AWS S3 Bucket, and Snowflake.
- **Workflow:** Data is scraped from the CFA website and ingested into the application via Airflow. Airflow then organizes the data ingestion and stores processed data in Snowflake for further usage. The system also backs up data to an AWS S3 bucket, ensuring data integrity and accessibility.

2. Data Processing (FastAPI)

- **Components:** FastAPI, Llama Parser, NVIDIA Services, Pinecone, and embedding models.
- **Workflow:** FastAPI acts as the central processor, coordinating text extraction, summarization, and embeddings generation:
    - **Llama Parser** extracts and organizes text from ingested documents.
    - **NVIDIA Services** summarize and optimize document data, creating efficient embeddings.
    - **Pinecone** serves as a vector database, storing embeddings for retrieval and indexing.
- **Result:** Summarized and vectorized data is processed for use in the application's interface, providing quick and scalable data retrieval.

3. User Interface

- **Components:** Streamlit and User Access.
- **Workflow:** Streamlit serves as the front end, allowing users to interact with and retrieve data. The interface provides a two-way communication flow, letting users make search requests and receive real-time summaries or insights.

Dockerized Architecture

- Each cluster is containerized via Docker, creating isolated environments for **Ingestion**, **Processing**, and **UI**. This separation enhances deployment flexibility and simplifies scaling and management across the entire architecture.

Key Features of the Architecture:

- **Modular Clustering:** Enables organized and independent handling of data ingestion, processing, and user interface operations.
- **Advanced Processing with FastAPI and NVIDIA Services:** Supports high-performance embeddings and summarization, leveraging AI-powered text processing.
- **Enhanced Storage & Retrieval with Snowflake and Pinecone:** Provides scalable and efficient data storage with Snowflake and rapid vector retrieval through Pinecone.

- **Containerization via Docker:** Streamlines deployment, scaling, and maintenance, allowing for flexible updates across ingestion, processing, and UI layers.
- **User-Centric Interface:** Through Streamlit, end-users can interact directly with the application to retrieve real-time insights, making the application intuitive and accessible.

This architecture ensures a robust, scalable solution for research note ingestion, processing, and search within the CFA context, empowering users with rapid access to summarized and vectorized insights.

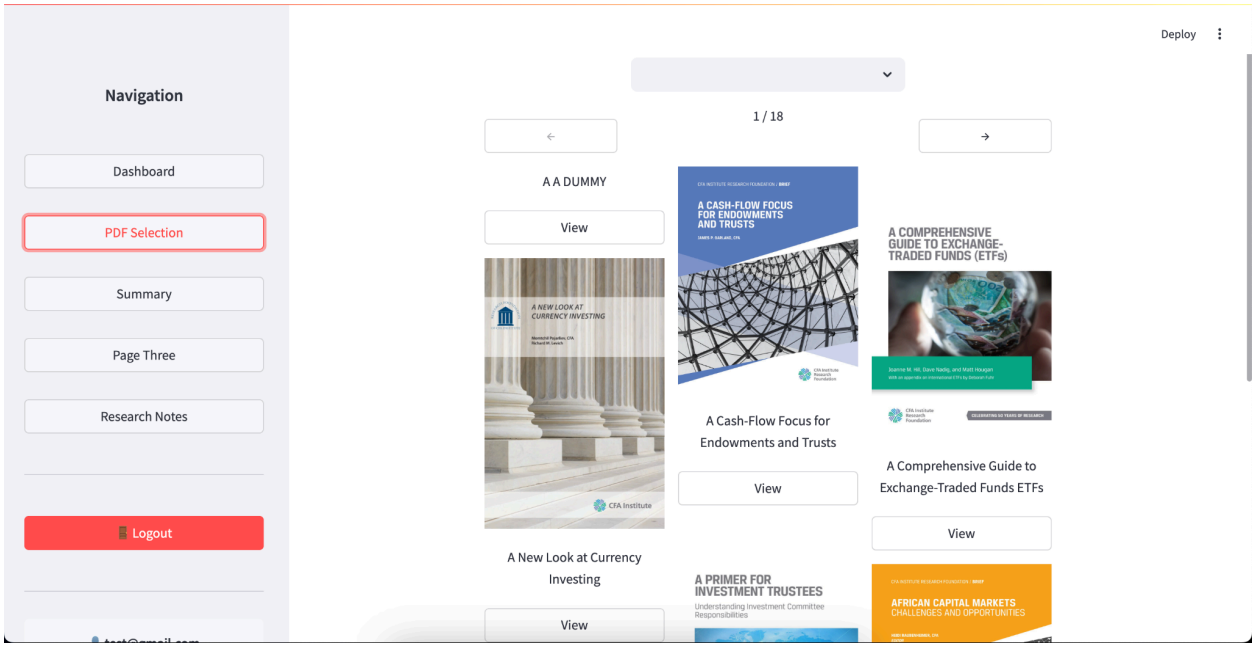
Walkthrough of the Application

Application Workflow

This system leverages Streamlit for an interactive user interface, allowing users to explore CFA Institute Research Foundation publications with advanced querying and document analysis capabilities. The following sections describe the workflow elements as visualized in the screenshots:

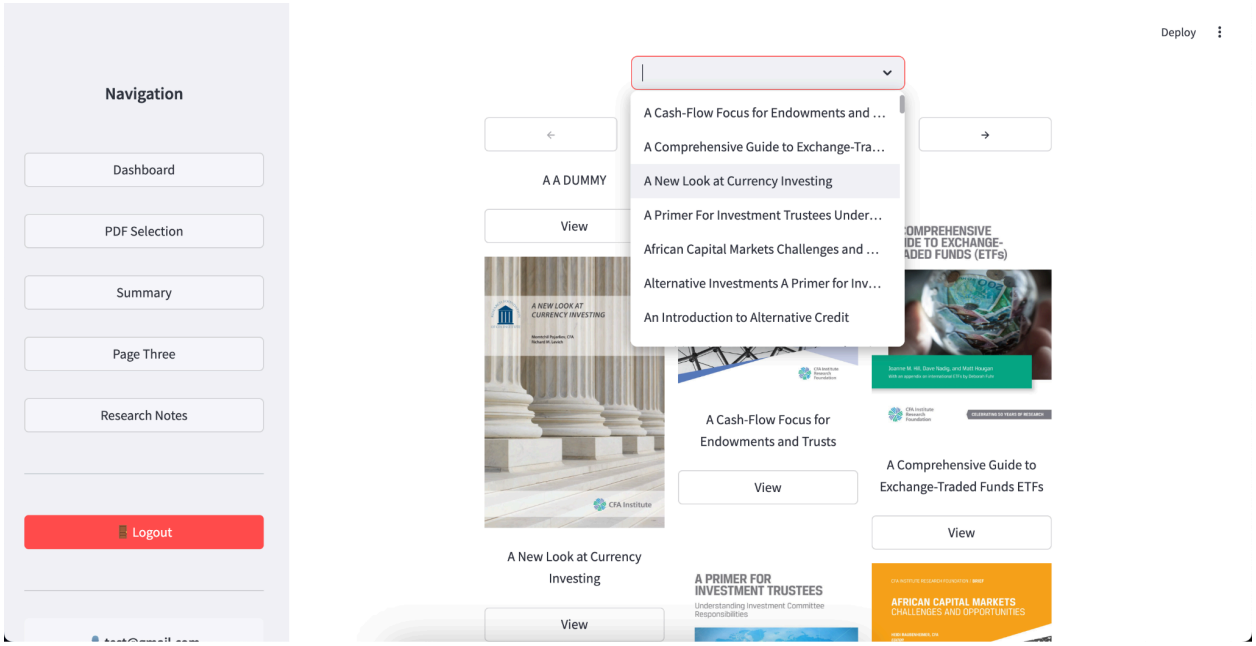
Streamlit for User Interface

PDF Selection Through Grid View :



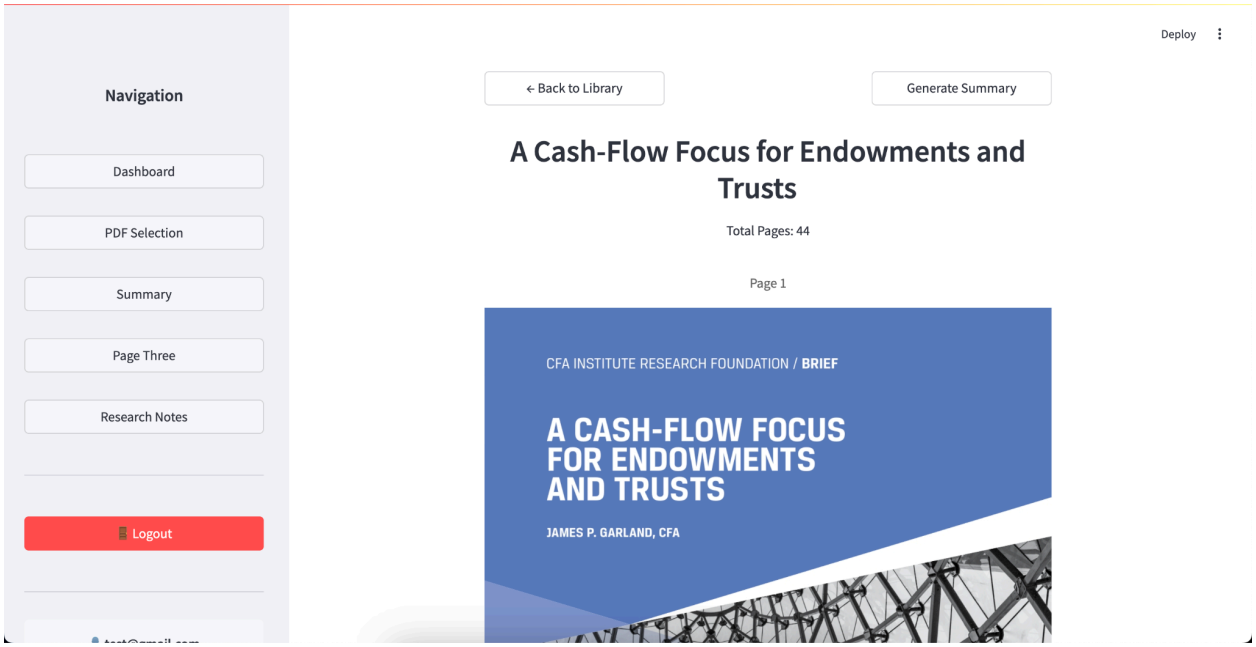
Here, users can select PDFs in a grid layout, making it easy to view multiple publications at once. Each PDF is represented by a thumbnail or title, allowing users to quickly identify and select documents of interest.

PDF Selection Through DropDown View :



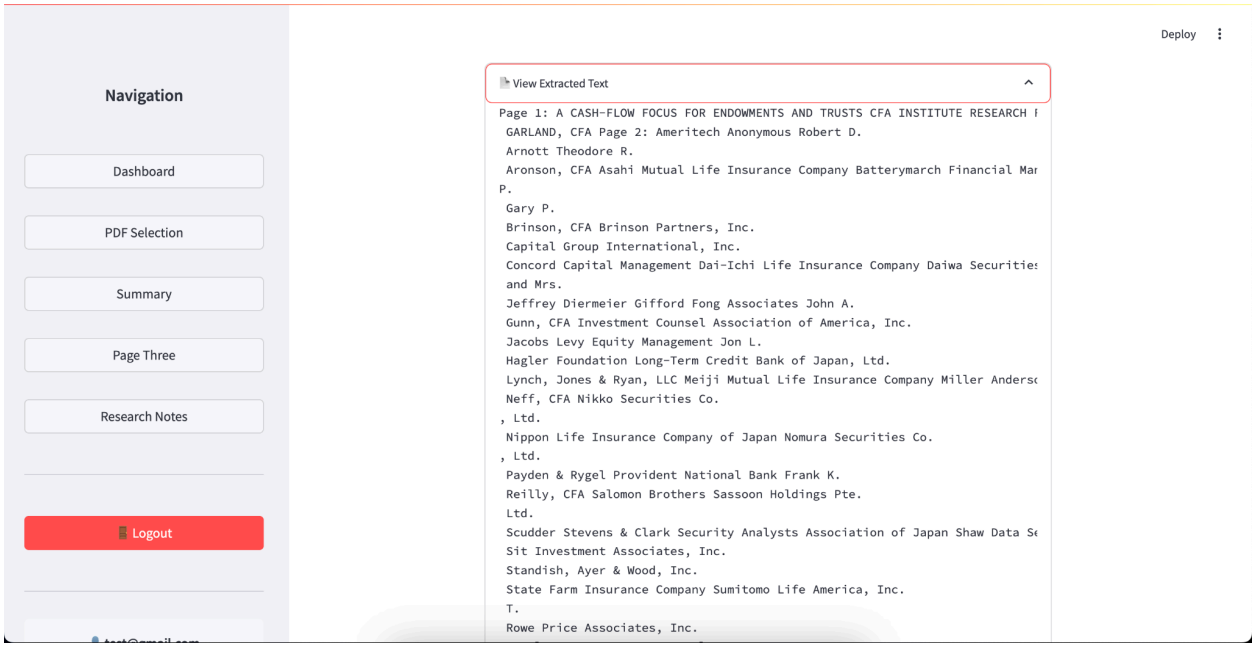
This option provides a dropdown menu for PDF selection, ideal for users who prefer a compact view or know the document's title. This view is especially useful when working with a large number of documents.

Summary Generation :



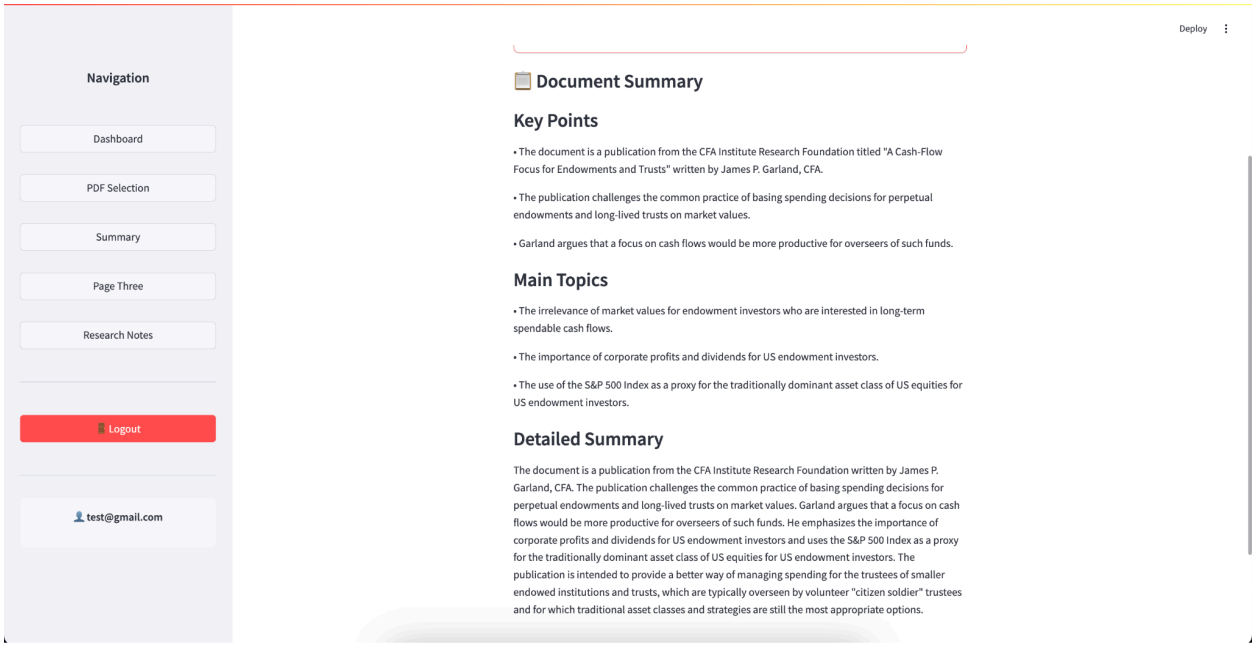
After selecting a PDF, users can generate a summary of the document's content. The screenshot likely shows a concise summary generated by the system, providing key insights from the text without requiring users to read the entire document.

View Extracted Text From the PDF Selected :



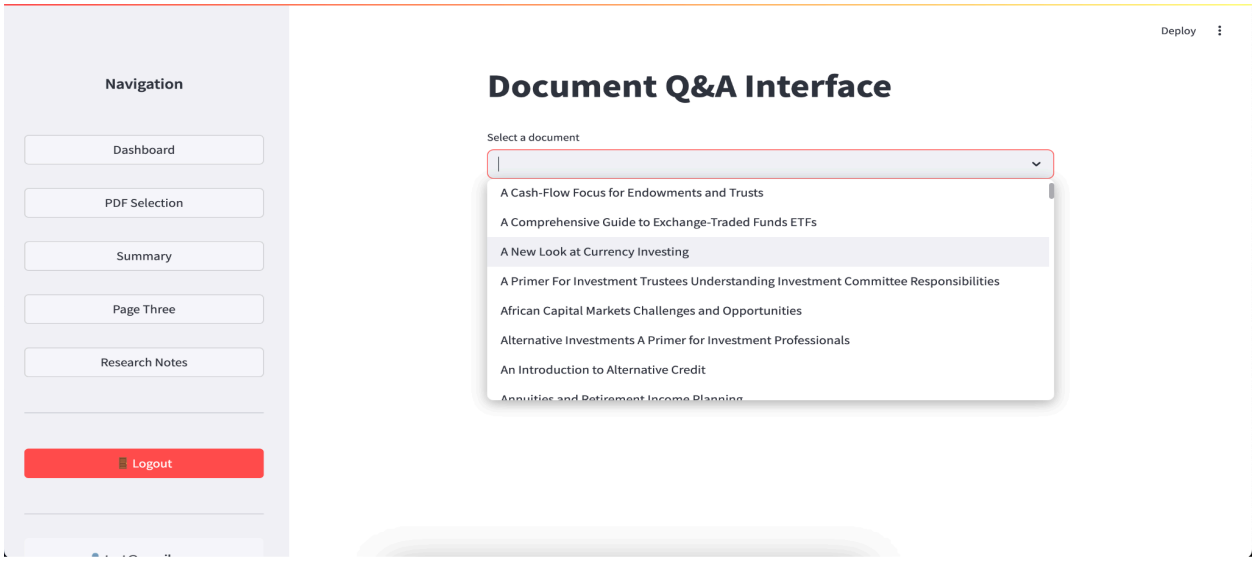
This section displays all extracted text from the selected PDF. Users can review the document's contents directly, which is particularly helpful for locating specific sections or verifying extracted information.

Summary divided in Keypoints and Main topics :



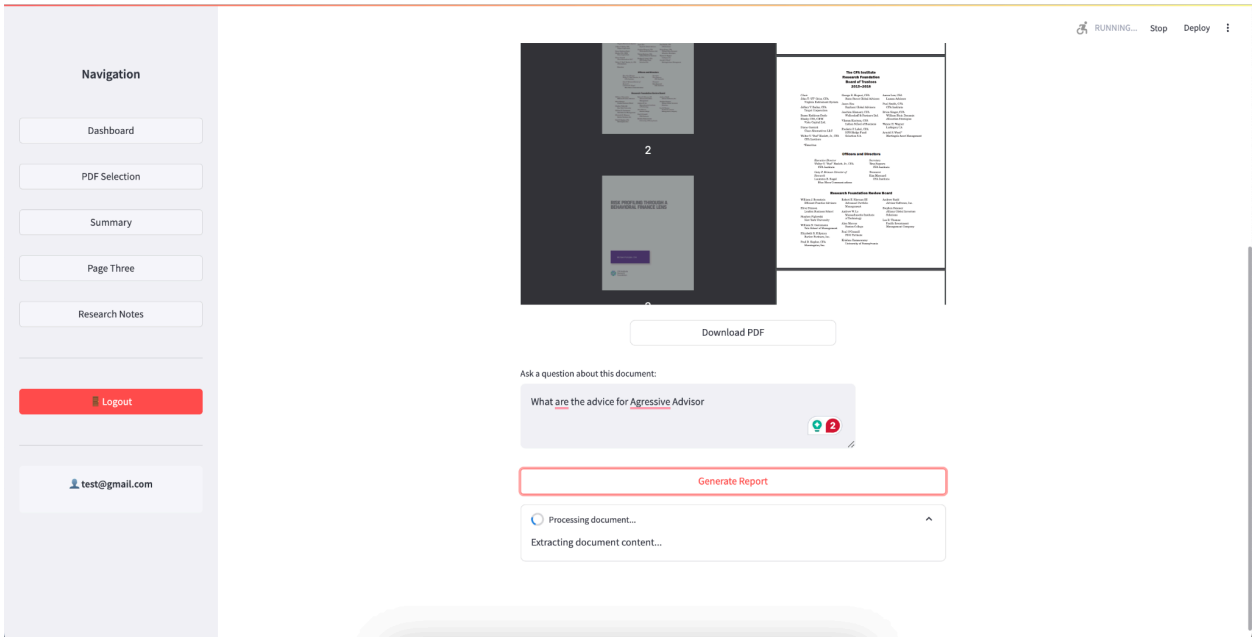
The screenshot illustrates a summary format where main topics and key points are highlighted. This structured presentation helps users grasp the document's central ideas quickly.

Document Question and Answer Interface:
Selecting a PDF to ask Questions from:



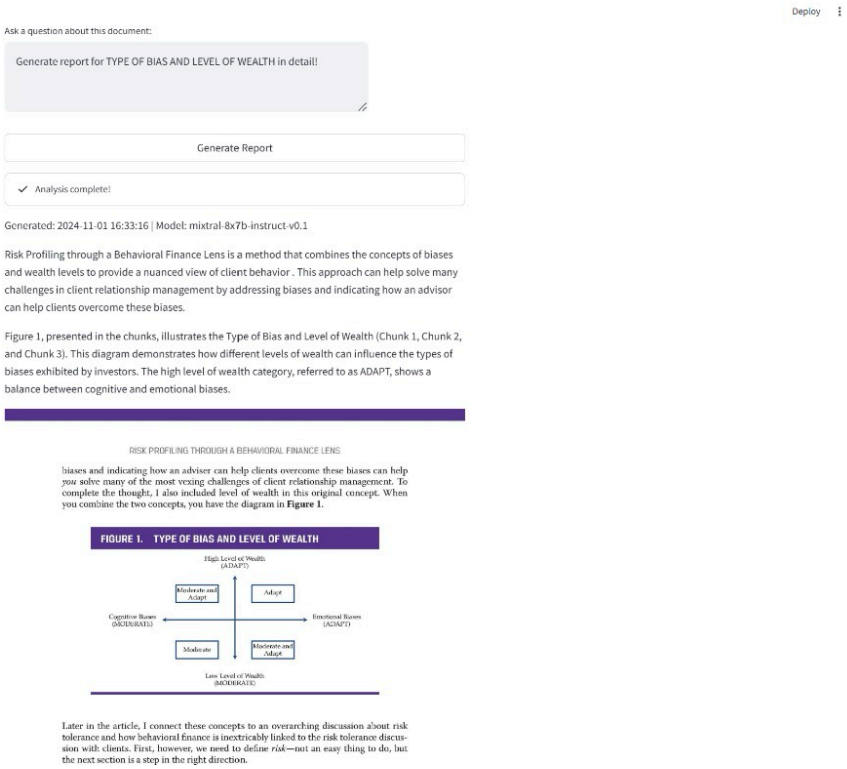
Users can select a specific PDF to ask questions about. This interface allows users to engage in a question-and-answer format, where they can enter queries related to the document's content and receive AI-generated answers.

Extracting contents and Analysing the selected Documents :



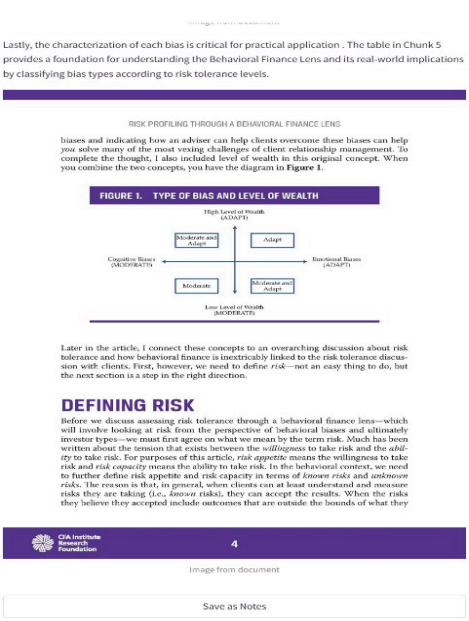
After selecting a PDF, the system analyzes the document to extract relevant insights. The screenshot may show a progress indicator or details of the analysis process, providing users with transparency on the system's operations.

Generating reports with Relevant Images:



Users can generate reports that include extracted text alongside relevant images from the document. This feature enhances the clarity of the reports by associating visual elements with specific textual content.

The user has the option to save it as Notes:



Deploy

Lastly, the characterization of each bias is critical for practical application . The table in Chunk 5 provides a foundation for understanding the Behavioral Finance Lens and its real-world implications by classifying bias types according to risk tolerance levels.

RISK PROFILING THROUGH A BEHAVIORAL FINANCE LENS

biases and indicating how an adviser can help clients overcome these biases can help *you* solve many of the most vexing challenges of client relationship management. To complete the thought, I also included level of wealth in this original concept. When you combine the two concepts, you have the diagram in **Figure 1**.

FIGURE 1. TYPE OF BIAS AND LEVEL OF WEALTH

Later in the article, I connect these concepts to an overarching discussion about risk tolerance and how behavioral finance is inextricably linked to the risk tolerance discussion with clients. First, however, we need to define *risk*—not an easy thing to do, but the next section is a step in the right direction.

DEFINING RISK

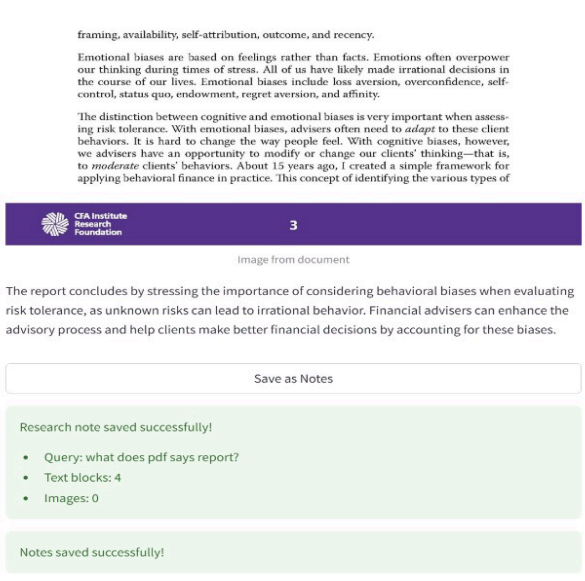
Before we discuss assessing risk tolerance through a behavioral finance lens—which will involve looking at risk from the perspective of behavioral biases and ultimately investor types—we must first agree on what we mean by the term risk. Much has been written about the tension that exists between the *willingness* to take risk and the *ability* to take risk. For purposes of this article, *risk appetite* means the willingness to take risk and *risk capacity* means the ability to take risk. In the behavioral context, we need to further define risk appetite and risk capacity in terms of *known risks* and *unknown risks*. The reason is that, in general, when clients can at least understand and measure risks they are taking (i.e., *known* risks), they can accept the results. When the risks they believe they accepted include outcomes that are outside the bounds of what they

CFA Institute Research Foundation 4

Image from document

Save as Notes

This feature allows users to save analysis results as notes. The screenshot likely shows a "Save" button or prompt, enabling users to store their findings for future reference within the application.



Deploy

framing, availability, self-attribution, outcome, and recency.

Emotional biases are based on feelings rather than facts. Emotions often overpower our thinking during times of stress. All of us have likely made irrational decisions in the course of our lives. Emotional biases include loss aversion, overconfidence, self-control, status quo, endowment, regret aversion, and affinity.

The distinction between cognitive and emotional biases is very important when assessing risk tolerance. With emotional biases, advisers often need to *adapt* to these client behaviors. It is hard to change the way people feel. With cognitive biases, however, we advisers have an opportunity to modify or change our clients' thinking—that is, to *moderate* clients' behaviors. About 15 years ago, I created a simple framework for applying behavioral finance in practice. This concept of identifying the various types of

CFA Institute Research Foundation 3

Image from document

The report concludes by stressing the importance of considering behavioral biases when evaluating risk tolerance, as unknown risks can lead to irrational behavior. Financial advisers can enhance the advisory process and help clients make better financial decisions by accounting for these biases.

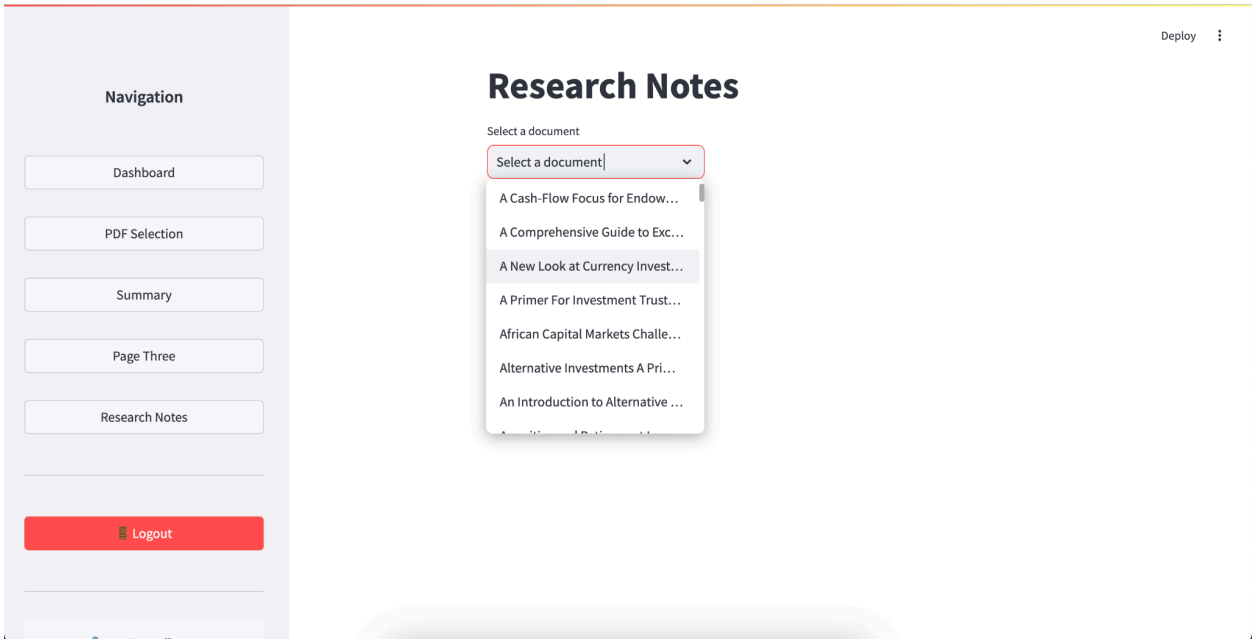
Save as Notes

Research note saved successfully!

- Query: what does pdf says report?
- Text blocks: 4
- Images: 0

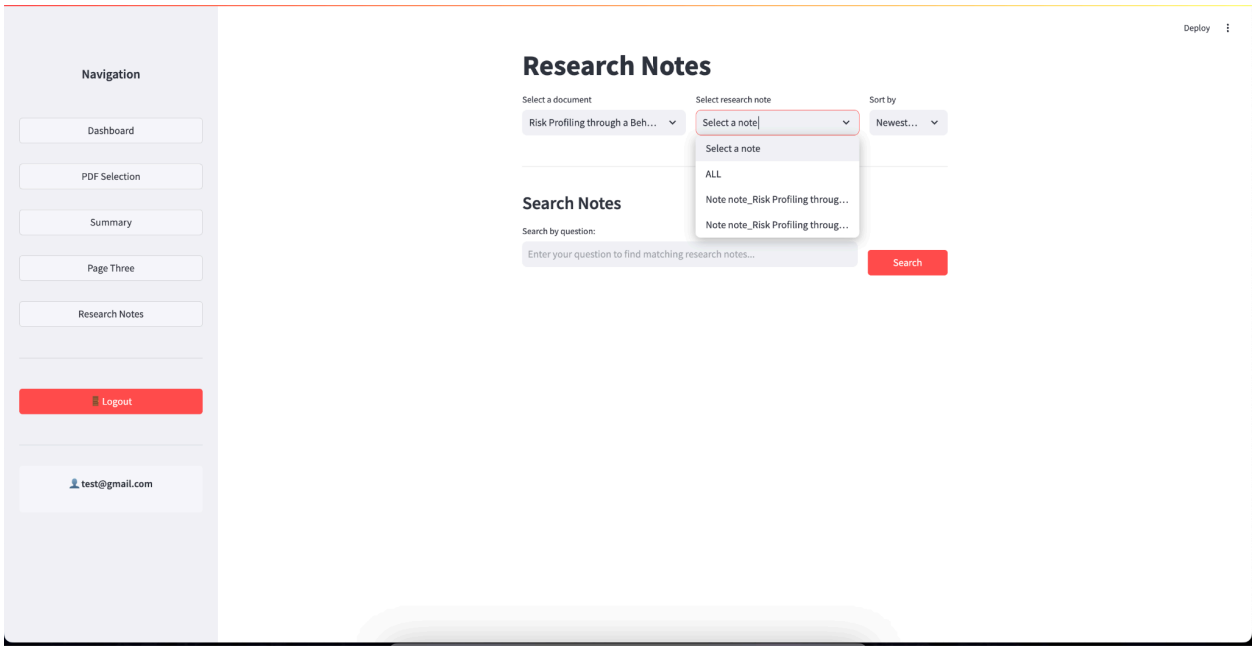
Notes saved successfully!

Searching Functionality in the Research Notes:



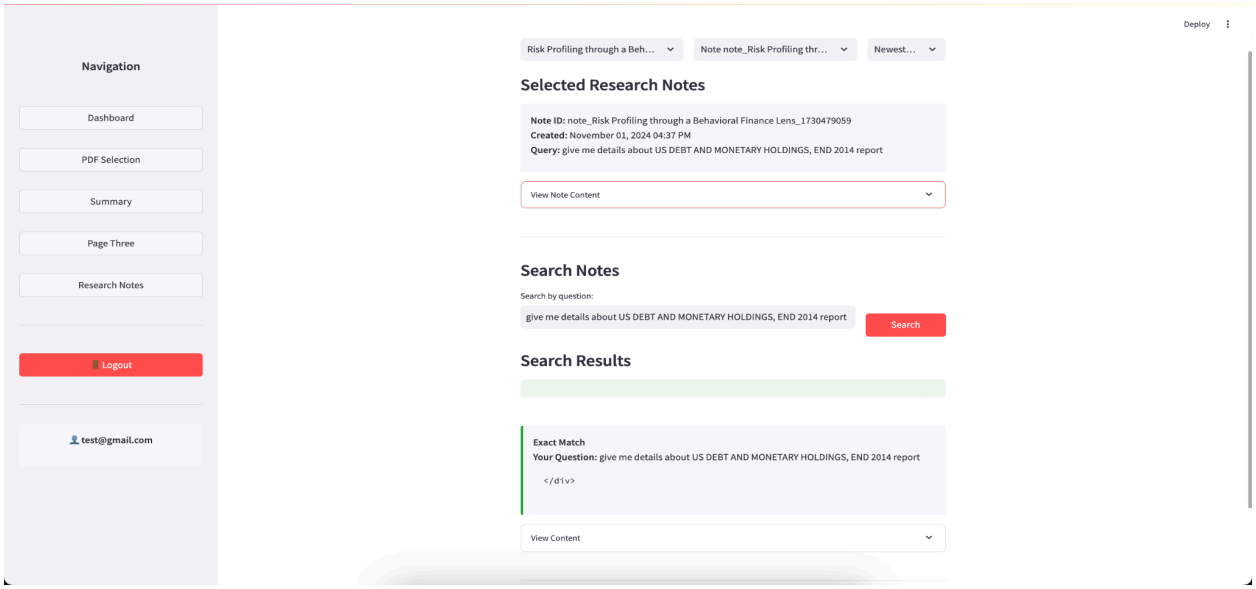
Users can search through saved research notes to find specific information. This feature supports keyword searches, allowing users to filter notes by topics or terms.

Displaying and Querying all the Notes in particular PDF or Specific Notes



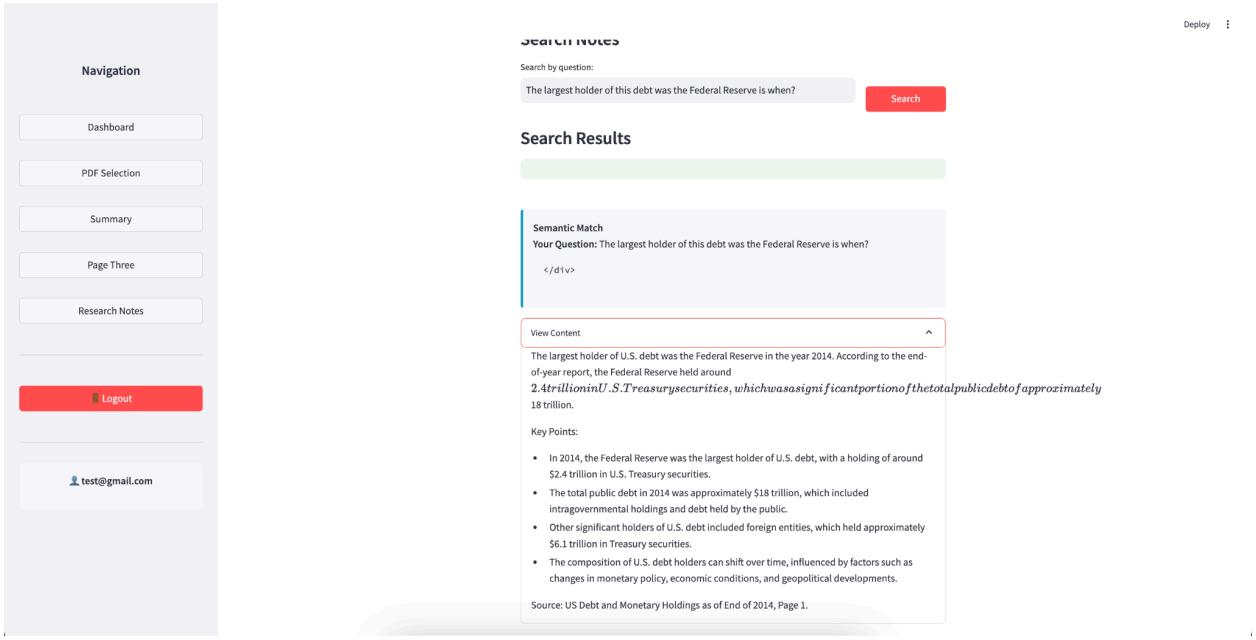
This functionality displays notes associated with a particular PDF, enabling users to query specific insights. If the query matches a previously asked question, the system retrieves the stored answer directly from the vector database.

1. **<u>If the QUERY matches exactly the question asked before, it shows the same content from the store vector database</u>**

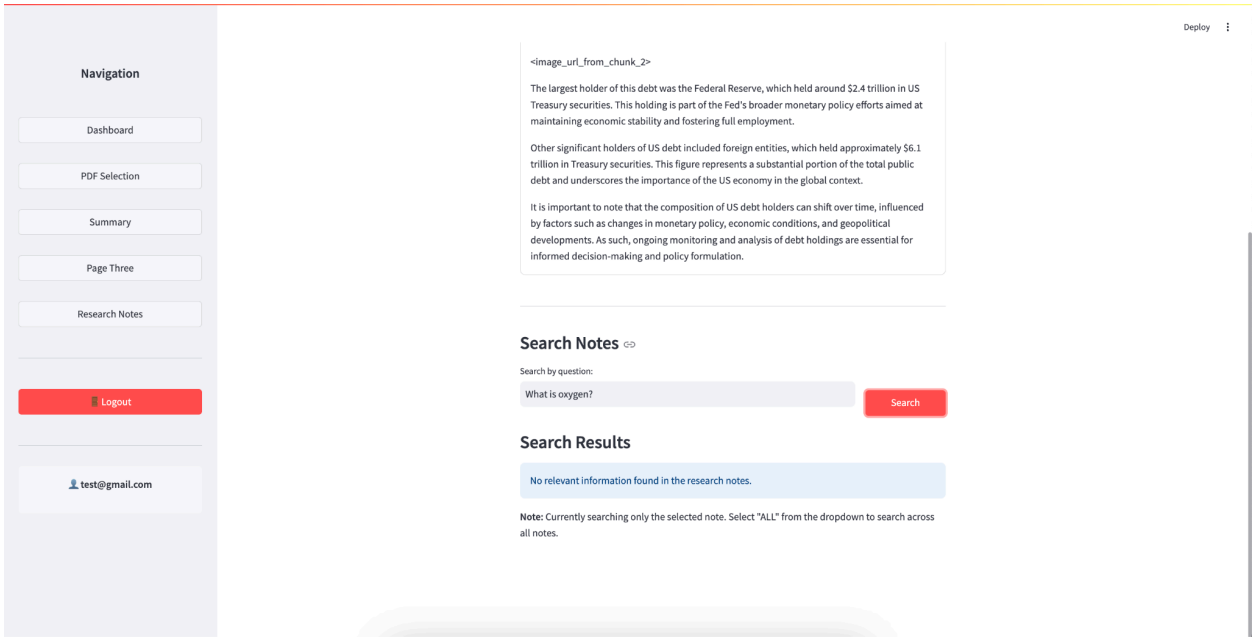


When a question is present in the document's content, the system displays the relevant answer. The screenshot might show a highlighted answer in response to a user's question, helping users locate information efficiently.

2. **<u>If the Question asked is present in the content it shows the relevant answer:</u>**

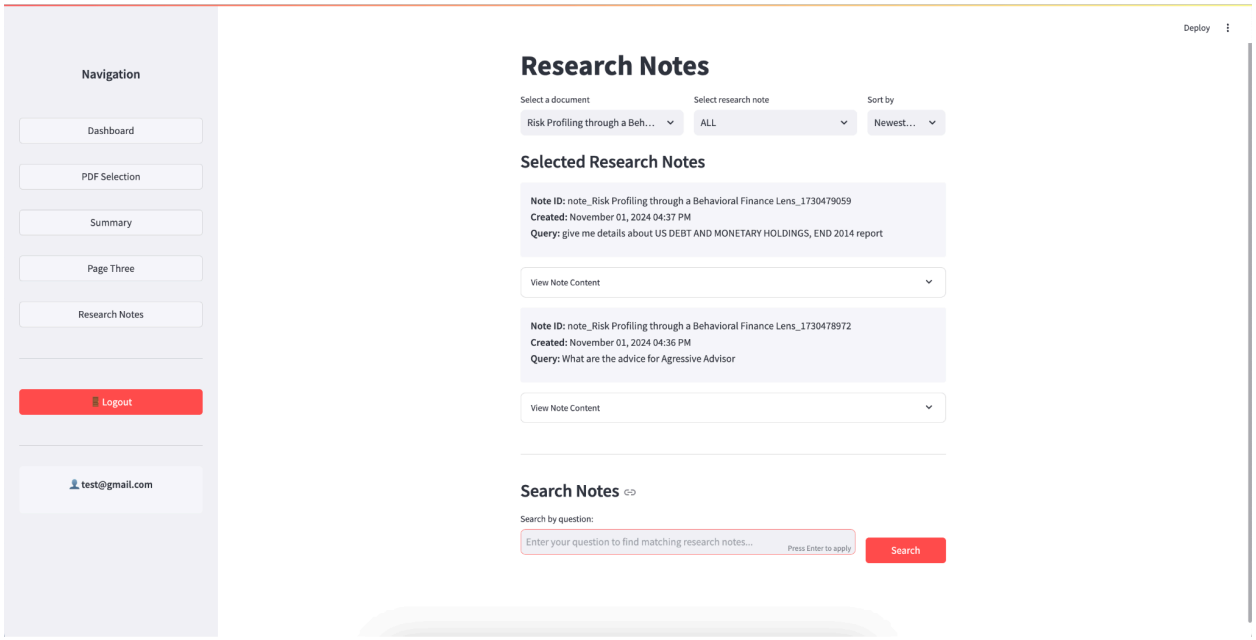


3. **<u>If asked a Question Irrelevant to the Document :</u>**



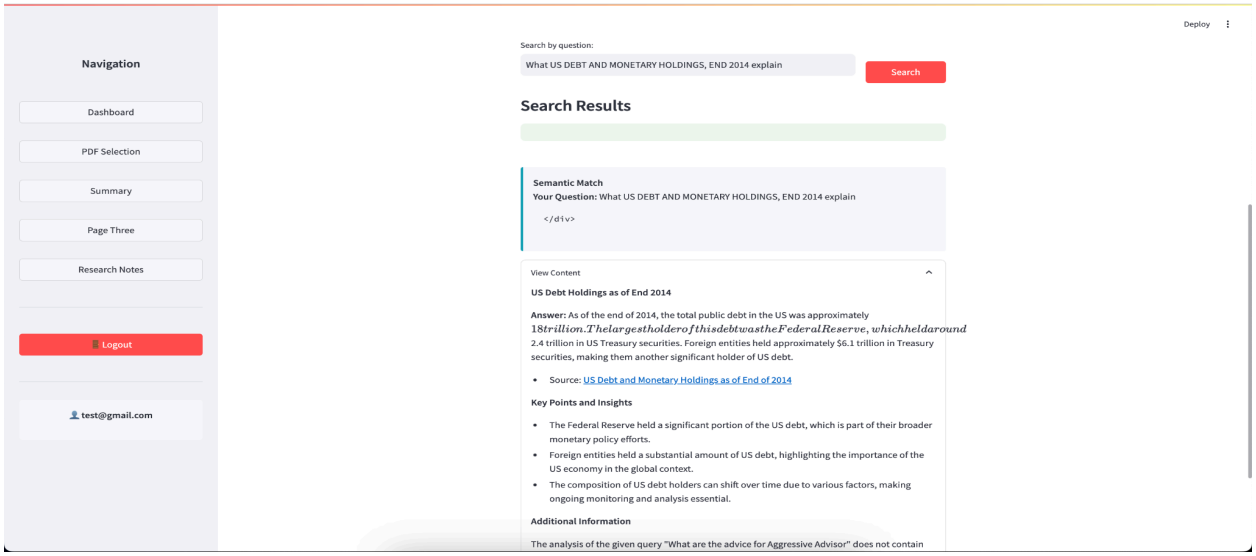
If a user's question is unrelated to the document, the system notifies them that no relevant data is available. This screenshot shows a "No relevant data" response, clarifying that the query does not match any content in the document.

<u>When selecting the option "ALL", it searches and Queries ALL the research notes and the Entire Document</u>



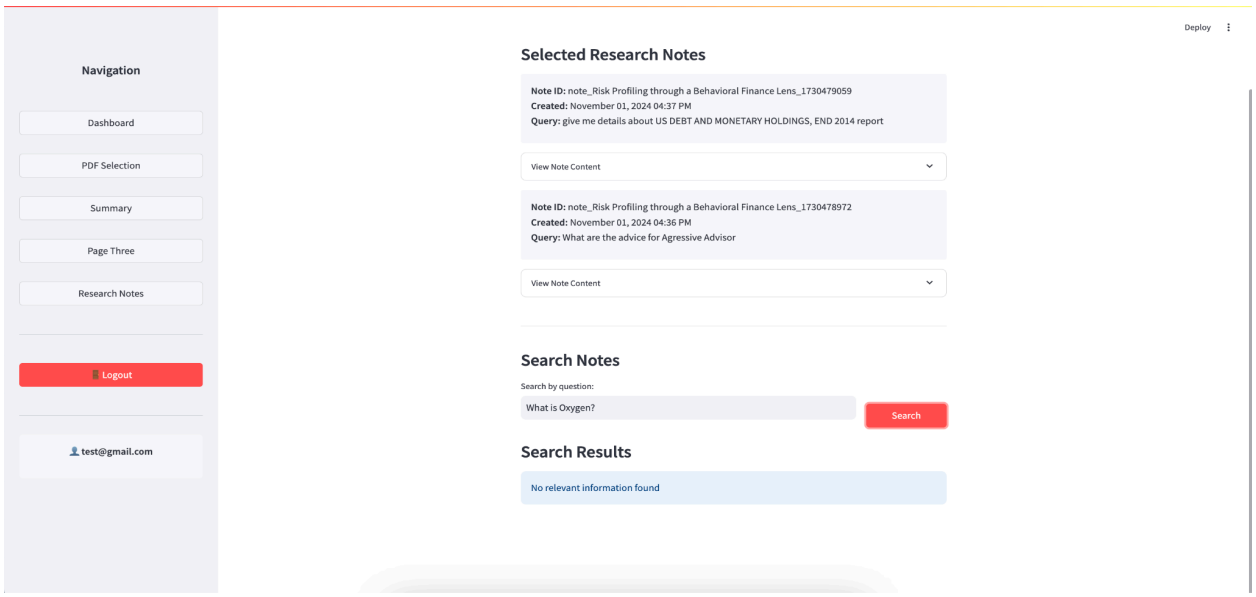
The “ALL” option allows users to expand their search to all research notes and documents. The screenshot may indicate this global search mode, showing how results are differentiated by source for clarity.

If the query is found in the entire document or Research note, it differentiates it using “SOURCE”



When a query is found in both research notes and the document, the results are tagged with the source ("SOURCE") to indicate where the information originates. This helps users understand the context and origin of each result.

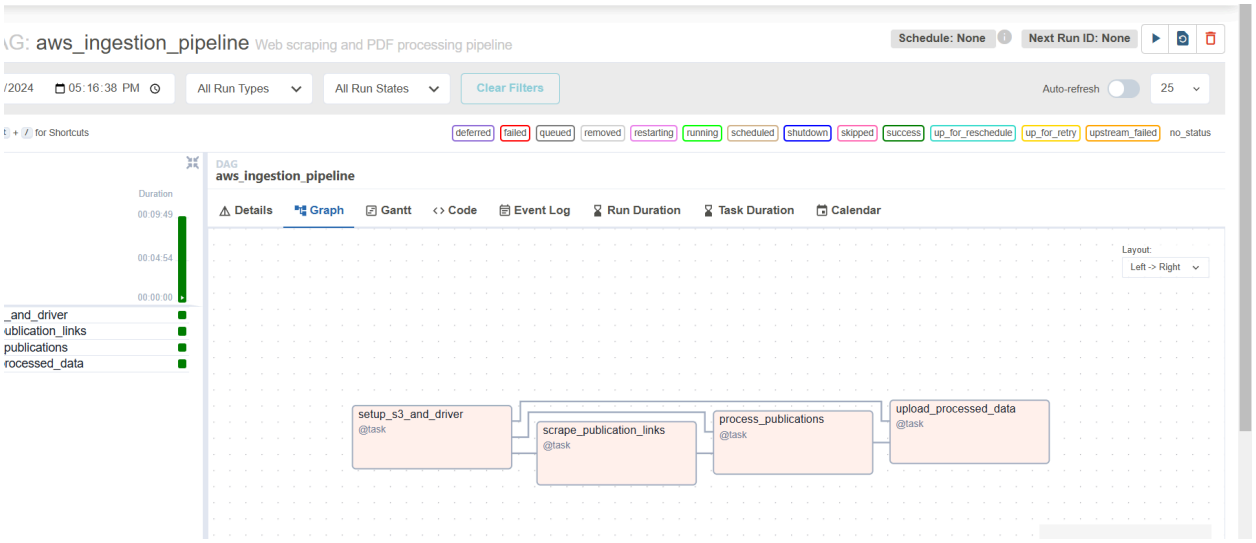
If searched for Irrelevant data it shows No relevant data

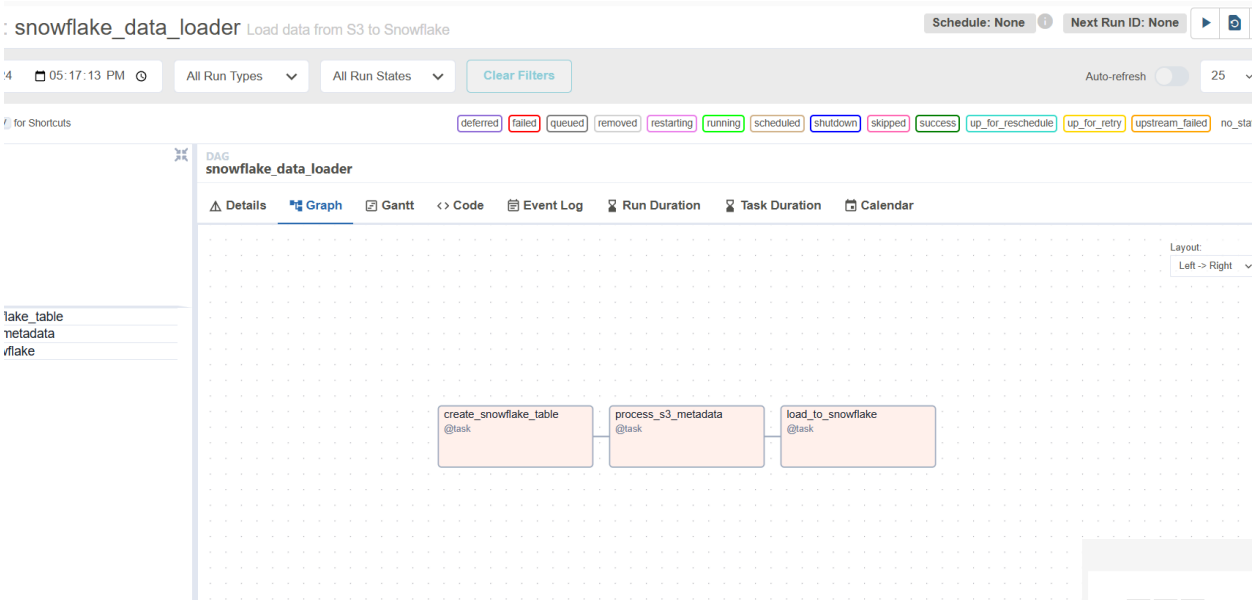


In scenarios where users search for information that does not match any content in the selected document or saved research notes, the system provides a clear **"No Relevant Data Found"** response. This feature ensures that users receive immediate feedback if their query does not yield any results, helping to manage expectations and guide them back to relevant content

Overview of Airflow DAGs

This application workflow consists of two primary data pipelines orchestrated by Airflow DAGs, responsible for the ingestion, processing, and storage of research publications from the CFA Institute Research Foundation. Each pipeline leverages Snowflake and AWS S3 for efficient data management and retrieval.





Pipeline 1: CFA Publication Scraper : The CFA Publication Scraper pipeline ingests metadata and content from CFA Research Foundation publications, transforming and storing it in Snowflake for further analysis and application use.

DAG Structure and Tasks:

- Start DAG: Initializes the pipeline with logging and validation.
- Scrape Metadata: Web scrapes publication metadata, including title, URL, summary, and publication date.
- Transform Data: Processes and structures scraped metadata.
- Store in Snowflake: Inserts processed publication metadata into Snowflake's CFA_PUBLICATIONS table.
- Validation Task: Ensures that the stored data in Snowflake aligns with the schema and contains the expected records.

Pipeline 2: Snowflake and S3 Data Loader: This pipeline automates the transfer and loading of metadata files from S3 into Snowflake for publications, enabling more efficient ingestion, storage, and accessibility.

DAG Structure and Tasks:

- Initialize Environment: Loads environment variables for Snowflake and AWS configuration.
- Setup S3 Client: Establishes a secure S3 connection using AWS credentials.
- Fetch Metadata from S3: Retrieves metadata JSON files from the specified S3 bucket.
- Process Metadata: Transforms metadata to extract URLs, titles, and other publication details, converting them into a standardized format.
- Insert Publication Data: Loads each processed metadata entry into the CFA_PUBLICATIONS Snowflake table.
- Cleanup and Close Connections: Ensures that resources are released and connections are closed after data loading.

  This workflow supports seamless data integration, enabling accurate and efficient search and retrieval for research publications in the application's front end.

FastAPI for Backend Services

FastAPI powers the backend of the application, providing a high-speed API layer for document retrieval, summary generation, and user queries:

1. **Document Exploration API**: Enables users to browse document metadata and select documents via grid or dropdown views.
2. **Dynamic Summaries**: Generates on-demand document summaries using NVIDIA's API, condensing complex content into accessible summaries.

3. **Multi-Modal Retrieval (RAG)**: Uses LLAMA and NVIDIA models to deliver relevant content based on user queries.
4. **Q/A Interface**: Allows users to ask questions, retrieving contextual answers from document embeddings in Pinecone.
5. **Report and Research Note Generation**: Creates detailed, link-embedded reports, storing verified answers as research notes for future reference.

Deployment of the Workflow

The application is containerized and deployed on GCP using Docker, ensuring scalability, public accessibility, and consistent performance:

1. **Containerization**: Both FastAPI and Streamlit are managed in Docker containers, with Docker Compose coordinating their interaction.
2. **Public Accessibility**: Hosted on GCP, both the API and frontend are accessible remotely, supporting secure, scalable interactions.
3. **Scalability**: GCP enables dynamic scaling, ensuring that the application can handle increasing user and query demands.

Summary of the Application Workflow:

1. **Data Ingestion**: Airflow scrapes CFA publications, storing metadata and document content in Snowflake and S3 for efficient processing.
2. **Backend Services**: FastAPI handles document queries, dynamic summaries, Q/A interactions, and report generation.
3. **User Interaction with Streamlit**: Users can browse documents, generate summaries, save research notes, and perform document-specific queries.
4. **Research Notes Search**: Pinecone indexes research notes, enabling users to search across notes and documents, with results differentiated by source.

References

- Apache Airflow Documentation: https://airflow.apache.org/docs/apache-airflow/stable/
- Streamlit: https://streamlit.io/
- FastAPI: https://fastapi.tiangolo.com/
- LLAMAMultimodal Report Generation Example llama_parse/examples/multimodal/multimodal_report_generation.ipynb at main · run-llama/llama_parse · GitHub
- Multimodal RAG Slide Deck Example llama_parse/examples/multimodal/multimodal_rag_slide_deck.ipynb at main · run-llama/llama_parse · GitHub
- NVIDIAMultimodal RAG Example GenerativeAIExamples/community/llm_video_series/video_2_multimodal-rag at main · NVIDIA/GenerativeAIExamples · GitHub