Assignment 3 - NVIDIA Multimodal Research Assistant

Application link: http://34.148.55.30:8501/

Video Link: Assignment3_Demo_Presentation.mp4

Team member 1: Dharun Karthick Ramaraj

Team member 2: Nikhil Godalla

Team member 3: Linata Deshmukh

Introduction

This project is a multimodal research assistant designed to extract, summarize, and interact with documents using NVIDIA's AI models, FastAPI, and Airflow. The primary goal is to automate the process of scraping documents, summarizing them, and providing an interactive Q&A interface for users to explore publications. Additionally, users can query not only the documents but also their research notes, which are indexed separately.

We integrated multiple technologies such as **Airflow**, **AWS S3**, **Snowflake**, **Pinecone**, **NVIDIA LLM**, and **FastAPI** to achieve this goal. The platform leverages a **Retrieval-Augmented Generation (RAG)** framework, which retrieves relevant chunks of a document using vector embeddings and passes them to an LLM (NVIDIA's llama-3.1-nemotron-51b-instruct model) to generate context-aware responses. The result is an interactive platform where users can explore research documents, generate summaries, perform document-based Q&A, and store notes for future reference.

Technologies Involved:

- **Airflow**: Orchestrates data pipelines for scraping and processing documents.
- **AWS S3**: Stores publication files, summaries, and research notes.
- **Snowflake**: Stores and manages metadata for publications.
- **Pinecone**: Indexes document embeddings for efficient querying.
- **NVIDIA AI Models**: Used for summarizing documents and answering questions.
- **FastAPI**: Provides a backend API to serve data and manage interactions with the frontend.
- **Streamlit**: A frontend web interface for users to explore documents and interact with Q&A functionalities.

Goals:

- **Automated Document Ingestion**: Automate the scraping, extraction, and processing of publication documents.
- **Interactive Exploration**: Provide a user-friendly web interface for exploring publications.
- **AI-Powered Summarization and Q&A**: Use NVIDIA's large language models to generate concise summaries and provide answers to document-based questions.
- **Research Notes Management**: Allow users to take and save research notes, which are indexed for efficient querying.
- **Efficient Querying using RAG**: Implement a RAG framework that retrieves the top relevant document chunks and passes them to an LLM for generating accurate responses.
-

Problem Statement

The exponential growth in digital publications, particularly research documents, has led to increasing challenges in effectively managing, exploring, and extracting insights from large document collections. Manual processing of these documents is not only time-consuming but also inefficient in handling large volumes of data. Researchers often require concise summaries, context-aware answers to specific queries, and an efficient way to take and manage research notes.

Challenges:

1. **Data Ingestion & Management**: Extracting and storing publication data efficiently, including metadata, images, summaries, and full-text documents.
2. **Automating Data Processing**: Automating the end-to-end data ingestion, scraping, and uploading processes with proper data validation and secure storage.
3. **Interactive Document Exploration**: Providing a user-friendly interface for document selection and exploration.
4. **AI-Powered Summary Generation**: Generating concise, context-aware summaries on-the-fly for user-selected documents using AI models.
5. **Efficient Retrieval and Q&A**: Implementing an efficient querying system based on user inputs, avoiding full document exchanges to improve efficiency.
6. **Research Notes Management**: Allowing users to create, save, and incrementally index research notes with the documents for future reference and search.
7. **Incremental Indexing and Search**: Maintaining separate indexes for documents and research notes while providing the capability for hybrid search functionalities.
8. **Public Accessibility and Security**: Ensuring that the system is publicly accessible, secure, and user-friendly, while maintaining data privacy.

Solution Overview:

To address these challenges, the **NVIDIA Multimodal Research Assistant** project was developed with three key parts:

Part 1: Data Ingestion and Database Population

- **Data Ingestion**: Scrape data from CFA Institute Research Foundation Publications, extract titles, images, summaries, and PDF files, and store them in an S3 bucket. Set up a Snowflake table to store metadata and automate the ingestion process using Airflow pipelines.

Part 2: Client-Facing Application using FastAPI + Streamlit

- **FastAPI Service**: Provide an API to explore documents, display publications, generate AI-powered summaries on-the-fly using NVIDIA services, and offer a multi-modal Retrieval-Augmented Generation (RAG) based Q&A interface.
- **Streamlit Frontend**: Develop a user-friendly interface for document selection, summary generation, and Q&A interaction, focusing on security and a smooth user experience.

Part 3: Research Notes Indexing and Search

- **Incremental Indexing**: Use a multi-modal RAG to incrementally index research notes for each document, allowing differentiated searches between the document's full text and the research notes index.
- **Search Functionality**: Provide options to search within the full document text, research notes, or both. Enable derived research notes to be added to the index for continuous learning.

Constraints and Requirements:

The solution requires integrating containerized services with Docker, secure data storage with AWS S3 and Snowflake, AI-powered summarization using NVIDIA's large language models, efficient indexing with Pinecone, and a publicly accessible deployment on a cloud platform. Comprehensive documentation, GitHub repository management, and deployment instructions ensure accessibility and ease of use for end-users.

Proof of Concept

The proof of concept focuses on demonstrating that the proposed architecture, data pipelines, and AI models can effectively address the outlined challenges and achieve the project's objectives. The goal is to establish a foundation for seamless integration between the components and validate the chosen technologies' effectiveness in delivering the required functionalities.

Major Technologies and Their Roles

1. **FastAPI**:
    - Acts as the backend API service, handling requests from the frontend and interfacing with external services like Snowflake, AWS S3, and NVIDIA's models.
    - Facilitates document exploration, querying, summarization, and interaction with databases and storage.
2. **Streamlit**:
    - Provides a user-friendly frontend interface, allowing users to explore documents, view summaries, perform Q&A, and manage research notes.
    - Interacts with the FastAPI backend to fetch data, update research notes, and display results in an intuitive format.
3. **Airflow**:
    - Automates the data ingestion and processing workflows using well-defined DAGs (Directed Acyclic Graphs).
    - Orchestrates the scraping of publication data, extraction of text from PDFs, and loading of metadata into Snowflake.
    - Ensures consistent and repeatable data ingestion processes with secure handling of S3 interactions.
4. **NVIDIA AI Models**:
    - **llama-3.1-nemotron-51b-instruct** is utilized for generating high-quality summaries and answering complex research questions.
    - **NVIDIA Embeddings** are used for creating vector representations of documents and research notes, enabling efficient querying through Pinecone.
5. **Pinecone**:
    - Serves as the vector database to store and manage embeddings generated from documents and research notes.
    - Supports fast and accurate retrieval of relevant information using vector similarity searches, allowing for multi-modal Retrieval-Augmented Generation (RAG).
6. **Snowflake**:
    - Stores metadata and publication information, such as titles, images, links to S3-stored PDFs, and summaries.
    - Acts as a centralized repository for structured publication data.
7. **AWS S3**:
    - Provides scalable storage for storing large volumes of publication files, extracted text, images, and research notes.

    - Ensures secure and accessible storage for project assets.
8. **Docker and Docker Repository**:
    - Containerizes FastAPI, Streamlit, and Airflow components for consistent deployment across development, staging, and production environments.
    - Images are pushed to a Docker repository and can be pulled onto a cloud instance for deployment, ensuring efficient and reliable deployment of the solution.

Initial Setup and Configuration

1. **Data Scraping & Ingestion**:
    - Set up an Airflow environment with a Dockerized Airflow pipeline for running DAGs to scrape data from the CFA Institute Research Foundation Publications.
    - Validate that extracted data (title, image, brief summary, and PDF) is uploaded to S3 and that metadata is stored in Snowflake.
2. **FastAPI and Streamlit Configuration**:
    - Set up FastAPI routes for fetching publication details, summarizing documents, handling Q&A requests, and managing research notes.
    - Configure Streamlit pages (Grid View, Detail View, Q/A Interface) to interact with the backend FastAPI service and display publication data.
    - Validate communication between the Streamlit frontend and FastAPI backend.
3. **AI and Embeddings Testing**:
    - Integrate NVIDIA's llama-3.1-nemotron-51b-instruct model with FastAPI to generate on-the-fly summaries for selected publications.
    - Test the NVIDIA Embeddings model to generate vector embeddings for documents and research notes.
    - Set up Pinecone to store document and research notes embeddings and validate vector similarity searches.

Deployment Approach

1. **Docker Image Preparation**:
    - Containerize all application components, including Airflow pipelines, FastAPI, and Streamlit.
    - Push the container images to a Docker repository for secure storage and version control.
2. **Cloud Deployment**:
    - Launch a cloud instance (e.g., AWS EC2) and configure necessary security and access settings.
    - Pull the container images from the Docker repository onto the cloud instance.
    - Start the containers using Docker Compose or Kubernetes, ensuring that all services are running and accessible to the public.

Challenges and Mitigation

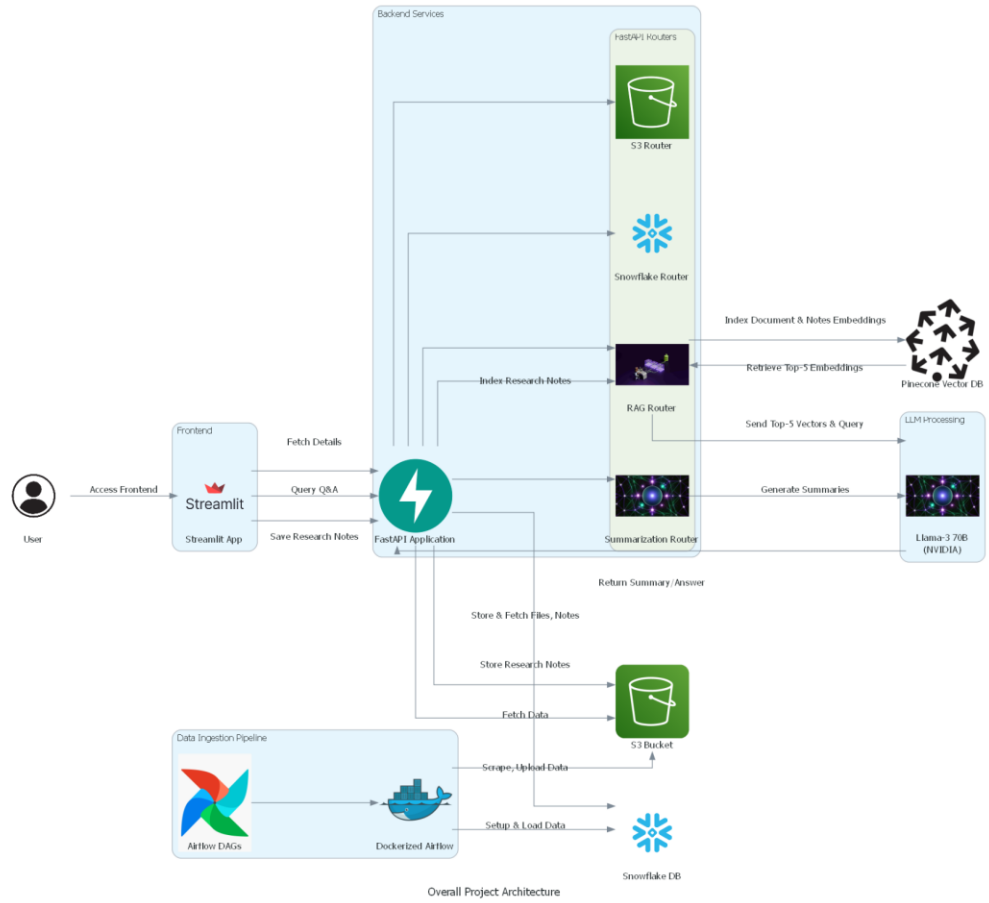
1. **Data Volume and Scalability**:

    - Airflow is used to automate and schedule data ingestion tasks, ensuring consistency even with large datasets. Docker containerization guarantees an isolated and scalable deployment.
2. **Efficient Querying and Summarization**:
    - Multi-modal RAG is implemented to limit the scope of the search and avoid full-document exchanges. This reduces response times and improves efficiency.
3. **Secure Storage and Management**:
    - AWS S3 and Snowflake provide secure and scalable storage solutions, while access is managed using environment variables and IAM roles.

Validation and Proof

- Successfully deploy the solution using Docker images, ensuring that FastAPI and Streamlit are containerized and accessible via the cloud.
- Validate the full end-to-end flow by scraping and storing publication data, generating summaries, performing Q&A, and saving research notes.
- Demonstrate efficient querying of research notes and documents using multi-modal RAG, backed by NVIDIA’s large language models and embeddings stored in Pinecone.

Architecture Diagram



The architecture diagram is divided into several key components and clusters, each playing a specific role in the project's workflow. Below are the main components described in the diagram:

1. **Frontend**:
    - **User**: Represents the end-user who interacts with the system through the Streamlit application.
    - **Streamlit App**:
        - **Grid View**: Allows users to browse through publications and select a document.
        - **Detail View**: Displays selected publication details, summaries, and research notes. Also offers an option to navigate to the Q&A interface.
        - **Q&A Interface**: A dedicated interface for users to perform question-answering tasks and manage research notes.
2. **Backend Services**:

    - **FastAPI Application**: The core backend service that interacts with multiple routers to handle data storage, retrieval, and querying operations. It acts as the bridge between the frontend and the backend database and services.
    - **FastAPI Routers**:
        - **Snowflake Router**: Manages interactions with the Snowflake Data Warehouse, fetching publication metadata and storing structured data.
        - **S3 Router**: Handles file storage and retrieval from the S3 bucket, managing PDFs, images, summaries, and research notes.
        - **Summarization Router**: Responsible for generating summaries using **NVIDIA llama-3.1-nemotron-51b-instruct**.
        - **RAG Router**: Manages document indexing, querying, and interactions with the Pinecone Vector Database using NVIDIA embeddings.
3. **Data Ingestion Pipeline**:
    - **Airflow DAGs**: Defined tasks for scraping publications, extracting text, loading metadata, and managing data flow.
    - **Dockerized Airflow**: A Docker containerized environment for running the Airflow pipelines and scheduling tasks.
4. **Data Storage and Vector DB**:
    - **Amazon S3 Bucket**: The primary storage solution for storing files, images, summaries, and research notes.
    - **Snowflake Data Warehouse**: Stores structured publication metadata for efficient querying and management.
    - **Pinecone Vector DB**: Stores document and research note embeddings for fast retrieval and matching during document queries.
5. **LLM Processing**:
    - **NVIDIA llama-3.1-nemotron-51b-instruct**: Utilized for generating summaries and contextually relevant answers to user queries. It works alongside the RAG Router to provide high-quality answers based on document and research note embeddings.

Walkthrough of the Application

1. Grid View - Explore and Select Publications

- **Purpose**: This page provides an overview of all available publications and allows the user to select one for detailed exploration.
- **Description**:
    - Users are presented with a grid layout that displays the title and cover image of each publication retrieved from Snowflake.
    - Users can click on a publication to view more details.
- **Key Features**:
    - Quick visual access to all publications.
    - Selection of a specific document to explore further.
- **How It Works**:
    - The front end calls the FastAPI application, which fetches publication details from Snowflake and displays them in the grid.
    - Upon selection, the application stores the selected document ID in the session state and navigates to the detailed view.

2. Detail View - View Summary and Research Notes

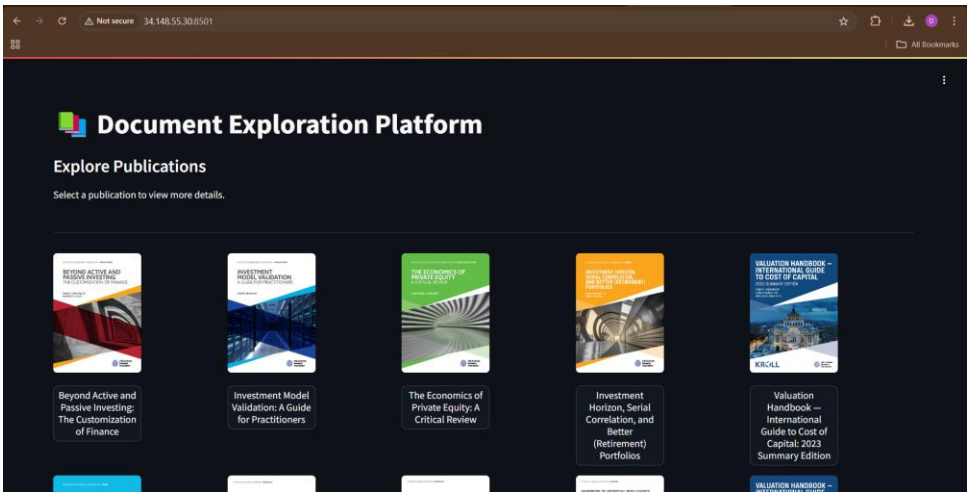
- **Purpose**: This page provides a more in-depth look at a selected publication, including its summary and associated research notes.
- **Description**:
    - Displays a concise summary of the selected document, which is generated on-the-fly using the **NVIDIA llama-3.1-nemotron-51b-instruct** model.
    - Users can refresh the summary if needed and view, add, or update research notes for the selected publication.
    - A button labeled **"Take Me to Q&A Interface"** allows the user to switch to the Q&A mode.
- **Key Features**:
    - Summary generation with an option to refresh.
    - Display and update of research notes.
    - Seamless navigation to the Q&A interface.
- **How It Works**:
    - When a publication is selected, the frontend retrieves its metadata and summary from the backend using FastAPI.
    - If the user requests a summary refresh, the FastAPI backend sends a request to the **NVIDIA llama-3.1-nemotron-51b-instruct** model for generating a new summary based on the document's extracted text from S3.
    - Research notes are fetched from S3 and displayed on this page, with the ability to add new notes.

3. Q&A Interface - Ask Questions, Take Notes, and Generate Reports

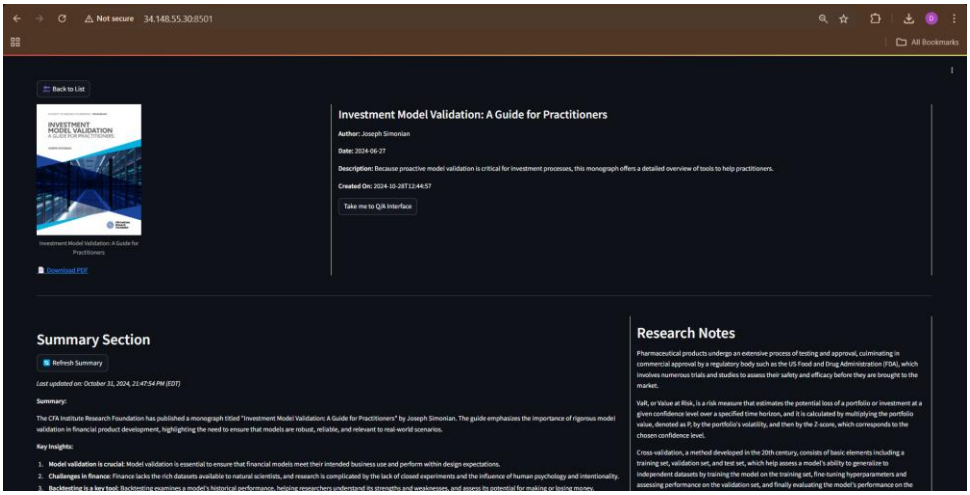
- **Purpose**: This interface allows users to ask questions about the selected document and view the results, including the ability to save answers to research notes and generate a comprehensive report based on saved research notes and recent queries.
- **Description**:
    - A text box is provided for users to input their questions related to the selected document or research notes.
    - The system retrieves the document or research notes from the Pinecone Vector Database and sends the top-5 relevant sections to **NVIDIA llama-3.1-nemotron-51b-instruct** to generate an answer.
    - Users can add the response to their research notes and save it for future reference.
    - The **Report Generation** feature analyzes the existing research notes and recent queries to create a detailed report. This report captures key insights, trends, and relevant information from both research notes and document queries.
- **Key Features**:
    - Interactive Q&A system using **NVIDIA llama-3.1-nemotron-51b-instruct** and Pinecone.
    - Option to append relevant answers to research notes and save them.
    - Indexed search on both the full document and research notes based on user selection.
    - **Report Generation** based on accumulated research notes and recent user queries.
- **How It Works**:
    - Users type in a question related to the selected publication or research notes.
    - The FastAPI application sends the question and document embeddings to Pinecone, which returns the top-5 most relevant sections.
    - These sections are passed to **NVIDIA llama-3.1-nemotron-51b-instruct**, which generates a contextual answer.
    - Users can review the answer and save it to their research notes, which are then stored in S3 and indexed using the RAG router.
    - When users choose to generate a report, the application gathers the saved research notes and recent queries. It then organizes and structures these insights into a cohesive report, summarizing key findings and linking them to their respective sources.
- **How to Generate a Report**:
    - Users can click on the **"Generate Report"** button on the Q&A Interface.
    - The system will automatically review all saved research notes and recent queries related to the current publication.
    - A final report is generated and displayed to the user, highlighting the most relevant insights and linking them to original sources within the document.

Snapshots of the Application:

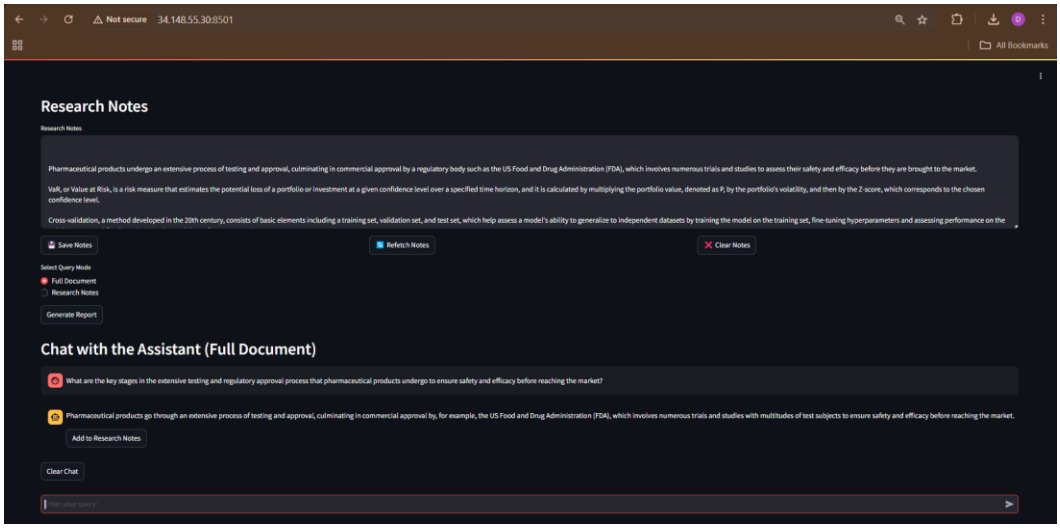
- **Grid View**:



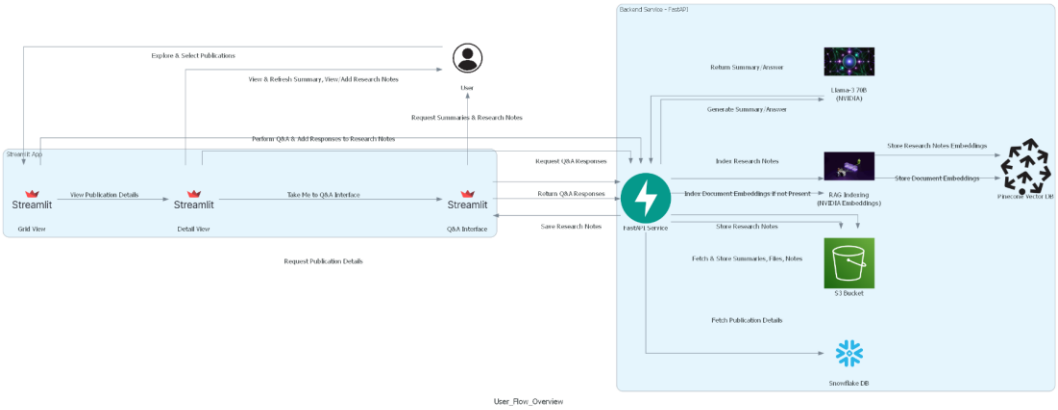
- **Detail View**:



- **Q&A Interface**:



User Flow Diagram:



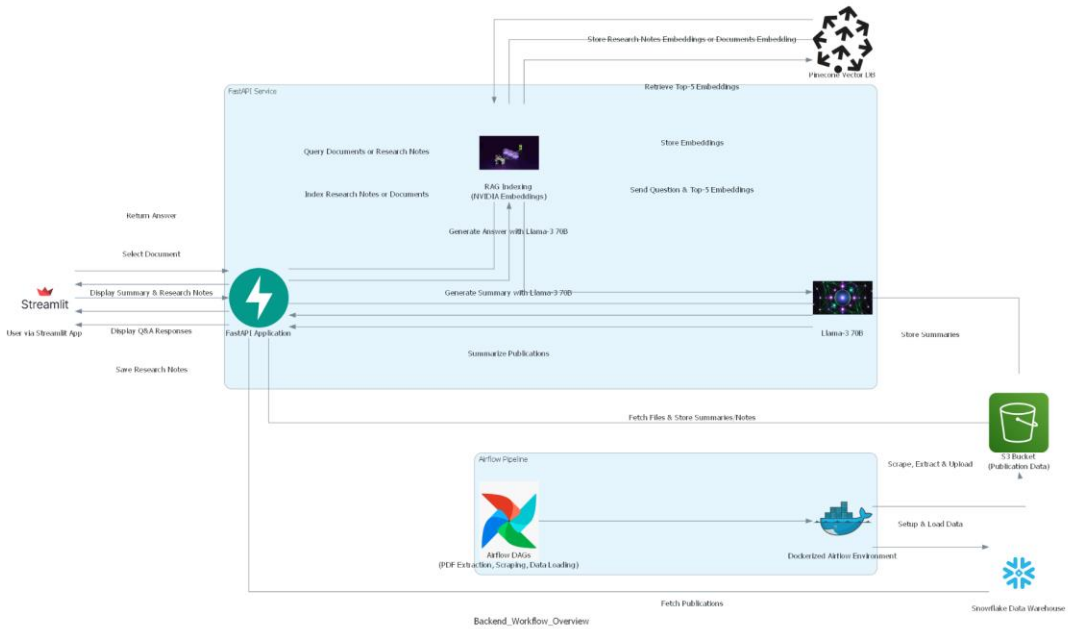
User_Flow_Overview

Key User Interactions:

1. **Publication Selection**:
   - User opens the application and sees the **Grid View** with a list of publications.
   - User selects a publication to explore further.
2. **Detail Exploration**:
   - User sees detailed information about the selected publication, including a generated summary and associated research notes.
   - User can refresh the summary or add new research notes.
3. **Question-Answer Interaction**:
   - User navigates to the **Q&A Interface** and asks questions about the publication.

- User receives contextual answers based on both the document content and the saved research notes.
- User saves the answer to the research notes if needed.

Application Workflow



In this section, we dive deep into the application's backend and the overall workflow, focusing on the data engineering tasks and backend processes. The goal is to explain how data flows through the system and to highlight key areas of processing, storage, and interaction between various components.

Full Data Flow Overview

The application's data flow starts from the data ingestion phase and proceeds to the document querying and research notes indexing.

1. **Data Ingestion Pipeline**:
    - **Scraping & Extracting**:
        - The Airflow pipeline initiates by triggering the `scrape_cfa_publications_dag.py` DAG. This DAG uses Selenium to scrape CFA publications, retrieving data like titles, brief summaries, images, and PDFs.
        - The scraped content is uploaded to **AWS S3** (PDFs and cover images) using the **S3 Router**.
        - Airflow triggers `pdf_extraction_dag.py` to extract text and content from PDFs using **PyMuPDF** and uploads the extracted text to the **S3 bucket**.
    - **Setting up Snowflake**:
        - The Airflow pipeline triggers the `snowflake_setup_dag.py` DAG to set up the Snowflake infrastructure (warehouses, databases, schemas, and tables).
    - **Loading Data into Snowflake**:
        - The pipeline triggers `snowflake_load_dag.py` to load the metadata CSV (generated during the scraping) into Snowflake tables.
2. **Summary**: Data is scraped, processed, and stored in AWS S3 for document storage and Snowflake for metadata management.
3. **Client-Facing Application**:
    - **Streamlit Grid View**:
        - Displays the list of publications with cover images, brief summaries, and titles.
        - Users can filter and select publications via the grid view or dropdown.
    - **Detail View**:
        - Upon selection, the **Detail View** page fetches detailed information about the publication, including the extracted text, images, and any saved research notes.
        - Users can refresh the summaries on this page, which generates an updated summary using **NVIDIA's llama-3.1-nemotron-51b-instruct model**.
    - **Q&A Interface**:
        - The Q&A Interface allows users to ask questions about the document or research notes. The **RAG (Retrieve and Generate) Router** sends the user's question and the relevant document sections to NVIDIA's llama-3.1-nemotron-51b-instruct for answer generation.
        - Users can append the generated answer to their research notes and save them for indexing.
    - **Report Generation**:

        - Based on saved research notes and recent queries, the system can generate a report summarizing key findings.
4. **Backend Architecture**:
    - **FastAPI Application**:
        - The backend is powered by FastAPI, with routers handling specific interactions:
            - **S3 Router** for interacting with AWS S3 to fetch files, summaries, and store research notes.
            - **Snowflake Router** for fetching metadata and publication details from Snowflake.
            - **Summarization Router** for generating concise summaries of publications using **NVIDIA's llama-3.1-nemotron-51b-instruct**.
            - **RAG Router** for indexing documents and querying based on user inputs, utilizing **NVIDIA Embeddings** and the **Pinecone Vector DB**.
    - **Pinecone and NVIDIA Integration**:
        - Document embeddings are generated using **NVIDIA's Embeddings** and indexed in the **Pinecone Vector Database**.
        - When a user queries a document, Pinecone returns the top-5 relevant sections. These sections are sent to **NVIDIA llama-3.1-nemotron-51b-instruct** along with the question to generate an answer.
    - **Research Notes Indexing**:
        - The saved research notes are also embedded and indexed in Pinecone, allowing for quick retrieval and efficient searching.
5. **Challenges and Solutions**:
    - **Data Consistency**: Ensured consistent data storage and retrieval by establishing clear workflows and using **AWS S3** and **Snowflake** for separation of file storage and metadata management.
    - **Efficient Querying**: Used **NVIDIA's Embeddings** and **Pinecone** to perform scalable and efficient retrieval-augmented generation (RAG) for document queries and research notes.
    - **Modular Application Design**: Separated different functionalities into routers and services within FastAPI, allowing for easy maintenance and scalability.

Key Code Snippets

- **Airflow DAG Example (`scrape_cfa_publications_dag.py`)**:

```python
def scrape_publications_with_selenium(ti, aws_access_key, aws_secret_key, aws_region, s3_bucket_name):
    all_data = []

    for page_num in range(0, 100, 10):  # Increment by 10 up to 90
        base_url = f"https://rpc.cfainstitute.org/en/research-foundation/publications#first={page_num}&sort=%40officialz32xdate%20descen
        driver.get(base_url)
        time.sleep(5)  # Allow page to load fully

        soup = BeautifulSoup(driver.page_source, 'html.parser')
        publications = soup.find_all('div', class_='coveo-list-layout CoveoResult')

        for pub in publications:
            title_tag = pub.find('a', class_='CoveoResultLink')
            title = title_tag.text.strip() if title_tag else None
            href = title_tag['href'] if title_tag else None
            href = f"https://rpc.cfainstitute.org{href}" if href and href.startswith('/') else href

            image_tag = pub.find('img', class_='coveo-result-image')
            image_url = image_tag['src'] if image_tag else None
            image_url = f"https://rpc.cfainstitute.org{image_url}" if image_url and image_url.startswith('/') else image_url

            summary_tag = pub.find('div', class_='result-body')
            summary = summary_tag.text.strip() if summary_tag else None

            pdf_link = None
            if href:
                driver.get(href)
```

- **FastAPI Summarization Router**:

```
backend > fast_api > routers > summarization_router.py > generate_summary
async def generate_summary(request: SummaryRequest):

    # Initialize ChatNVIDIA client with the correct API key
    client = ChatNVIDIA(
        model="nvidia/llama-3.1-nemotron-51b-instruct",
        api_key=os.getenv("NVIDIA_API_KEY"),
        temperature=0.5,
        top_p=1,
        max_tokens=1024,
    )

    # Create the prompt for generating a summary
    prompt = (
        "Create a concise and clear summary for the following text, highlighting key insights and important points. "
        "Keep the summary short and focused on essential information: "
        f"{truncated_text}"
    )

    try:
        # Send the prompt to ChatNVIDIA API and stream the response to generate a summary
        summary_chunks = client.stream([{"role": "user", "content": prompt}])
        summary = "".join(chunk.content for chunk in summary_chunks)

        # Step 3: Overwrite the summary in S3 in the `silver/publication_summary/` path
        s3_client.put_object(Bucket=bucket_name, Key=summary_key, Body=summary.encode('utf-8'))

        return {"summary": summary, "message": "Summary generated and saved successfully!"}
```

- **RAG Router Query Endpoint**:

```python
async def query_index(data: QueryRequest):

    # Create a query engine with specified similarity settings and response mode
    query_engine = index.as_query_engine(
        similarity_top_k=5,
        streaming=False,
        response_mode="tree_summarize"  # This encourages more complete responses
    )

    # Modify the prompt to encourage a complete sentence answer
    enhanced_prompt = f"Please provide a complete sentence answer to the following question: {data.question}"

    # Query the index with the enhanced prompt
    response = query_engine.query(enhanced_prompt)

    # Extract the answer text
    answer = getattr(response, "response")

    # Post-process the answer to ensure it's a complete sentence
    if not answer.endswith(('.', '!', '?')):
        answer += '.'

    # If the answer doesn't seem to be a complete sentence, prepend context
    if not answer[0].isupper() or len(answer.split()) < 3:
```

Key Workflows Explained

1. **Data Ingestion Workflow**:
    - Data is scraped, extracted, and uploaded to S3 using Airflow DAGs.
    - Snowflake tables are created and populated with publication metadata using the Airflow pipeline.
2. **Document Query Workflow**:
    - When a user selects a document, its metadata and related files are fetched from Snowflake and S3.
    - The selected document is indexed in Pinecone if not already present, and embeddings are created using NVIDIA's Embeddings.
    - When a query is made, Pinecone returns the top-5 relevant sections, which are sent along with the question to llama-3.1-nemotron-51b-instruct for generating the final answer.
3. **Research Notes Indexing**:
    - Research notes are saved to S3 and embedded for efficient indexing using NVIDIA's Embeddings.
    - Users can search through the indexed research notes separately from the full document.

References

1. FastAPI Documentation: https://fastapi.tiangolo.com/
2. Streamlit Documentation: https://docs.streamlit.io/
3. Airflow Documentation: https://airflow.apache.org/
4. NVIDIA AI Services: https://build.nvidia.com/explore/discover
5. Pinecone Documentation: https://docs.pinecone.io/
6. Snowflake Documentation: https://docs.snowflake.com/
7. PyMuPDF Documentation: https://pymupdf.readthedocs.io/
8. Docker Documentation: https://docs.docker.com/
9. GitHub Repository Best Practices: https://guides.github.com/features/wikis/