# Assignment 3

Due: 1st Nov 2024 03:59 pm EST

**Required Attestation and Contribution Declaration**
*WE ATTEST THAT WE HAVEN'T USED ANY OTHER STUDENTS' WORK IN OUR ASSIGNMENT AND ABIDE BY THE POLICIES LISTED IN THE STUDENT HANDBOOK*
**Contribution**:

- Member 1: 33%
- Member 2: 33%
- Member 3: 33%

**Instructions**:

## Part 1: Data Ingestion and Database Population

1. **Data Ingestion**:
    - **Scrape Data**: Extract data from CFA Institute Research Foundation Publications. Retrieve details like *Title*, *image*, *brief summary*, and *PDF file* from each publication.
    - **Store Data**: Upload the retrieved images and PDF files to S3.
    - **Database Setup**: Create a table in Snowflake with columns for *Title*, *brief summary*, *image link (S3)*, and *PDF link (S3)*. Load the collected data into this table.
2. **Automating Data Processing**:
    - Use Airflow pipelines for scraping and uploading data to S3.
    - Integrate the ingestion process with the Snowflake database to automate data storage.

## Part 2: Client-Facing Application using FastAPI + Streamlit

1. **FastAPI**:
    - **Explore Documents**: Develop an API allowing users to explore the stored documents.
    - **Grid and Dropdown Selection**: Implement a Streamlit interface where users can select a document via a grid view (image) or dropdown list.
    - **On-the-Fly Summary Generation**: Display selected documents and generate summaries using NVIDIA services.
    - **Multi-modal RAG**: Integrate a multi-modal RAG for querying documents based on user inputs. (Refer to LLAMA and NVIDIA examples).
    - **Q/A Interface**: Develop a Q/A interface for interacting with documents using the multi-modal RAG. Ensure it avoids full document exchanges for efficient processing.

    - **Report Generation**: Use the report generation feature to provide answers to user queries. Format responses as research notes, ensuring they include links to relevant graphs, tables, and pages.
    - **Validation and Storage**: Manually verify the generated answers for accuracy before saving them as "Research Notes" linked to each document.
2. **Streamlit Application**:
    - Create a user-friendly interface for document selection, summary generation, and Q/A interaction.
    - Ensure a smooth user experience with a focus on security and data privacy.

## Part 3: Research Notes Indexing and Search

1. **Incremental Indexing**:
    - Use multi-modal RAG to incrementally index research notes for each document.
    - Maintain separate indexes for each document or implement a hybrid search that filters based on document IDs.
2. **Search Functionality**:
    - Display saved research notes when revisiting a document.
    - Enable search within research notes specific to a document or across the entire document.
    - Differentiate between searching through the document's full text and the research notes index.
    - Allow derived research notes to be added to the original research note index for continuous learning.
3. **Example Searches**:
    - Research questions and searching through both full documents and research notes:
        1. "How did the economy grow in 2023?" (Search full document)
        2. "How did the economy grow in 2024?" (Search full document)
        3. "Did the economy grow or fall in 2024 compared to 2023?" (Search through research note index)

## Deployment:

1. **Containerization**:
    - Ensure the FastAPI and Streamlit applications are containerized using Docker.
    - Deploy to a public cloud platform using Docker Compose for ease of access.
2. **Public Accessibility**:
    - Ensure that both the API and the Streamlit application are publicly accessible.
    - Provide clear instructions for users to interact with the application and explore its functionalities.

## Submission:

1. **GitHub Repository**:
    - Include a *Project Summary*, *PoC*, and other relevant information.
    - Use GitHub Issues to track tasks and optimize the implementation process.
    - Include diagrams, a fully documented *Codelab*, and a *5-minute video* showcasing the solution.
    - Provide a link to the hosted application and backend services.
2. **Documentation**:
    - Provide comprehensive documentation, including a *README.md* detailing the repository's structure.
    - Ensure that users have clear guidance for accessing and using the deployed applications.

## Resources:

- LLAMA Multimodal Report Generation Example
- Multimodal RAG Slide Deck Example
- NVIDIA Multimodal RAG Example