Assignment 1

Due: 31st Jan 2025, 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%

Links to github tasks and tasks owned by each member

Instructions:

You are working in a startup that is planning to build an AI application that requires a significant amount of data from unstructured sources like pdf files and web pages. To evaluate feasibility and understand tool compatibility and capability, you have been tasked to evaluate multiple products and build out a prototype application that can be used to test out the capabilities. Your team is given the following requirements:

- You need to try open source tools and an out of the box enterprise service and compare and contrast features.
  For the downstream process, you need to organize the files you scraped into a standard format for which you will use docking/markitdown. Review pros and cons of using both
- You need to organize the files with links to images for retrieval later in an S3 bucket. Research and come up with a methodology to organize your files.

Tasks:

Format	Approach		Tools to Convert to Markdown
	Python Open Source	Enterprise Service	
PDF	pypdf2	Adobe Acrobat etc	Docling, Markitdown
Web	Beautiful Soup, scrapy etc	?	Docling, Markitdown

1. **Data Extraction:**
    - PDF Files:
        - Develop two Python functions to parse selected PDF files containing images, charts, and tables.
        - Utilize open-source tools such as `PyPDF2` and `pdfplumber` for text and element extraction.
        - Implement an enterprise service like Microsoft Document Intelligence to process PDFs and convert them to Markdown format.
    - Webpages:
        - Develop two Python functions to scrape chosen webpages containing images, charts, and tables.
        - Use libraries like `BeautifulSoup` for HTML parsing and `requests` for HTTP requests.
        - Implement Microsoft Document Intelligence to process webpage content and convert it to Markdown format.
2. **Comparison of Tools:**
    - Open-Source vs. Enterprise Solutions:
        - Evaluate the performance, accuracy, and ease of use between open-source tools (`PyPDF2`, `pdfplumber`, `BeautifulSoup etc`) and enterprise services (Microsoft Document Intelligence).
        - Document the pros and cons of each approach, considering factors like cost, scalability, and integration capabilities.
3. Standardization with **Docling** and **MarkItDown**:
    - Integration:
        - Develop functions to integrate Docling and MarkItDown into your processing pipeline.
        - Use **both** tools to convert and standardize the extracted content into Markdown format, preserving the structure and formatting.
        - Compare and contrast Pros and Cons for both approaches

Part 2: File Organization and Storage

1. Organizing the output Files in S3:
    - Methodology:
        - Implement a structured naming convention for S3 buckets and objects to facilitate easy retrieval and management.
        - Use prefixes (folders) to categorize files by type (e.g., PDFs, images, Markdown files) and source (e.g., scraped websites, processed documents).
        - Assign metadata tags to objects for enhanced searchability and access control.
    - Best Practices:
        - Ensure bucket names are globally unique and reflect their purpose.
        - Implement proper data partitioning strategies to optimize performance.

        - Apply appropriate access controls and encryption to secure data.

Part 3: API Development and Deployment

1. API Development with FastAPI:
    - Create API endpoints to handle PDF and webpage uploads, triggering the respective processing functions.
    - Develop additional endpoints to interact with Docling and MarkItDown for document standardization.
    - Implement functionality to store processed files and metadata in S3.
2. Client-Facing Application with Streamlit:
    - Develop a Streamlit interface allowing users to upload PDFs or input webpage URLs.
    - Integrate the interface with FastAPI to process inputs and retrieve results.
    - Host the Streamlit application on Streamlit's public cloud platform for accessibility.

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.
3. **AI Use disclosure:**
    - Along with the readme.md, there should be a *AiUseDisclosure.md* detailing what AI tools you used and why

Resources:

- Docling GitHub Repository
- MarkItDown GitHub Repository
- AWS S3 Best Practices
- FastAPI Documentation
- Streamlit Documentation