# Assignment

## Problem Statement

* Implement an agentic RAG system with OpenWebUI as the frontend.
* The system will going to be 100% opensource.
* The system will be using ops for observability, management and evaluation.

## Requirements - The Baseline

Your solution must incorporate the following tools and frameworks:

1. Docling – Document processing
2. LlamaIndex + PGVector/PostgreSQL – RAG methodology
3. Contextual Agentic RAG (Anthropic-style) – Embeddings / LLM / Re-ranking
4. Conversation Memory
5. Citation handling using meta-data
6. Ollama – Model hosting
7. Crew.AI – Agentic orchestration framework
8. Arize Phoenix – Prompt lifecycle management and observability
   1. Initialize prompts in Arize Phoenix
   2. Retrieve prompts from Arize when needed in your code
   3. Use Arize for tracing, debugging, and observability
9. RAGAs – Evaluation of your Agentic RAG application
10. OpenWebUI (Docker deployment) – Chatbot interface connected to your backend API

## Design Choices

The following being not defined in the baseline, must be chosen by the candidate

* The REST APIs controller + API contract
* The backend code structure
* The document splitter (chunking) strategy
* The embedding model
* The LLM model on the Ollama
* The retrieval mechanism
* The re-ranking mechanism
* The prompts for the crew's agent(s)
* The integration between OpenWebUI and the backend
* The assignment deployment structure using Docker and Composer

# Enhancements

After baseline implementation, one can do the following enhancements

1. Have more better splitter
2. Change the embedding model to a better one
3. Change the retrieval and/or re-ranking strategy
4. Proper implementation of short term memory (conversational memory)
5. Better grounding
6. Add guardrails in the system
7. Add more agents to the crew
   1. Validating the response from the first agent
   2. Add filter agent in the beginning for regenerating the query
   3. Add router to also take care of the non-question queries
8. Implement Self-Reflective, Corrective RAG strategies, etc.

Note: Doesn't value enhancement on New UI

# Delivery

The following will be the applications in the assignment

1. The document indexer ... A pythonic console based application
2. A custom backend ... A pythonic application that exposes REST endpoints for the OpenWebUI application
3. RAG evaluator ... A pythonic console based application that uses RAGAs to evaluate the current RAG implementation in #2
4. PostgreSQL
5. OpenWebUI
6. Phoenix
7. Ollama

Deliver the assignment as a public GitHub repo with the working code on the 'master' branch

Make sure to have README.md, fully defined with all the details on how to setup the code, and run it.

# The Final Interview

## Presentation

Prepare a slide deck for the interview with the following slides ...

1. Problem Statement, requirements, your design-choices
2. Overall software design that shows the RAG ingestion and retrieval workflows
3. How can you enhance the assignment and what value it will bring to the software solution?

## Assessment Criteria

Your work will be evaluated on:

* Successful integration of backend with Open WebUI (Docker deployment + API connectivity)
* Correct indexing and retrieval of client-provided documents
* Accuracy and contextual relevance of chatbot responses
* Proper use of Arize Phoenix (prompt initialization, retrieval, tracing, debugging)
* Code quality and documentation clarity
* Creativity and completeness of your implementation