Major Project: Subject Matter Expert AI Agent

Mid Evaluation: **25th October 2025**
Final Submission Deadline: **12th November 2025 (Fixed; No Change)**
Evaluation starts on 13th November 2025 (class)

Submission link: https://classroom.github.com/a/PHuQlbNP

Project Overview and Objective

This Major Project requires you to design, implement, and document a highly extensible Retrieval-Augmented Generation (RAG) system capable of serving as a subject-matter expert.
Focused on Agentic Capabilities, RAG, Workflow, and Tool Calling

Define the SME and Scope

The first step is to define the specialized domain for your AI agent. Teams must select one domain and create a detailed description of the AI SME's role, capabilities, and limitations.

Choose a Domain

- K–12 Education: Assisting students and teachers with learning materials.
- Engineering: Aiding in technical problem-solving, documentation, and creating learning materials like notes for students and teachers.
- Medical: Supporting medical education and clinical case analysis.
- Legal: Assisting with legal research, case preparation, and document summarization.
- General-Purpose: Focusing on a specific civic area (e.g., traffic rules, good citizenship, workplace safety).

Example SME Role Definitions

- K–12 SME Agent: Helps students with step-by-step explanations, generates adaptive quizzes, and emails progress reports.

- Engineering SME Agent: Solves technical problems, generates lab assignments, prepares final reports (PDF/DOCX), and creates technical presentations (PPT).
- Medical SME Agent: Provides structured explanations, generates clinical case-based exam questions, drafts reference material, and emails compliance reports with disclaimers.
- Legal SME Agent: Summarizes statutes, generates case studies, prepares arguments/counter-arguments, and generates court-style documents (DOCX/PDF).

A core requirement is to design workflows that chain multiple tool calls to fulfill a complex user request. A mandatory feature for all domains is robust question-answering on the subject matter, which can be implemented using a RAG-based approach.

==

Data Preparation

A. Document Collection & Organization

- All source documents must be collected in a root-level directory.
- The system must support heterogeneous formats, including PDF, DOCX, PPTX, TXT, and MD. Web-scraped text files are also encouraged.
- Each document must be associated with metadata (e.g., subject, source, context, timestamp).
- The pipeline should automatically detect types and process documents.
- There is no data volume which can be suggested for the project, it would vary depending on your chosen domain. You need to use your judgement and decide. Collect data you feel is optimal and justify your decision.
- ***Guideline:*** *Typically, every subject/domain should cover most important text books and other authoritative content without duplication. In addition, there can be an equivalent amount of other relevant/supplementary content.*

B. Preprocessing & Chunking

- Decide on the right chunking strategy. Think about various possibilities such as, fixed-size, content-aware (e.g., paragraph-based), and recursive character splitting and decide what you think is the best option for your domain. Justify your decision in the documentation.
- Segment documents at multiple granularities (e.g., 2048, 512, 128 tokens) to facilitate hierarchical retrieval.
- Implement a context-aware overlap strategy to ensure semantic continuity.
- Preprocessing must include deduplication, tokenization, lowercasing, and removal of non-informative content.
- **[BONUS]** Develop an automated batch ingestion pipeline with comprehensive error logging.

C. Embedding & Indexing

- For generating embeddings, teams might consider using libraries like sentence-transformers (with a model like all-mpnet-base-v2 as a starting point). You are encouraged to experiment with and evaluate at least one other domain-specific embedding model.
- Store embeddings with parent-child relationships to maintain document structure.
- Chunks should be indexed into a vector database. Options to explore include Elasticsearch, Pinecone, or Milvus.
- **[BONUS]** Implement a reranking step (e.g., using BGE Reranker) with a fallback to basic similarity search.

For mid evaluation, you have to submit everything till this point.

D. Core SME Capabilities

- Each team should select at least two tasks; for example, expert content generation (such as quiz or report or dynamic learning modules, or curricula creation with export to PDF, DOCX, or slides and email delivery to a specified contact/email from chat history), as well as standard chat or question-answering tasks for their chosen SME.
- Adaptive Explanations & Multi-Step Reasoning: Explain concepts clearly and provide answers based on logical workflows.

- Teams can achieve this by using a framework like LangChain or by implementing their own orchestration logic.
- **[BONUS]** Self-Learning & Adaptation: Incorporate human feedback to reorganize content, improve clarity, and discover new information sources.

E. Agent for Chat, Planning/Reasoning, and Routing

- Need to write Agent/workflow for **conversational planning, reasoning, agent routing for given conversation**.
- Focus on the **task/subtask-specific prompting, context/memory management, and decision strategies** to achieve robust multi-step reasoning.
- Record observations and iterations to refine prompt design and overall system performance.

F. Available LLM Models for execution

Teams are expected to leverage existing open-source LLMs or available APIs.

- Focus on prompt-based learning: design effective prompts, few-shot prompting, flow specific prompting, and iterative prompting correction adaptation strategies to guide the model to produce high-quality, domain-specific outputs.
- Experiment and learn which method works for the chosen SME domain: identify which prompt structures, example selections, or chaining approaches produce reliable results, and which do not.
- Organize your prompts, few-shot examples, and model outputs systematically for evaluation. Use structured formats such as JSON pairs for easy tracking.
- You may use the same pre-trained model across multiple tasks or select different models based on task performance, inference speed, or model size.
- Document your learning process: highlight successful strategies, failures (list a few scenarios where your chosen model and pipeline fails), and any adaptations made to address these failures for different tasks. Compare performance across approaches and document insights that help improve output quality.

G. RAG : Retrieval and Relevance Ranking Mechanisms

- It is highly recommended to implement a hybrid retrieval system that combines dense vector search with a keyword-based approach (such as BM25) for improved relevance.
- Intelligently assemble the top-K most relevant chunks into a coherent prompt and implement a hybrid scoring mechanism combining vector similarity, reranker scores, and metadata filters.

H. Tool-Using Capabilities

- Knowledge Retrieval: Integrate RAG and search tools for content discovery and feedback.
- Document Generation: Create and export reports in DOCX, PPT, and PDF formats.
- Email Automation: Send generated reports or instructions to specified users.
- Error Handling & Recovery: Implement multi-call recovery to ensure robustness (e.g., if DOCX generation fails, retry with PDF).

I. System Components & Architecture

- Main API Server: An API server is required to handle requests. Frameworks like FastAPI or Flask are suitable choices, and the implementation should include robust validation and asynchronous support where possible.
- Chat/Tool/Agent Servers: This component should manage user-specific chat contexts and histories. Similar for Tools/Agent server.
- A modular RAG pipeline (e.g., retrieve-and-rerank) integrated with above servers.

Guardrails **[Bonus]**

Input/Output Guardrails: Implement robust input sanitization to prevent prompt injection and output moderation to filter harmful content.

Project Deliverables

- Modular and Well-Documented Codebase.
- Data: The complete domain corpus, and tool-task example corpora.
- A video of demo: End-to-end walkthrough of all capabilities of your SME.

- A detailed report including this project brief and a comprehensive write-up of your implementation, design choices, and evaluation results.
    - In your documentation include instructions/steps for setting up, running, and using the system.
    - System Architecture & Workflow Diagram: Explanation of the RAG pipeline and tool-calling layer.
    - API Usage Examples: Clear examples of how to interact with the system's API.
    - *Clearly mention which of the bonuses you have attempted along with detailed explanation.*

Reference:

https://docs.langchain.com/oss/python/langchain/overview

https://docs.langchain.com/oss/python/langchain/rag