# COE548 Final Project: Building a Specialized LLM Agent

## Project Overview

In this project, you will design and develop a specialized Large Language Model (LLM) agent using LangChain and Gemini Pro. The agent will incorporate advanced features like retrieval augmented generation (RAG) and custom tools of your choice, all accessible through a user-friendly Streamlit interface. This project will enhance your skills in AI development, API integration, data handling, and comprehensive documentation. **You have the freedom to determine the primary function and use-case of your LLM agent**, allowing you to explore areas that interest you or address specific problems you find compelling. You will be split into groups of three.

## Objectives

By the end of this project you will be able to:

1. Develop a custom LLM agent: Utilize LangChain and Gemini Pro to create a functional and specialized LLM agent.
2. Integrate multiple tools: Implement Vectorized RAG and at least two custom tools to enhance the agent's capabilities.
3. Design user interfaces: Build an intuitive Streamlit interface for seamless interaction with the LLM agent.
4. Manage APIs: Effectively handle API integrations.
5. Document development processes: Create comprehensive documentation covering system design, tool functionalities, and setup instructions.
6. Demonstrate functionality: Produce a screen-recorded demo showcasing the working project.
7. Apply best practices: Implement proper error handling, type hints, docstrings, and conversation history management.
8. Incorporate ethical considerations: Ensure responsible AI usage by integrating ethical guidelines and safeguards through effective prompting strategies.

## Core Requirements

1. Create a Custom LLM Agent
    a. Use LangChain and Gemini Pro to develop the core LLM agent.
2. Implement at Least 3 Tools
    a. Vectorized RAG (Required): Implement a vectorized RAG mechanism for data handling.
    b. Two or more tools: Choose and implement at least two additional tools from the provided suggestions (one of which must be a customized tool you design).
3. Build a Streamlit Interface
    a. Develop a clean and intuitive Streamlit interface to facilitate user interaction with the LLM agent.

## Final Report Requirements

Your report should be thorough and cover the following sections:

1. Project description: Detailed explanation of what your project does.
2. Prompting strategy:
    a. Describe the style of prompting used.
    b. Explain the intended outcomes of your prompting style.
3. Development process: Discuss the trial and error in prompting that led to the final implementation.
4. System design: Include a system design diagram showing component interactions.
5. Tool documentation: Provide clear documentation for each tool, detailing its purpose and functionality.
6. System design diagram: Create a diagram illustrating how different components of the system interact.
7. Installation and setup instructions: Offer step-by-step instructions to install and set up the project environment.
8. File path instructions: Guide on how to modify file paths for data storage, especially where RAG is utilized (this will be so I know where to look for file path names and change them so I can run them locally wherever needed).
9. API information: List any paid APIs used, including relevant details (leave out the actual API keys).
10. Additional aspects: Cover other significant aspects of your project to provide a comprehensive understanding.
11. Team contribution and fairness: Detail how tasks and responsibilities were divided among each group member, ensuring that contributions are fairly distributed (all team members must have done some work on the code).

## Screen-Recorded Demo

1. Video demonstration: A video showcasing your project in action.
    a. Ensure it clearly demonstrates the main features and functionalities.
    b. Recommended duration: 3-5 minutes (not including any necessary loading/downloading times required to access certain files or data).

## Evaluation Criteria

Your project will be assessed based on the following:

1. Functionality (30%)
    a. Does the project work as intended?
    b. Are all core and additional requirements met?
2. Report Quality (30%)
    a. Is the report comprehensive and well-written?
    b. Does it cover all required sections with sufficient detail?
3. Code Quality (15%)
    a. Is the code well-organized and readable?

    b. Are best practices followed (e.g., type hints, docstrings)?
4. Demonstration (15%)
    a. Is the screen-recorded demo clear and effective in showcasing the project?
    b. Does it highlight the main features and functionalities?
5. Innovation and Creativity (10%)
    a. Does the project demonstrate original thought and creativity?
    b. Are the chosen tools and use cases unique or particularly well-executed?

## Important Notes

1. API Keys Security
    a. DO NOT include actual API keys in any submitted files (this is for your own sake).
    b. Use [insert API key here] as a placeholder in your code and documentation.
    c. Store actual API keys securely using environment variables (e.g., .env files) (recommended).
2. Paid APIs
    a. If your project utilizes paid APIs, clearly mention them in your report.
    b. Provide details such as API name, purpose, and any associated costs.
3. Data Privacy
    a. Ensure that any data used complies with privacy standards.
    b. Avoid including sensitive or personal information in your project.
4. Environment Setup
    a. Use virtual environments to manage dependencies (recommended).
    b. Provide a requirements.txt or similar file for easy installation of dependencies (recommended).
5. Licensing and Attribution
    a. Respect licensing agreements of any third-party tools or libraries used.
    b. Reference any external resources appropriately in your documentation.