**Student Intern: Shivang**

**Project Title**

**"Empirical Evaluation of Multi-Agent Orchestration Frameworks for Complex Task Automation"**

**Objectives**

1. Design and implement multiple AI agents with distinct roles across selected tasks.
2. Integrate these agents using leading open-source orchestration frameworks.
3. Evaluate the orchestration efficiency and agent collaboration using quantitative metrics.
4. Compare frameworks based on performance, modularity, scalability, and adaptability.

**Tasks for Agent Evaluation**

Design agents to collaborate on these **3–4 benchmark tasks**:

| **Task** | **Description** | **Example Sub-Agents** |
| --- | --- | --- |
| **Multi-Doc Summarization** | Summarize insights from 3–5 documents | Retriever, Summarizer, Evaluator |
| **Python Code Generation** | Generate a Python script from a user requirement | Analyst, Planner, Coder, Tester |
| **Web Research Agent** | Find best tools for a given topic | Searcher, Ranker, Note-Taker |
| **Report Writing** | Generate a structured report from mixed inputs | Data Collector, Writer, Formatter |

**Open-Source Tools for Agent Orchestration**

Use minimum **3 orchestration frameworks** to compare: **(ADD MORE if time permits)**

| **Framework** | **Notes** |
| --- | --- |
| **LangChain Agents** | Most flexible, with agent/tool abstractions |
| **CrewAI** | Intuitive role-based agent assignment |
| **AutoGen** (Microsoft) | High-level orchestration, strong LLM-to-LLM chat |

**Quantitative Evaluation Metrics**

| **Category** | **Metric** | **Description** |
| --- | --- | --- |
| **Accuracy** | Task Success Rate | % of tasks completed correctly |
| **Efficiency** | Execution Time | Time from start to end of orchestration |
| **Scalability** | Agent Load Time | Time/resource scaling with #agents |
| **Coherence** | Output Consistency | Evaluated using GPT-based scoring or ROUGE/BLEU |
| **Cost** | API Calls / Tokens | Total usage per task execution |

**Tech Stack**

| **Component** | **Stack** |
| --- | --- |
| LLM Backend | OpenAI GPT-4 (via API), Mistral 7B (via Ollama) |
| Agent Frameworks | LangChain, CrewAI, AutoGen |
| Embedding & Search | FAISS / ChromaDB, HuggingFace Transformers |
| Evaluation | ROUGE / BLEU / GPT-4-based Evaluator |
| Backend | Python, FastAPI (for APIs if needed) |
| Logging/Tracking | LangSmith, WandB (optional) |
| Versioning | GitHub + Git |
| IDE | VS Code / Jupyter Notebooks |

**Step-by-Step Implementation Plan**

**Step 1: Task Design and Data Preparation**

* Choose/define 3–4 tasks
* Gather datasets or simulate inputs (e.g., document sets, user prompts)

**Step 2: Agent Design**

* Define roles for each task (e.g., "researcher", "summarizer", etc.)
* Create modular agents using LangChain Agent API / CrewAI roles / AutoGen Agent configuration
* Add tool interfaces (search, code execution, summarization)

**Step 3: Orchestration Framework Setup**

* Create equivalent agent teams in:
  + LangChain using initialize\_agent()
  + CrewAI using Crew and Task classes
  + AutoGen using AssistantAgent and GroupChatManager

**Step 4: Execution & Logging**

* Execute the same task across all frameworks
* Log execution time, success, agent messages, token usage

**Step 5: Evaluation**

* Use:
  + Task Success Rate (manual or rule-based)
  + Output Quality (e.g., ROUGE/BLEU for text tasks)
  + Time and token cost
* Use GPT-4 as a "judge" for qualitative scoring if applicable

**Step 6: Analysis & Reporting**

* Compare results framework-wise:
  + Performance
  + Ease of use
  + Agent flexibility
* Visualize results (e.g., bar charts, spider plots)