AI Agent-Based Deep Research System

Submission for Kairon.ai Assignment
Author: Prajakta Gurav
Date: April 25, 2025

1. Project Overview

This project presents a dual-agent AI system that automates deep online research using LangChain, LangGraph, and Tavily API. The system comprises two independent but collaborative agents: one responsible for performing comprehensive information retrieval across the web and another dedicated to synthesizing the results into coherent, human-readable answers.

The goal is to simulate the function of a human research assistant capable of collecting information, reasoning over the content, and drafting answers with clarity.

2. Objectives

- Automate in-depth online research using Tavily.
- Employ a dual-agent architecture for separation of concerns.
- Utilize LangGraph to structure the flow between agents.
- Leverage LangChain for orchestration, prompt engineering, and integration.
- Ensure modularity, reusability, and clarity in design.

3. System Architecture

Agent Roles:

- **Research Agent**: Fetches and compiles raw data from multiple web sources using the Tavily Search API.
- **Answer Drafting Agent**: Processes the research findings and constructs detailed, conversational summaries or answers.

Flow:

1. Input a user query.
2. The Research Agent uses Tavily to collect search results and condense the findings.
3. The summarized findings are passed to the Answer Drafting Agent.
4. The final draft is returned as a structured response.

Technologies:

- **LangChain**: Prompt handling, memory management, agent interface
- **LangGraph**: Agent-to-agent workflow logic
- **Tavily API**: Web data retrieval
- **Python (Colab)**: Execution environment

4. Implementation Details

- **Agent Tools**: Built with langchain.agents and custom toolkits for Tavily and LLM-based summarization.
- **Prompt Strategy**: Employed few-shot examples and role-based prompts for answer generation.
- **Error Handling**: Integrated fallback messages and quota awareness checks for API limits.
- **Reusability**: Functions modularized for easy scaling or integration with new tools/APIs.
- **Security**: Removed all API keys from the public repository; used .env locally during development.

5. Unique Features

- Modular dual-agent design with flexible integration paths.
- Readable code with educational inline comments for clarity.
- Independent debugging blocks for testing each agent.
- Option to switch between summarization styles (detailed, concise, bullet-point).

6. Challenges Faced

- Tavily API quota limits led to temporary failures, handled using retries and fallback prompts.
- LangGraph documentation required careful review to model the dual-agent flow correctly.
- Ensuring clean and modular code with comments while balancing functionality.

7. How to Run

1. Clone the GitHub repository.
2. Install dependencies: pip install -r requirements.txt
3. Set up environment variables (API keys) in .env
4. Run the notebook in Google Colab or locally via Jupyter.

8. Future Enhancements

- Add a third agent for critical evaluation or fact-checking.
- Integrate semantic search using vector databases.
- Convert to a web-based UI using Streamlit or Gradio.
- Add support for multiple query inputs for comparative research.