OpenDeepResearcher: Agentic LLM Research Framework

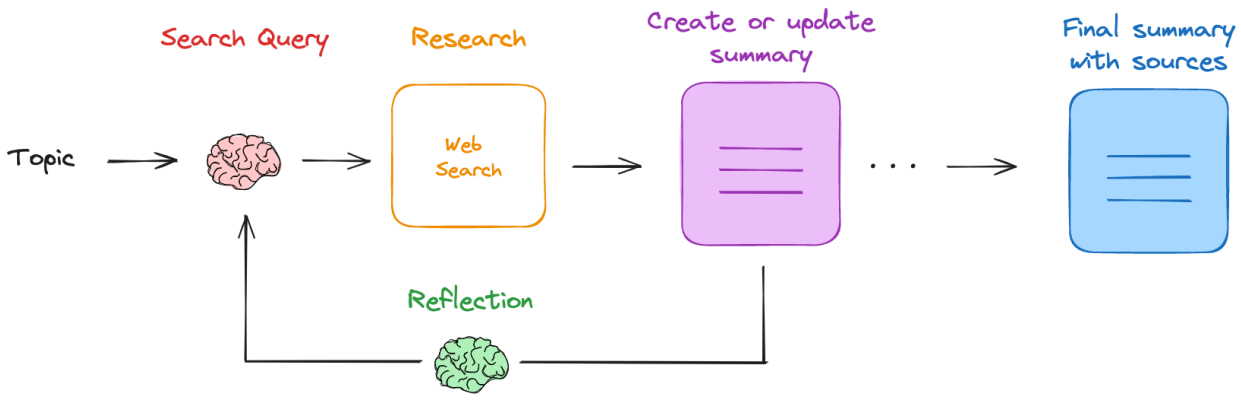
Objective

The objective of the Open Deep Researcher project is to develop an AI-powered research assistant capable of autonomously conducting in-depth explorations of complex topics by leveraging large language models (LLMs), agentic workflows, and web search APIs. The system aims to streamline the research process by automatically planning, retrieving, analyzing, and summarizing information from credible sources. By simulating the behavior of a skilled human researcher, the tool enables users to generate high-quality, multi-perspective reports efficiently and cost-effectively.

Project Workflow

- **Planning**: A Planner Agent breaks the prompt into sub-questions and creates a research roadmap.
- **Information Retrieval**: A Searcher Agent queries the web (e.g., via Tavily) to collect relevant content.
- **Synthesis**: A Writer Agent analyzes and summarizes the retrieved data using an LLM.
- **Report Generation**: The system compiles all findings into a structured research report.
- **Session Memory (Optional)**: Research threads are stored for continuity and future reference

Architecture Diagram



Components

- **Planner Agent**: Breaks down the main query into sub-questions and creates a research plan.
- **Searcher Agent**: Uses web APIs (e.g., Tavily) to retrieve relevant, up-to-date content.
- **Writer Agent**: Synthesizes retrieved data into structured, coherent summaries using an LLM.
- **Memory System**: Stores session data to support iterative and multi-step research.
- **Execution Graph**: Manages the flow between agents using LangGraph.
- **Model Interface**: Allows integration with local or hosted LLMs (e.g., via LM Studio).

Tech Stack

The project is built using a clean and efficient stack focused on agentic workflows, local LLM inference, and structured research synthesis.

Programming Language

- Python — Core language used for building all components and logic.

Core Libraries & Frameworks

- LangGraph — For defining and executing multi-agent workflows.
- LangChain — Used for LLM integration, memory handling, and tool coordination.

LLM Integration

- LM Studio/Ollama/API — Runs local language models and provides an OpenAI-compatible API interface.
- Qwen2.5-7B-Instruct or any other — Instruction-tuned LLM used for both planning and writing tasks.

Web Search Integration

- Tavily API — Enables real-time retrieval of relevant research content from the web.

Session & Memory Management

- MemorySaver — Tracks research threads and enables continuity across multi-step research tasks.

Environment & Tooling

- Python 3.10+
- pip / venv — For dependency and environment management.
- Git — For version control.

Detailed Plan

The project is divided into four milestones, each spanning two weeks. Evaluations are scheduled at the end of each milestone to review progress and ensure alignment with objectives.

Milestone 1: Weeks 1–2 – Foundation Setup

- Set up the development environment
    - Python environment and virtualenv
    - Install required dependencies
- Design project structure and architecture
    - Define agent responsibilities and data flow
- Integrate local LLM via LM Studio
- Configure external services (e.g., Tavily)

Evaluation – Week 2

- Functional development setup
- Architecture drafted and discussed
- Stubbed research flow returns basic output

Milestone 2: Weeks 3–4 – Core Agent Development

- Build functional agents
    - Planner Agent: break down main topic
    - Searcher Agent: fetch content using Tavily

    - Writer Agent: synthesize responses using LLM
- Implement execution pipeline using LangGraph
- Validate basic research loop from input to summary

Evaluation – Week 4

- Research flow executes successfully
- Agents return relevant, coherent responses
- LangGraph pipeline operates end-to-end

Milestone 3: Weeks 5–6 – UI and Memory Integration

- Build user interface
    - Clean, ChatGPT-like design
    - Input prompt and result display
- Connect UI to backend agent pipeline
- Implement session memory
    - Thread tracking and continuity support

Evaluation – Week 6

- UI integrated and interactive
- Research session memory functional
- Midpoint testing conducted

Milestone 4: Weeks 7–8 – Refinement and Final Output

- Improve agent coordination and prompt tuning
- Structure final output
    - Organized sections
    - Cited sources and timestamps
- Optimize performance and response quality
- Finalize documentation and prepare for demo

Evaluation – Week 8

- Full report generation with structure and citations
- Clean, reliable system behavior
- Documentation and demo completed