DeepResearchAgent Implementation Document

Overview: The DeepResearchAgent project is designed to automate research and answer drafting using a modular, agent-based workflow. It leverages LangGraph for workflow orchestration, Tavily for web search, and Hugging Face for answer generation. The system is structured to be extensible and easy to maintain.

PROJECT STRUCTURE:

- *main.py*: Entry point and workflow definition.
- agents/
  - *research_agent.py*: Defines the research agent for web search.
  - *answer_drafter_agent.py*: Defines the answer drafting agent.
  - *answer_drafter_agent_QA.py*: (Optional) For QA-specific answer drafting.
- requirements.txt: Python dependencies.
- *streamlit_app.py*: (Optional) Streamlit UI for the agent.
- *readme.md*: Project documentation.

WORKFLOW DESCRIPTION:

1. User provides a research query.
2. The research agent performs a web search using Tavily and returns context.
3. The answer drafter agent summarises the context using a Hugging Face model.
4. The final answer is presented to the user.

KEY COMPONENTS:

1. *main.py*
   - Loads environment variables.
   - Defines a shared state (GraphState) with query, context, and final_answer.
   - Implements two nodes: a. research_node: Uses create_research_agent() to perform a web search. b. answer_node: Uses generate_answer() to summarise the context.
   - Sets up a LangGraph workflow with research and answer nodes.
   - Entry point: Prompts user for a query and prints the final answer.
2. *research_agent.py*
   - Contains create_research_agent(), which initialises and returns a research agent capable of running web searches.
3. *answer_drafter_agent.py*
   - Contains generate_answer(), which takes context (and optionally a query) and returns a summarised answer using a Hugging Face model.

EXTENSIBILITY:

- Additional nodes or agents can be added to the workflow for more complex pipelines.
- The answer drafter can be swapped for a QA-specific version if needed.

DEPENDENCIES:

- langgraph
- python-dotenv
- Tavily API (for web search)
- Hugging Face Transformers (for answer generation)
- streamlit (optional, for UI)

USAGE:

1. Install dependencies from requirements.txt.
2. Set up environment variables as needed (API keys, etc.).
3. Run main.py and enter a research question when prompted.

4. The system will output a summarised answer.

This document provides a high-level overview and implementation details for the DeepResearchAgent codebase. For further details, refer to the code and *readme.md.*