LangGraph Multi-Agent Coding Exercise: Research Assistant

Problem Statement

Build a multi-agent research assistant using LangGraph that helps users gather information about companies. The system should have multiple specialized agents working together, support follow-up questions, and request human clarification when queries are ambiguous.

Requirements

1. Agent Architecture

Implement **4 specialized agents**:

a) Clarity Agent

- Analyzes if the user's query is clear and specific
- Checks if company name is mentioned or if query is too vague
- **OUTPUT**: Sets `clarity_status` to "clear" or "needs_clarification"
- **ROUTES TO**: Interrupt (if unclear) OR Research Agent (if clear)

b) Research Agent

- Searches for company information (news, financials, recent developments)
- This agent should derive this information from a search tool (Tavily MCP would be preferred) or you can use the mock data below.
- **OUTPUT**: Returns research findings and assigns a `confidence_score` (0-10)
- **ROUTES TO**: Validator Agent (if confidence < 6) OR Synthesis Agent (if confidence ≥ 6)

c) Validator Agent

- Reviews research quality and completeness
- Checks if information is sufficient to answer the user's question
- **OUTPUT**: Sets `validation_result` to "sufficient" or "insufficient"
- **ROUTES TO**:
  - Research Agent (loop back if insufficient AND attempts < 3)
  - Synthesis Agent (if sufficient OR max attempts reached)

d) Synthesis Agent

- Takes research findings and creates a coherent summary
- Formats the response in a user-friendly way
- Maintains context from conversation history
- **ROUTES TO**: END

2. Core Features to Implement

Multi-turn Conversation

- Maintain conversation history across multiple queries
- Each agent should access previous messages for context
- Support follow-up questions like "What about their competitors?" or "Tell me more about the CEO"

Human-in-the-Loop (Interrupt)

- When Clarity Agent detects an unclear query, interrupt the workflow
- Request clarification from the user (e.g., "Which company are you asking about?")
- Resume processing after receiving clarification

State Management

- Use a proper state schema that includes

Conditional Routing

- Appropriate routing across the CLarity, Research & Validation agents

```
None

```

3. Mock Data (You Can Use)

Since you don't need real API calls, mock your research with sample data like:

```python
mock_research = {
    "Apple Inc.": {
        "recent_news": "Launched Vision Pro, expanding services
revenue",
        "stock_info": "Trading at $195, up 45% YTD",
        "key_developments": "AI integration across product line"
    },
    "Tesla": {
        "recent_news": "Cybertruck deliveries ramping up",
        "stock_info": "Trading at $242, volatile quarter",
        "key_developments": "FSD v12 rollout, energy storage
growth"
    }
}
```

3. Deliverable

A zipped code repo that demonstrates, at minimum

1. A working LangGraph with 4 agents
2. State schema definition with all required fields
3. 3 conditional routing functions properly implemented
4. Feedback loop from Validator back to Research with attempt counter
5. Interrupt mechanism for unclear queries
6. Multi-turn conversation handling with memory
7. At least 2 example conversation turns showing the loop in action
8. Showcase software engineering best practices (classes, repo structure, formatting, etc)
9. Instructions on how to run the agentic system in Readme.md
10. This is an open ended problem, so if you have any confusion then make reasonable assumptions and state those in the Readme.md
11. Anything that can be implemented beyond the above would get a higher rating. Please call that out clearly in Readme.md in a section titled "Beyond Expected Deliverable”