ATG Technical Assignment: Multi-Agent Debate DAG using LangGraph

Machine Learning Intern Deliverables

Task Overview

Construct a debate simulation system using LangGraph where two AI agents engage in a structured argument over a fixed topic. The system should use memory, control over turns, and a judging node that evaluates the debate logically and declares a winner.

Deadline: 48 hours from task assignment

Objective

Design and implement a LangGraph-based workflow with two alternating agents debating a fixed topic. The system should preserve and summarize the dialogue history, validate logical progression, and end with an automated judgment using a dedicated node.

Workflow Nodes

- **UserInputNode:** Accepts the debate topic at runtime from the user.
- **AgentA and AgentB:** Two profession-specific personas (e.g., Scientist vs Philosopher) that take alternate turns making arguments.
- **Rounds:** The debate must consist of exactly 8 rounds — 4 arguments per agent, alternating.
- **MemoryNode:** Stores and updates a structured summary or full transcript of arguments. Each agent only receives relevant memory (no full state sharing).
- **JudgeNode:**
  - Reviews memory and all argument nodes
  - Produces a full debate summary
  - Declares a winner with logical justification

Additional Requirements

- All node messages and responses must be logged to a file, including the final judgment.
- Implement state validation to ensure:
  - Each agent only speaks in their assigned turn
  - No argument is repeated
  - Logical coherence is maintained across the flow
- The entire debate must operate via a clean CLI interface.
- Include a DAG diagram (either static using `graphviz`, or generated programmatically via LangGraph tools).
- Submit a full log of all state transitions, memory content, and final verdict.

Expected Output

- CLI-based interaction:

```
Enter topic for debate: Should AI be regulated like medicine?

Starting debate between Scientist and Philosopher...
[Round 1] Scientist: AI must be regulated due to high-risk applications.
[Round 2] Philosopher: Regulation could stifle philosophical progress and autonomy.
...
[Round 8] Philosopher: History shows overregulation often delays societal evolution.

[Judge] Summary of debate:
...

[Judge] Winner: Scientist
Reason: Presented more grounded, risk-based arguments aligned with public safety princi
```

Deliverables

1. **Source Code**: Modular files for all LangGraph node definitions and execution logic.
2. **README.md**:
   - How to run the program and install dependencies
   - Node and DAG structure explained

3. **DAG Diagram**: Visual layout of the LangGraph architecture (static image or auto-generated).
4. **Chat Log File**: Full log of all messages, transitions, memory updates, and final verdict.
5. **Demo Video**: 2–4 minute walkthrough (screen + face-cam) explaining:
    - Project structure and key files
    - CLI flow
    - Judgment process

Submission Format

- GitHub repository or zipped folder containing:
    - All source code
    - README.md
    - DAG diagram
    - Log file
    - Demo video (or shareable link)

Evaluation Criteria

- Correctness and functionality of LangGraph DAG
- Debate round control and memory handling
- Judge logic and decision explanation
- Code quality and documentation
- Communication and clarity in the demo video