🎭 Multi-Agent Collaborative Reasoning System

1. Introduction

This project implements a multi-agent system where distinct AI personas; the Commander, Rationalist, and Dramatist, all engage in structured, autonomous dialogue to collaboratively reason through complex user queries.

Their conversation is orchestrated using LangGraph, which ensures dynamic turn-taking, agent-to-agent communication, and multi-stage reasoning leading to a final unified synthesis by a fourth agent, the Synthesizer.

The purpose of this system is twofold:

1. To demonstrate agent collaboration with persistent state and communication.
2. To maintain traceable citations and evidence-based reasoning from each agent's dedicated knowledge base.

This document explains:

- How the dataset was preprocessed and split into persona-specific knowledge bases.
- How the personas were designed and justification for the same.
- The architecture of inter-agent collaboration (including LangGraph state flow).
- The reasoning behind code design choices in each module.
- Future improvements for scaling collaboration, speed, and transparency.

2. Dataset and Preprocessing

2.1 Source Data

The base dataset consisted of movie and dialogue transcripts. Each line was tagged with metadata such as:

- Movie ID
- Character name
- Line ID
- Dialogue text

This format allows to analyse speaking styles, emotional intensity, and rational density of language.

2.2 Preprocessing Steps

Preprocessing scripts were written to:

1. Clean text: Removed special characters, empty lines, and one-word interjections.
2. Normalize casing: Ensured consistent lowercasing except for named entities.
3. Filter: Discarded lines with fewer than 5 tokens or more than 220 characters to maintain semantic balance.
4. Encode and index:
   - Each line was embedded using the all-MiniLM-L6-v2 SentenceTransformer model.
   - Embeddings were normalized and stored in FAISS indexes for each persona.

2.3 Persona Segmentation

Dialogues were analysed through semantic clustering (via embedding space) and linguistic tone profiling:

- Lines rich in verbs, imperatives, and deadlines clustered as decisive/action-oriented → Commander.
- Lines emphasizing logic, data, or hypotheticals clustered as analytical → Rationalist.
- Lines heavy on imagery or emotion clustered as reflective or empathetic → Dramatist.

This clustering became the foundation for persona-specific knowledge bases stored under:

data/processed/personas/commander/
data/processed/personas/rationalist/
data/processed/personas/dramatist/

Each folder contains:

- A FAISS index (.faiss)
- A metadata file (.meta.jsonl)

During inference, each agent retrieves relevant snippets only from its own knowledge base, maintaining boundaries while allowing communication learning from others via dialogue.

3. Persona Design and Justification

Persona	Core Function	Communication Style	Grounded In
Commander	Converts insights into decisive actions and checkpoints.	Direct, assertive, goal-driven.	Operational language in film dialogue.
Rationalist	Challenges assumptions and seeks falsifiable logic.	Calm, objective, hypothesis-based.	Analytical reasoning lines and debates.
Dramatist	Reveals emotional stakes and human motivations.	Expressive, metaphorical, empathetic.	Emotionally dense, reflective dialogues.
Synthesizer	Integrates all viewpoints into a unified insight.	Balanced, explanatory, summarising.	Built-in reasoning (no external retrieval).

Rationale

Personas needed to be distinct (different knowledge bases) and methodologically complementary:

- The Commander grounds the conversation in action.
- The Rationalist brings logic and verification.
- The Dramatist injects context and meaning.

Together they mimic how multidisciplinary teams collaborate, balancing fact, strategy, and emotion.

4. Collaboration Architecture

4.1 LangGraph Flow

LangGraph was used to manage dialogue phases and transitions.

The system runs as a state machine with the following flow:

init_state_node → round1_node → dialogue_node → challenges_node → synthesis_node

Each state function updates the shared GraphState dictionary:

- round1: Each agent gives its first independent answer.
- dialogue: Agents converse dynamically, taking turns or tagging each other using @mentions.
- challenges: The Rationalist challenges others, Commander rebuts, Dramatist reconciles.
- synthesis: Synthesizer integrates all responses into one actionable output.

The router node automatically directs transitions based on the phase key in the state.

4.2 Inter-Agent Communication

Agents communicate autonomously:

- @Commander, @Rationalist, or @Dramatist triggers targeted replies in subsequent turns.
- If no @mention is used, LangGraph cycles through agents in a fixed rotation (ROLE_ORDER).

This ensures a natural yet controlled debate flow.

4.3 State Management

The GraphState object tracks:

- The user query
- Dialogue history (thread)
- Pending turn targets (pending_target)
- Remaining turns
- Each phase's intermediate results (round1, dialogue, challenges, synthesis)

This modular design makes LangGraph easily extendable to additional personas or reasoning phases.

5. Citation and Evidence System

5.1 Retrieval Layer

Each agent has its own PersonaRetriever, which:

- Loads its FAISS index and metadata file.
- Filters retrieved text for minimum length and informativeness.
- Re-ranks results using MMR (Maximal Marginal Relevance) for diversity.
- Returns the top k snippets with associated metadata.

5.2 Citation Formatting

The citations.py module ensures consistency.

Each retrieved line is displayed as:

[Character, line_id, movie_id]

or in concise text form:

John · line 54 · movie 102

These appear below each agent's bubble as 🔗 citations, making reasoning transparent and verifiable.

6. User Interface (Streamlit Frontend)

6.1 Design Philosophy

The frontend aims to make reasoning visual and human, not purely textual.

Hence, a color-coded bubble chat interface was implemented with typing animation.

Agent	Colour	Symbol
User	Grey	👨
Commander	Blue	🧭
Rationalist	Green	🧠
Dramatist	Purple	🎭
Synthesiser	Teal	🧩

6.2 Streamlit Deployment Notes

To make the project easily accessible for evaluation, a Streamlit app version of the Multi-Agent Collaboration System was deployed on Streamlit Community Cloud.

The deployed version allows users to input a query and observe simulated dialogue between the Commander, Rationalist, and Dramatist agents, along with a synthesized consensus. However, since the local implementation uses Ollama for local LLM inference, the hosted version operates with limited capability due to the following constraints:

- Ollama models cannot be accessed from Streamlit Community Cloud, as they require a local backend for inference.
- As a result, the deployed version relies on a simplified fallback text generation process to demonstrate the interface flow rather than the full agent reasoning pipeline.
- While the conversation structure, UI, and workflow execution (LangGraph pipeline) remain intact, the agent responses are shorter and less nuanced compared to the locally executed version.

The local implementation (as documented in the GitHub repository) provides the full experience with:

- Multi-turn reasoning between agents
- Persona-specific dialogue dynamics
- Full synthesis and citation tracing

The Streamlit version serves as a functional demo for visualization and user interaction, showing how the conversation pipeline would appear in a hosted environment.

Streamlit App (Demo): https://vectorial-multiagent-svxs8zkkzccdmxhs3orb9m.streamlit.app/

6.3 Key Features

- Animated typing output for realism.
- Color-coded dialogue bubbles for clarity.
- Sectional display:
    - 🗣️ Round 1 – Individual perspectives
    - 💬 Dialogue – Agent collaboration
    - ⚖️ Challenge – Debates & rebuttals
    - 🧩 Synthesis – Unified decision
- Clear History button to reset sessions.

6.4 Performance Enhancements

The warm_up() function preloads the model to prevent first-call delays.

Parallel processing (via ThreadPoolExecutor) speeds up Round 1 responses.

7. Setup and Running Instructions

7.1 Environment Setup

```
git clone https://github.com/<your-repo>/multiagent-collab
cd multiagent-collab
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

7.2 Running the System

```
streamlit run app.py
```

7.3 Data Index Setup

If FAISS indices are not included, run:

```
python scripts/build_indices.py
```

This script reads the raw dialogue dataset and outputs persona-specific FAISS files.

8. Usage Guide

1. Launch the Streamlit app.
2. Enter a query (e.g., “How should I handle conflict at work?”).
3. Observe:
   - Round 1: Each agent responds independently.
   - Dialogue: They debate, tag each other, and build context.
   - Challenge Round: Rationalist challenges others, Commander rebuts, Dramatist reconciles.
   - Synthesis: A final unified insight is presented.
4. Hover over 🔗 citations to see evidence sources.
5. Click “Clear History” to restart.

9. Limitations and Future Work

9.1 Current Limitations

- Latency: Each LLM call runs sequentially within dialogue rounds. Larger LangGraphs with more nodes increase runtime.
- Retrieval Simplification: Current FAISS retrieval uses only text-based heuristics; no dynamic query reformulation yet.
- Knowledge Isolation: Agents rely only on their own retrievers, so true cross-knowledge fusion happens only at the Synthesizer stage.
- Fixed Personas: Current system uses static prompt templates; personality traits are not dynamically adaptive.

9.2 Future Enhancements

- Add Adaptive Persona Modulation using reinforcement learning.
- Incorporate graph visualization of conversations within the UI.
- Replace static FAISS with vector database APIs (e.g., Pinecone, Weaviate).
- Extend to more complex collaboration scenarios (e.g., 5–6 agents or adversarial debate).

10. Testing and Validation

10.1 Unit Tests

Lightweight tests were included for:

- FAISS retrieval quality (retriever_test.py)
- Agent response formatting (agent_test.py)
- LangGraph state transitions (graph_test.py)

Run tests with:

pytest tests/

10.2 Manual Validation

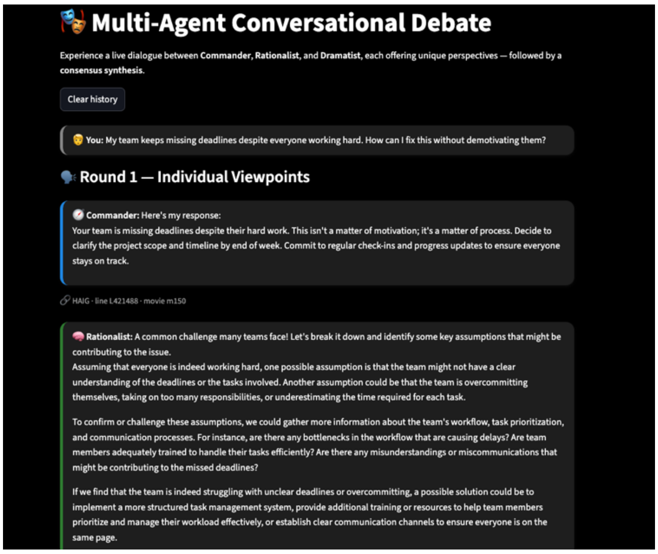
Example prompts:

- “How should a leader handle team burnout?”
- “When logic and emotion conflict, what should guide decisions?”
- “Is it better to prioritize consistency or creativity in management?”

Each of these triggers a 4-phase collaboration where agents display distinct reasoning, validate evidence, and reach a reconciled output.

11. Optional Demo Artifacts

Screenshot Examples:



⚖️ Challenge Round — Agents Respond to Each Other

🧠 Rationalist → 🧭 Commander

I appreciate the proposal. However, I'd like to challenge the assumption that "hard work" is directly correlated with meeting deadlines. It's possible that the team is indeed working hard, but their process or approach might not be effective in achieving the desired outcomes.
To test this assumption, we could observe the team's workflow and identify any potential bottlenecks or inefficiencies that might be contributing to the missed deadlines. This could involve conducting a brief workflow analysis or gathering feedback from the team members themselves.

If our observation confirms that the team is indeed working hard but still struggling to meet deadlines, a fallback approach could be to focus on process improvements rather than simply relying on increased effort. This might involve implementing new tools, streamlining tasks, or providing additional training to help the team stay on track.

🔗 HANS · line L165327 · movie m323

🧠 Rationalist → 🎭 Dramatist

I appreciate the creative analogy of the ticking clock and its implications for team dynamics. However, I'd like to challenge the assumption that "hard work is not translating into timely results" is the primary issue.
To test this assumption, we could observe the team's workflow and task prioritization to see if there are any inefficiencies or bottlenecks that might be contributing to the missed deadlines. Are there any tasks that can be delegated or optimized to improve overall productivity?

If this assumption holds true, and the team is indeed struggling with inefficiencies, then reframing the problem as one of shared responsibility and exploring workflow improvements would be a great approach. However, if the assumption fails, and the team is simply facing an unusual surge in complexity or unexpected challenges, we might need to consider alternative solutions, such as providing additional resources or adjusting expectations.

🔗 JACK · line L148794 · movie m46

🧩 Consensus Summary — Unified Insight

🧩 **Synthesis:** Here's a synthesis of the three perspectives:
While it's true that your team is missing deadlines despite their hard work, it's essential to recognize that this issue is not solely a matter of motivation. Instead, it's likely a combination of factors, including unclear project scope and timeline, overcommitting, and potential bottlenecks in the workflow.

To address this without demotivating your team, consider reframing the problem from one of blame to one of shared responsibility. By working together to identify the root causes and implementing targeted solutions, you can help the team stay motivated and productive while still meeting their deadlines.

Next steps:

- Clarify the project scope and timeline by end of week
- Commit to regular check-ins and progress updates to ensure everyone stays on track
- Gather more data by tracking the team's workflow and task management processes

🔗 inline (reused from personas when present)

🧠 **Rationalist:** I understand the concern. To hit critical milestones without burning out the team, let's first assume that the hiring freeze is indeed a significant constraint. This assumption would be true if the team's current workload and responsibilities are already overwhelming, and the absence of new hires would exacerbate the situation.
To confirm or challenge this assumption, we could gather data on the team's current workload, their capacity to take on additional tasks, and the impact of the hiring freeze on their morale and productivity. For instance, we might analyze the team's task completion rates, employee satisfaction surveys, and project timelines before and after the hiring freeze.

If the assumption holds true, then it's crucial to prioritize tasks, delegate responsibilities effectively, and provide necessary support and resources to the existing team members. This could involve reassigning tasks, providing additional training or mentorship, and offering flexible work arrangements to help manage workload and stress.

On the other hand, if the assumption is false, and the team is capable of adapting to the hiring freeze without significant burnout, then we might focus on identifying and addressing any underlying systemic issues that are driving the perceived need for new hires.

In either case, a conditional principle could be: If the team's workload and responsibilities can be managed effectively with the current team members, then prioritize task delegation and support; otherwise, explore alternative solutions, such as temporary contractors or project adjustments, to ensure the project's success without compromising team well-being.

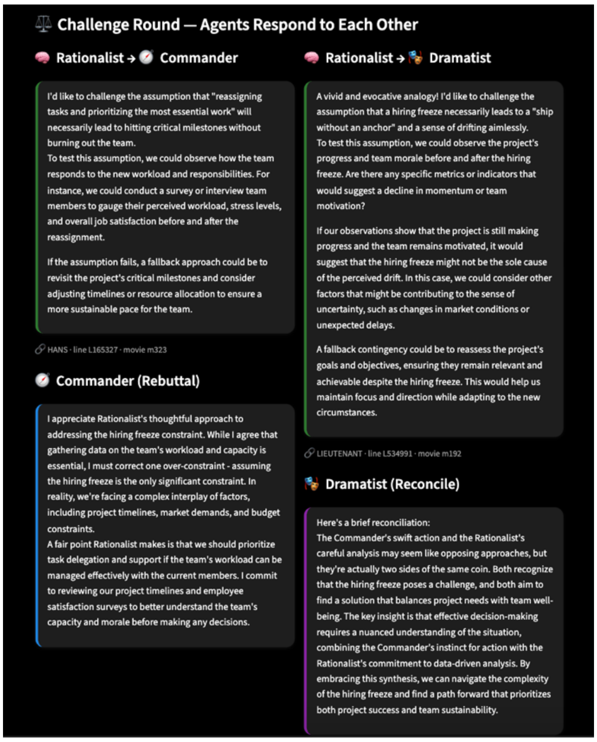
🔗 ROYCE · line L241605 · movie m66

🎭 **Dramatist:** A hiring freeze, a sudden stillness in the midst of momentum. Your project's trajectory is like a ship without an anchor, drifting aimlessly on the open sea.
This image reveals the tension between the need for progress and the fear of being stuck. You're torn between the desire to reach critical milestones and the concern that your team will burn out under the weight of uncertainty.

To navigate this treacherous landscape, I suggest you focus on the present moment, like a sailor charting a course through uncharted waters. Break down the project into manageable tasks, and prioritize the most critical ones. This will help you maintain momentum without overwhelming your team.

Remember, the key is to find a balance between progress and preservation. By doing so, you'll not only reach your milestones but also ensure that your team remains energized and motivated throughout the journey.

🔗 LIANE · line L304142 · movie m101



12. Conclusion

This project demonstrates that multi-agent collaboration, when structured via LangGraph and supported by persona-specific retrieval, can yield nuanced, transparent reasoning.

Each agent brings a distinct lens, tactical, analytical, and emotional, while maintaining dialogue coherence and evidence traceability.

The result is a modular, explainable system that models human-like collaborative problem solving in AI.