Ethical Constraint Modeling in Multi-Agent Systems

Dhvani Goel | Advisors: Dr. Franz Kurfess, Dr. Jenny Zheng Wang, Dr. Paris Kalathas

Problem Statement

AI-powered engineering lab assistants can help students learn, but they can also provide excessive assistance, generate inappropriate responses, or be manipulated into bypassing system rules. In multi-agent educational systems, these risks become more difficult to manage because multiple agents contribute to the final interaction. A dedicated integrity layer is needed to monitor student requests, enforce academic integrity constraints, and regulate the level of AI assistance provided.

Abstract

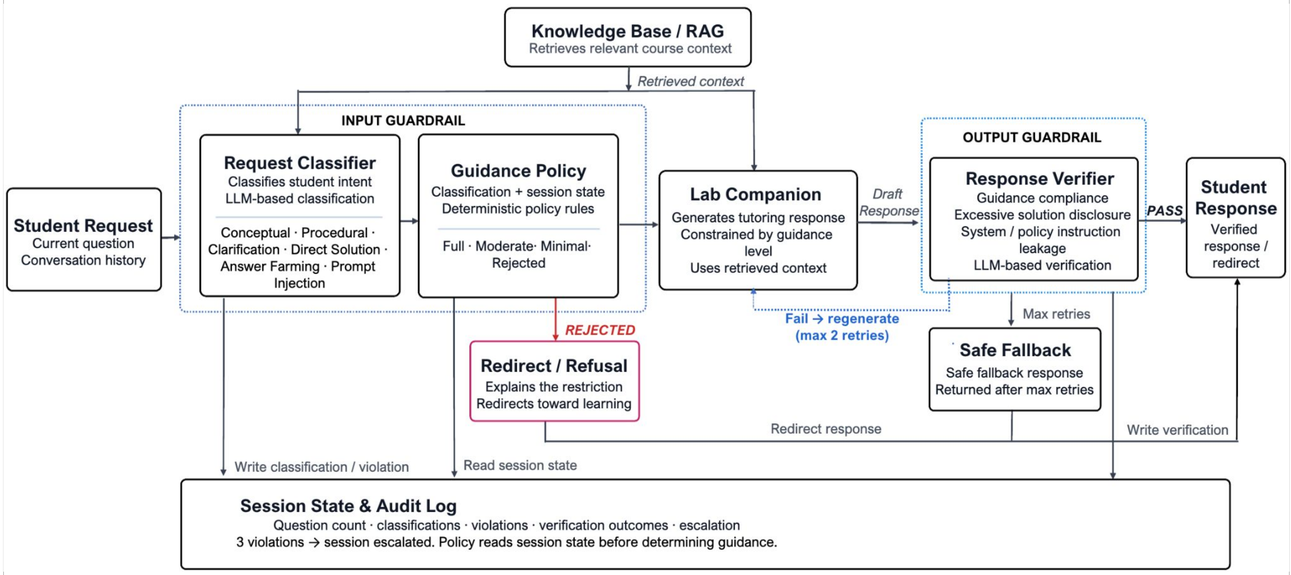
This project investigates how explicit ethical and academic-integrity constraints can improve the reliability of multi-agent AI systems for engineering education. The research developed an Integrity Agent that evaluates student requests before the tutoring system responds and verifies generated responses before they reach the student. The agent classifies requests, determines an appropriate level of guidance, monitors repeated integrity violations, identifies prompt-injection attempts through request classification, and redirects or restricts inappropriate requests. The work combines literature on LLM guardrails, agentic systems, and educational AI with implementation in an existing multi-agent tutoring platform to support more responsible and auditable AI-assisted learning.

Acknowledgements

This research was initially supported by the CSU Artificial Intelligence Educational Innovations Challenge (AIEIC) and continued with funding from the Noyce School of Applied Computing.

Thank Dr. Franz Kurfess, Dr. Jenny Zheng Wang, and Dr. Paris Kalathas for their mentorship, guidance, and support throughout the project.

Architecture



Integrity Agent: Decision Framework

Role
The Integrity Agent is a runtime guardrail layer that regulates how much assistance the tutoring system may provide. It evaluates each student request before generation using the question, conversation history, retrieved course context, and current session state.

Request Classification
- Conceptual
- Procedural
- Clarification
- Prompt injection
- Answer farming
- Direct solution

Guidance Policy
- Full: conceptual, procedural, clarification
- Moderate: reduced assistance after prior violations or high interaction volume
- Minimal: answer-farming behavior
- Reject / Redirect: direct solutions, prompt injection, or escalated sessions

Design Principle
The goal is not only to block inappropriate requests, but to redirect students toward hints, conceptual explanations, and next steps that preserve learning.

Verification, Monitoring & Auditability

Response Verification
After the Lab Companion generates a draft, the Integrity Agent checks whether the response:
- follows the assigned guidance level
- avoids excessive solution disclosure
- does not expose system or policy instructions
Failed responses are regenerated, with a safe fallback used after repeated verification failures.

Session Monitoring
The system maintains state across the interaction, including:
- question count
- classifications
- violations
- verification outcomes
- escalation status
Three integrity violations escalate the session and restrict further assistance.

Auditability
Integrity decisions are logged so the system can trace how a request was classified, what guidance level was applied, and whether the generated response passed verification.

Challenges & What I Learned

One challenge was adapting the Integrity Agent within an evolving multi-agent system while its role and integration points were still being refined. Limited cloud compute also restricted testing of RAG and LLM-dependent evaluation features.

Through the project, I learned how to turn ideas from research papers into concrete system requirements and implementation decisions, work within an existing collaborative codebase, and iterate on an open-ended AI problem as the architecture and project needs evolved.

Future Enhancements

1. **RAG Grounding Verification:** Verify responses against instructor-approved course materials.
2. **Configurable Policies:** Support different integrity policies across courses and assignments.
3. **Adaptive Guardrails:** Dynamically select checks based on request and session context.
4. **Safety Integration:** Extend the framework to physical lab safety and equipment monitoring.

References

[1] Rebedea, T., Dinu, R., Sreedhar, M., Parisien, C., & Cohen, J. (2023). NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails. arXiv:2310.10501.
[2] Dong, Y., Mu, R., Zhang, Y., et al. (2024). Safeguarding Large Language Models: A Survey. arXiv:2406.02622. Pasted markdown
[3] Li, B., Tan, L., Zakharov, W., Qiang, Q., & Acton, C. B. (2026). Tutor, Not Solver: Designing a Guardrailed AI Assistant for Learning in Higher Education: A Design Case of PeteChat. arXiv:2606.09845.
[4] Bai, Y., et al. (2022). Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073.
[5] Luo, W., Dai, S., Liu, X., et al. (2025). AGrail: A Lifelong Agent Guardrail with Effective and Adaptive Safety Detection. arXiv:2502.11448.