TartanMaroon: Multi-Agent Academic Advising with Iterative Negotiation and Transparent Collaboration

Peidi Dong[1] and **Houda Bouamor**[1] and **Yunze Xiao**[2] and **Devi G. Kurup**[1]
[1]Carnegie Mellon University in Qatar
[2]Carnegie Mellon University
{peidid, hbouamor, yunzex, dkurup}@andrew.cmu.edu

Abstract

We present TartanMaroon, a deployable multi-agent academic advising system that handles the full complexity spectrum of student queries, from factual lookups to constrained multi-semester planning. We make three contributions: (1) a proposal–critique negotiation protocol in which a Planning Agent generates degree plans evaluated in parallel by domain-specialized agents, enabling detection of cross-domain constraint violations that single-pass outputs miss; (2) a real-time transparency interface streaming agent reasoning and negotiation rounds to users, supported by pilot feedback showing increased trust over standard LLM chatbots; and (3) *TartanBench*, a difficulty-stratified benchmark of 220 advising queries across five complexity tiers, released open-source without exposing individual student records. A five-configuration ablation study establishes a *complexity–necessity curve*: single-agent systems perform competitively on simple queries, while multi-agent coordination yields gains of up to +31 points on planning tasks.

1 Introduction

Academic advising is a critical yet resource-intensive component of higher education (Young-Jones et al., 2013; Mu and Fosnacht, 2019; Chan et al., 2019). Students must simultaneously navigate degree requirements, prerequisite chains, course schedules, and institutional policies under constraints that interact in non-obvious ways: a seemingly simple question such as *"How can I graduate on time if I add a Psychology minor?"* can trigger cascading cross-domain violations that are difficult to detect in isolation. These challenges impose a significant cognitive burden on both students and advisors (McKinney et al., 2024), and advising errors carry real academic and financial consequences (White, 2015).

LLMs have shown promise in specialized question answering (Brown et al., 2020; Wei et al., 2022), yet applying a single LLM to academic advising remains brittle: a monolithic model must simultaneously maintain awareness of heterogeneous institutional constraints, making it prone to cross-domain oversights on complex planning queries (Liu et al., 2024). Multi-agent architectures offer a natural decomposition, and frameworks such as AutoGen (Wu et al., 2023) and MetaGPT (Hong et al., 2024) have made agent orchestration increasingly accessible. The closest related system is MARAUS (Nguyen-Duc et al., 2025), a multi-agent RAG chatbot for university admissions counseling; however, it targets largely lookup-based queries, provides no transparency into its coordination process, and offers no empirical analysis of when multi-agent coordination is actually necessary. A fundamental question therefore remains: *given the overhead of multi-agent systems, when do they actually improve outcomes over a well-prompted single agent?*

In this paper, we present *TartanMaroon*, a multi-agent academic advising system with four domain-specialized agents, an LLM-driven coordinator, a proposal–critique negotiation protocol for multi-semester planning, and a real-time transparency interface that makes the full coordination process legible to end users.[1] We make three contributions: (1) a deployable advising system addressing the full spectrum of student query complexity; (2) *TartanBench*, a difficulty-stratified benchmark of 220 queries grounded in realistic academic scenarios across five complexity tiers; and (3) a five-configuration ablation study establishing a *complexity–necessity curve*, empirical

[1]Demo: https://multi-agent-advising-bot.vercel.app/,
Code: https://github.com/peidid/Multi-Agent-Advising-Bot,
Video: https://www.youtube.com/watch?v=JRDQ1g5k52I

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 840–850

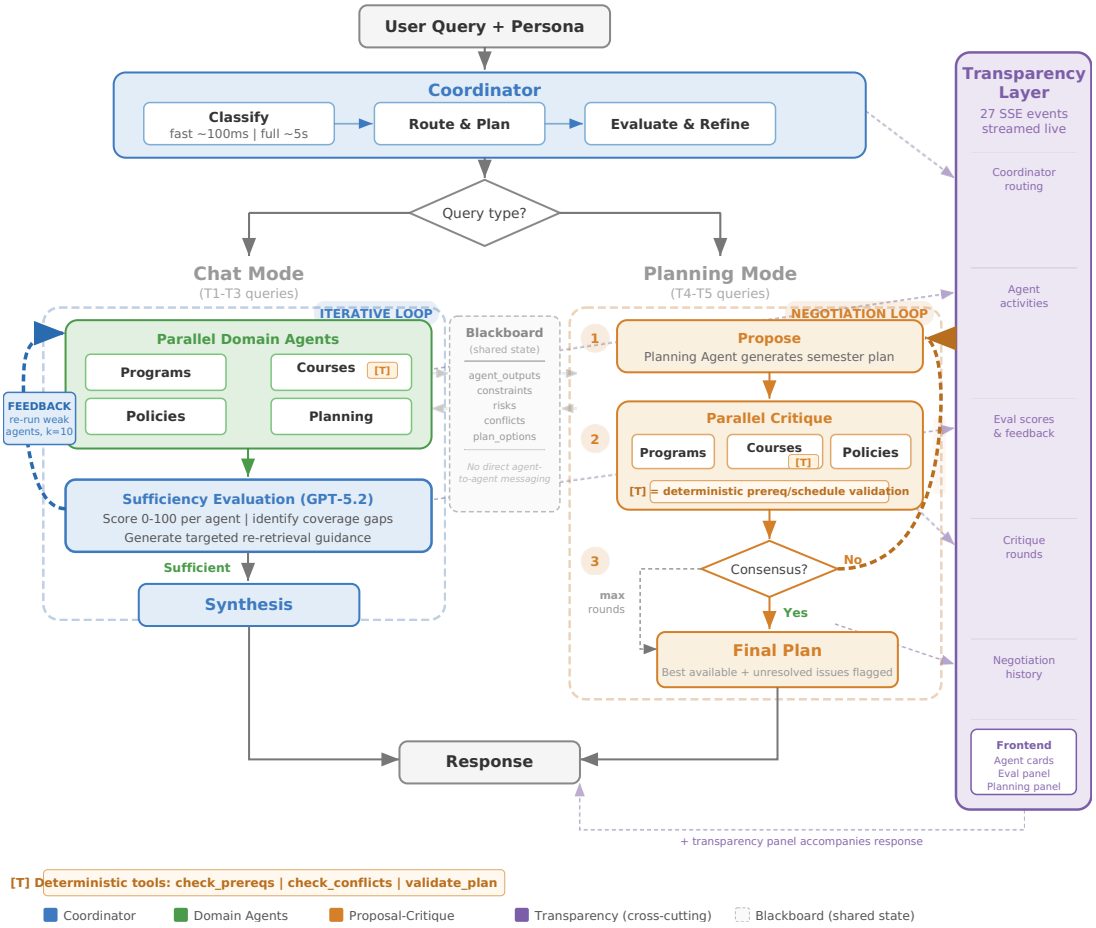

Figure 1: Architecture of *TartanMaroon*. An LLM coordinator routes queries to Chat Mode (T1–T3) or Planning Mode (T4–T5). Domain-specialized agents operate over dedicated knowledge stores and coordinate through a shared blackboard, while a transparency layer streams system activity to the interface.

evidence of precisely when multi-agent coordination is justified and when it constitutes unnecessary overhead. The system serves students seeking self-service planning support, advisors managing high-volume routine queries, and researchers studying multi-agent reasoning in education.

2 System Architecture

TartanMaroon processes free-text student queries through four domain-specialized LLM agents coordinated by a central LLM coordinator (Figure 1). Queries are routed to one of two execution modes based on predicted complexity: **Chat Mode** (T1–T3) for informational queries, which uses an iterative evaluation loop, and **Planning Mode** (T4–T5) for multi-semester planning, which uses a proposal–critique negotiation protocol. A transparency layer streams system activities to the user interface in real time. Additional details on the coordination design and event-streaming mechanism are provided in Appendix A.

2.1 LLM Coordinator

The coordinator performs three functions: CLASSIFY, ROUTE & PLAN, and EVALUATE & REFINE. In the first, it predicts the query complexity tier (T1–T5) using a fine-tuned LLM classifier (details in Appendix B) and selects Chat Mode or Planning Mode accordingly (at the "Query type?" decision point). Second, it produces a *WorkflowPlan* that decomposes the user's query into domain-specific subtasks and assigns them to the relevant agents. Finally, after agents return their outputs, the coordinator triggers sufficiency evaluation and, when necessary, targeted re-retrieval or negotiation.

2.2 Domain-Specialized Agents

Four domain-specialized agents operate in parallel (green "Parallel Domain Agents" block in Figure 1): **Courses** (catalog and prerequisites), **Programs** (degree requirements and plans), **Policies** (academic rules), and **Planning** (course offerings

and schedules). Each agent retrieves information from a dedicated vector store (Appendix C) and communicates through a shared **blackboard** containing agent outputs, constraints, risks, and detected conflicts (Appendix D). Agents execute in parallel and do not communicate directly; coordination flows through the blackboard and the coordinator. Each agent returns a structured `AgentOutput`. The Courses Agent additionally invokes deterministic validation tools for prerequisite checking, schedule conflict detection, and plan validation, ensuring that critical constraints are verified programmatically.

2.3 Iterative Semantic Evaluation

In **Chat Mode** (T1–T3), agent outputs are evaluated by a *sufficiency evaluator* (GPT-5.2 (OpenAI, 2025)) that scores accuracy, completeness, and safety on a 0–100 scale. We set the sufficiency threshold to 80 based on pilot calibration to balance quality and latency: lower thresholds triggered redundant re-retrieval with minimal gains, while higher thresholds missed correctable omissions (e.g., uncited constraints). If the score falls below a threshold of 80, the evaluator generates targeted feedback identifying missing information, and the coordinator re-activates only the relevant agents with deeper retrieval. The loop executes for up to three rounds to bound latency in interactive settings; the system then proceeds to *synthesis* and returns the final response to the user. As illustrated by the iterative loop in Figure 1, this targeted re-retrieval focuses on identified gaps rather than re-running the full query across all agents.

2.4 Proposal–Critique Negotiation Protocol

When the coordinator classifies a query as a planning task (T4–T5) at the "Query type?" decision node in Figure 1, the system activates **Planning Mode** via a structured proposal–critique negotiation protocol. Planning queries typically involve multi-semester degree planning requiring reasoning over interacting constraints: prerequisites, degree requirements, scheduling availability, and institutional policies. The **Planning Agent** acts as *proposer*, generating a candidate semester-by-semester plan (`CoursePlanJSON`), while the **Programs**, **Courses**, and **Policies** agents act as *critiquers* in parallel: Programs checks degree requirements, Courses verifies prerequisites and scheduling conflicts using deterministic validation tools, and Policies validates unit limits and academic-standing rules. If all agents approve, the plan is finalized; otherwise, critiques are aggregated and returned to the Planning Agent for revision. The loop runs for up to three rounds, after which the best available plan is returned with unresolved issues explicitly flagged—enabling domain-specialized agents to catch constraint violations that single-pass outputs systematically miss. In practice, Chat Mode queries (T1–T2) complete in 8–15 seconds end-to-end under the full multi-agent pipeline. T3–T5 queries vary depending on the number of evaluation or negotiation rounds triggered; the majority complete within 40 seconds, though worst-case scenarios where all three rounds are exhausted can reach up to 120 seconds, acceptable for an advising context where the alternative is waiting days for an appointment.

2.5 Transparency Layer

The system includes a transparency layer that streams coordination events to the interface in real time (Figure 1). Users can observe agent retrieval, evaluation feedback, and negotiation rounds as they occur. This transparency is deliberate: in high-stakes academic advising, seeing that the Policies Agent flagged a unit-limit violation with high confidence provides stronger guidance than presenting only a final answer. Additional transparency-layer implementation details are provided in Appendix E.

2.6 Implementation Details

TartanMaroon is built with FastAPI and Next.js 14 with TypeScript, using LangGraph for workflow orchestration, ChromaDB for per-agent vector storage (Appendix C), and MongoDB for conversation history. The system is deployed via Docker Compose and is released under the MIT license. Adapting *TartanMaroon* to a new institution requires only replacing the domain document collections and updating agent capability descriptions; the coordination architecture, negotiation protocol, and evaluation mechanisms do not require modification.

3 *TartanBench*: A Difficulty-Stratified Advising Benchmark

To evaluate *TartanMaroon* and provide a reusable resource for the community, we construct *Tartan-*

Bench, a difficulty-stratified benchmark of natural-language academic advising queries. Unlike prior advising evaluations that rely on simple retrieval metrics over single-domain questions, *TartanBench* is grounded in realistic student personas and designed to expose the failure modes of both single-agent and multi-agent systems across the full complexity spectrum of advising queries.

3.1 Query Taxonomy

TartanBench organizes queries into five tiers of increasing natural language and reasoning complexity, directly corresponding to the Chat Mode (T1–T3) and Planning Mode (T4–T5) execution paths in *TartanMaroon* (Section 2):

- **T1 — Factual lookup** (n=50): Single-fact retrieval from a single knowledge domain (e.g., *"What are the prerequisites for IS 67-272?"*).
- **T2 — Single-domain reasoning** (n=50): Queries requiring inference within one domain (e.g., *"Which of my remaining requirements can I satisfy next semester?"*).
- **T3 — Cross-domain constraint reasoning** (n=50): Queries requiring joint inference over two or more domains (e.g., *"Can I take these three courses simultaneously, given my standing and prerequisites?"*).
- **T4 — Longitudinal multi-step planning** (n=40): Multi-semester planning queries with interacting constraints across all domains (e.g., *"Given my completed courses, plan my remaining three semesters to finish the IS major with a CS minor while staying under 54 units per semester."*).
- **T5 — Adversarial and edge cases** (n=30): Queries designed from documented advising edge cases to expose known failure modes of LLM-based advising: conflicting policy interactions, near-miss prerequisite chains, unit-limit edge cases, and queries with no valid solution.

The distribution is deliberately weighted toward T3–T5 (n=120 of 220 total), the tiers where multi-agent coordination has the greatest potential impact and where single-agent reasoning is most likely to fail.

3.2 Student Personas

All queries are grounded in 64 anonymized student personas constructed from de-identified historical advising records across four programs (Computer Science, Information Systems, Business Administration, and Biological Sciences), balanced across class years, requirement completion status, and academic standing.[2] Each persona includes a chronological course history, program declarations, and a narrative context that reflects realistic student situations (e.g., *"considering adding a BA minor with three semesters remaining and a part-time job constraint"*). Grounding queries in personas ensures that natural language complexity; ambiguous references, implicit constraints, and competing goals; reflects genuine advising interactions rather than artificially clean synthetic formulations. Personas are retained internally and are not distributed; the released benchmark contains only the queries and structured reference answers, which are sufficient for reproducible evaluation without exposing underlying student records.

3.3 Construction and Validation

We apply a two-track construction process matched to query complexity. **T1–T2** queries are systematically generated using curriculum data with LLM assistance and spot-checked against source documents for factual accuracy. The generation-and-verify approach is appropriate at these tiers because factual and single-domain queries have unambiguously correct answers that can be validated against structured data.

T3–T5 queries are authored, answered, and validated entirely by human students and academic advisors, who cross-validated both for correctness and difficulty, essential because constraint interactions and known traps must reflect genuine advising complexity.

Each reference answer is structured as a four-field object enabling reproducible fine-grained evaluation: `key_facts` (required information units), `required_reasoning` (inference steps that must be present), `known_traps` (common errors a correct answer must avoid), and `acceptable_alternatives` (valid answer variants). This structure supports both automated scoring and human evaluation without requiring exact-match comparison to a single gold string. *TartanBench* is released alongside the system under the same open-source license to facilitate reproducible evaluation of academic advising systems and multi-agent reasoning architectures.

[2] No personally identifiable information is included

	S3	S4	S2	S1	S0
LLM calls (typ.)	1	1	2+N	7–10	7–10
CoT reasoning	–	✓	–	–	–
Coordinator routing	–	–	✓	✓	✓
Specialized agents	–	–	✓	✓	✓
Eval. loop + re-retrieval	–	–	–	✓	✓
Proposal–critique	–	–	–	✓	✓
Transparency	–	–	–	–	✓

Table 1: Ablation configurations. Each row adds one capability over the previous. S4→S2 replaces CoT with domain specialization; S0 and S1 produce identical outputs, differing only in the transparency layer.

	T1	T2	T3	T4	T5	All
S3 (Single)	91	84	71	53	37	67.2
S4 (+CoT)	93	88	77	60	45	72.6
S2 (+Agents)	92	87	83	73	58	78.6
S1 (+Eval.)	93	89	87	82	76	85.4
S0 (+Transp.)	93	89	87	82	76	85.4
Δ(S4→S1)	0	+1	+10	+22	+31	+12.8

Table 2: Mean scores (0–100) by tier. The Δ row isolates S1 gains over S4; gains concentrate on T4–T5, establishing the complexity–necessity curve.

4 Evaluation and Results

We evaluate five configurations on *TartanBench* to isolate the contribution of (1) domain specialization, (2) iterative semantic evaluation, and (3) proposal–critique negotiation.

4.1 Ablation Configurations

We define five configurations forming a controlled ablation ladder. All configurations share the same backbone model, RAG collections (k=5), and deterministic tools (Table 1). Note that configuration labels reflect system identifiers rather than capability order; the capability ladder progresses as **S3**→**S4**→**S2**→**S1**→**S0**, with each step adding one component over the previous.

S3 is a single LLM call over all five RAG domains (25 retrieved chunks). **S4** adds a five-step chain-of-thought scaffold over the same context. **S2** replaces CoT with LLM-driven coordinator routing and four parallel specialized agents, testing domain specialization as an alternative mechanism for structured reasoning. **S1** adds the iterative evaluation loop (up to 3 rounds, k=10 re-retrieval) and the proposal–critique protocol for planning queries. **S0** adds real-time transparency; outputs are identical to S1.

4.2 Evaluation Protocol

We evaluated all five configurations on the full *TartanBench* of 220 queries across five tiers grounded in 64 student personas. Each of the 220 system responses is scored by Claude Opus (distinct from the GPT-based backbone model) on factual accuracy, completeness, trap avoidance, and coherence (0–5 each). The evaluation uses structured reference answers consisting of `key_facts`, `required_reasoning`, and `known_traps`, enabling fine-grained scoring without requiring exact string matching. Two independent judges evaluate each response, and their scores are averaged; the agreement between judges is $\kappa = 0.81$. We also conducted a small pilot qualitative user study with five students and one academic advisor.

4.3 Results

Because *TartanBench* provides structured reference answers capturing required facts and reasoning steps, automated evaluation provides a scalable and reproducible assessment of system behavior across all configurations.

Complexity–necessity curve. Table 2 reveals the paper's central finding: configurations converge on T1–T2 and diverge sharply on T4–T5. On factual queries, S4 (single agent + CoT) performs within 1–2 points of the full system at one-seventh the LLM calls. On constrained planning (T4) and adversarial cases (T5), the full pipeline outperforms S4 by +22 and +31 points respectively. Gains decompose across two mechanisms: **domain specialization** (S4→S2) contributes +13 on T4 by reducing cross-domain retrieval noise, while **iterative proposal–critique collaboration** (S2→S1) contributes a further +9 on T4 and +18 on T5 by catching constraint violations—prerequisite ordering errors, unit-limit exceedances, and scheduling conflicts—that first-pass outputs systematically miss. The "All" column reports the unweighted mean of tier scores; weighted averages over the 220 queries yield similar ordering but differ slightly due to unequal tier sizes (n=50, 50, 50, 40, 30). This establishes a clear design principle: multi-agent coordination introduces unnecessary overhead for simple queries but becomes essential for planning tasks involving interacting institutional constraints. As S0 and S1 produce identical outputs differing only in the transparency layer, the automated benchmark does not capture S0's contri-

bution; the value of real-time transparency is instead evaluated through the qualitative pilot study, where participants consistently identified it as a key driver of trust.

Limitations of multi-agent coordination. Table 2 surfaces two regimes where multi-agent coordination fails to justify its overhead. First, on factual and single-domain queries (T1–T2), **S4** (single agent + CoT) falls within 1–2 points of the full pipeline while requiring one-seventh the LLM calls and completing in under 2 seconds, compared to 8–15 seconds for **S1**. The coordination machinery, routing, parallel retrieval across four agents, and the sufficiency evaluation loop, adds measurable latency with negligible quality return at these tiers, representing unnecessary overhead for deployments dominated by factual traffic.

Second, coordinator misclassification introduces a structural failure mode absent in single-agent baselines: a T2 query routed to Planning Mode activates agents whose retrieved constraints are irrelevant to the actual question, polluting the shared blackboard with spurious conflicts and degrading response quality below even **S3**. Although our classifier achieves high accuracy on TartanBench, this failure mode is consequential precisely because it is silent, the system returns a response without signaling that over-activation occurred.

Error analysis by trap type. Table 3 breaks down errors among failed T4–T5 queries by trap category across configurations. The percentages reflect the proportion of errors attributable to each trap type within each configuration's failure set; as overall error counts decrease across configurations, these proportions shift to reveal which traps persist as the system improves. Two patterns emerge. First, multi-agent coordination eliminates structurally detectable traps almost entirely: unit-limit exceedances drop from 3% to 0% by **S2**, and cross-policy conflicts drop from 26% to 6% by **S1**, as domain-specialized agents and deterministic validation tools catch these violations programmatically. Prerequisite violations similarly decrease to 13% under **S1**. Second, scheduling conflicts and unflagged infeasibility account for a growing share of the residual error pool, not because absolute failures increase, but because they are the traps that survive after easier violations are resolved. Scheduling conflicts are difficult to eliminate because they depend on semester-specific offering data that is sparse in the knowledge store. Unflagged infeasibility reflects a more fundamental limitation: the proposal, critique negotiation loop is designed to converge on a valid plan, and consequently resists concluding that no valid plan exists. These two categories represent the primary targets for future improvement.

Trap Type	S3	S4	S2	S1
Prerequisite violation	15%	17%	22%	13%
Unit-limit exceedance	3%	2%	0%	0%
Scheduling conflict	50%	43%	57%	65%
Cross-policy conflict	26%	23%	10%	6%
Unflagged infeasibility	48%	52%	55%	63%

Table 3: Error breakdown by trap type among failed T4–T5 queries. Percentages reflect the share of errors within each configuration's failure set; rows do not sum to 100% as a single query may exhibit multiple trap types. As overall failures decrease across configurations, scheduling conflicts and unflagged infeasibility constitute a growing share of the residual error pool.

Human Evaluation. The pilot study presented participants with a mix of query types spanning T1–T5, allowing evaluation of both factual lookup and multi-semester planning interactions. Students reported that the system substantially reduces the logistical overhead and waiting time of scheduling advising appointments (often days) for questions that can be resolved instantly through *TartanMaroon*, and highlighted the transparency interface as a key driver of trust: participants noted that observing agent coordination and multi-step verification made the system feel meaningfully different from a standard LLM chatbot, increasing their confidence in the answers provided (one student stated: *"It doesn't feel like plain ChatGPT where hallucinations can happen. It seems like a more sophisticated system that does a lot behind the scenes before giving an answer, which makes me trust it more."*) Students also found the multi-semester planning functionality particularly valuable for mapping out degree timelines without requiring an appointment. The advisor identified two benefits: the system serves as an effective entry point for sessions, shifting appointments from information-gathering to decision-making, focusing the advisor's expertise on high-stakes situations requiring human judgment; and it consolidates program requirements and policies otherwise scattered across institutional websites. The advisor emphasized that visible reasoning steps are essential for professional adoption, allowing

advisors to verify recommendations rather than passively endorse a black-box answer.

5 Related Work

Research on LLM-based systems relevant to *TartanMaroon* spans three main directions: (1) multi-agent coordination architectures, (2) iterative refinement mechanisms for improving LLM reasoning, and (3) conversational AI systems for educational support and academic advising.

Multi-Agent Coordination and Iterative Reasoning. Recent frameworks such as AutoGen (Wu et al., 2023), MetaGPT (Hong et al., 2024), and AgentVerse (Chen et al., 2023) demonstrate the expressiveness of multi-agent LLM architectures but treat coordination overhead as a fixed cost, without characterizing when it is necessary over capable single-agent baselines. Iterative refinement approaches, including Self-Refine (Madaan et al., 2023), multiagent debate (Du et al., 2024), and Reflexion (Shinn et al., 2023) improve outputs through feedback but rely on homogeneous model instances. Our proposal–critique protocol differs in two ways: critique is performed by *heterogeneous, domain-specialized agents*, enabling detection of cross-domain constraint violations that self-critique cannot surface; and our ablation reveals that iterative negotiation yields gains only above a certain query complexity threshold.

RAG, Tool Reasoning, and Educational AI. *TartanMaroon* builds on retrieval-augmented generation (Lewis et al., 2020) and tool-augmented reasoning (Yao et al., 2023; Schick et al., 2023), distributing retrieval across specialized agents rather than a single chain. LLMs have shown great promise in higher education (Kasneci et al., 2023; Hellas et al., 2024), and multi-agent approaches show early promise for educational engagement (Chu et al., 2025).

Academic Advising Systems. The most directly related system is MARAUS (Nguyen-Duc et al., 2025), a multi-agent RAG chatbot for university admissions counseling. However, it differs from *TartanMaroon* in four key respects: admissions counseling involves lookup over static rules, whereas academic advising requires *cross-domain constraint satisfaction* where decisions cascade across prerequisites, credit limits, and scheduling; MARAUS exposes no internal coordination process; it lacks a proposal–critique protocol; and crucially, it does not analyze when multi-agent coordination is actually necessary. *TartanBench* and our five-configuration ablation fill both the system and evaluation gaps left by prior work.

6 Conclusion and Future Work

We present *TartanMaroon*, a deployable multi-agent academic advising system with proposal–critique negotiation and real-time transparency, and *TartanBench*, a difficulty-stratified benchmark for rigorous evaluation. Our five-configuration ablation establishes a *complexity–necessity curve*: multi-agent coordination is essential for constrained planning (T3–T5) but unnecessary overhead for simpler queries (T1–T2). Both are released open-source to support evidence-based adoption of multi-agent architectures in institutional settings. Future work will pursue two directions. First, a larger-scale user study with students and academic advisors, currently pending IRB approval, will assess longitudinal impact on advising outcomes and system trust, with a human evaluation subset specifically targeting T4–T5 planning queries across four trap categories: prerequisite ordering, no-solution detection, policy contradictions, and edge-case unit limits. Second, we will investigate explicit infeasibility reasoning to improve system behavior on adversarial T5 cases where no valid plan exists.

Ethical Considerations

TartanMaroon handles student academic records and advising queries, which raises privacy and reliability concerns. All student personas used in *TartanBench* are reconstructed from de-identified advising records; no personally identifiable information is included, and raw records are not distributed. The system is designed as a decision-support tool, not a replacement for human advisors: responses are accompanied by transparent reasoning traces that allow students and advisors to critically evaluate outputs before acting on them. Unresolved conflicts are explicitly flagged rather than silently resolved, reducing the risk of consequential errors in high-stakes planning decisions. Deployment in institutional settings should include human oversight, particularly for T4–T5 planning queries where constraint interactions are complex and errors carry real academic consequences.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Zenobia CY Chan, Ho Yan Chan, Hang Chak Jason Chow, Sze Nga Choy, Ka Yan Ng, Koon Yiu Wong, and Pui Kan Yu. 2019. Academic advising in undergraduate education: A systematic review. *Nurse education today*, 75:58–74.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Zhendong Chu, Shen Wang, Jian Xie, Tinghui Zhu, Yibo Yan, Jingheng Ye, Aoxiao Zhong, Xuming Hu, Jing Liang, Philip S. Yu, and Qingsong Wen. 2025. LLM agents for education: Advances and applications. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13782–13810, Suzhou, China. Association for Computational Linguistics.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the International Conference on Machine Learning*.

Arto Hellas, Juho Leinonen, and Leo Leppänen. 2024. Experiences from integrating large language model chatbots into the classroom. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 1*. ACM.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *Proceedings of the International Conference on Learning Representations*.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36.

Lyle McKinney, Gerald V Bourdeau, Andrea Backscheider Burridge, Mimi Lee, Melissa Miller-Waters, and Yolanda M Barnes. 2024. " i advise, you decide": How academic advisors shape community college students' enrollment and credit load decisions. *The Review of Higher Education*, 47(4):519–547.

Lanlan Mu and Kevin Fosnacht. 2019. Effective advising: How academic advising influences student learning outcomes in different institutional contexts. *The Review of Higher Education*, 42(4):1283–1307.

Anh Nguyen-Duc, Chien Vu Manh, B. A. Tran, Viet P. Ngo, Lu Chi, and Anh Quang Nguyen. 2025. MA-RAUS: An empirical study of multi-agent RAG for real-world university admissions counseling. *arXiv preprint*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Eric R White. 2015. Academic advising in higher education: A place at the core. *The Journal of General Education*, 64(4):263–277.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izheng Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations*.

Adena D Young-Jones, Tracie D Burt, Stephanie Dixon, and Melissa J Hawthorne. 2013. Academic advising: does it really impact student success? *Quality Assurance in Education*, 21(1):7–19.

A Coordination Design and Event Streaming

Reasoning vs. Orchestration Design Principle A key design principle of *TartanMaroon* is the separation of *language-model reasoning* from *system-level orchestration*. LLMs are responsible for interpreting natural-language queries, generating domain-specific proposals, and synthesizing responses. In contrast, workflow control, constraint validation, and negotiation structure are implemented through deterministic orchestration components.

This separation improves reliability for planning tasks involving institutional constraints. For example, prerequisite chains, unit limits, and schedule conflicts are validated programmatically rather than relying solely on LLM reasoning. The LLM generates candidate solutions, while deterministic tools and the coordinator enforce structural correctness and workflow control.

Event Streaming and Transparency To expose the internal coordination process during interaction, *TartanMaroon* streams system activity to the frontend through Server-Sent Events (SSE). The transparency layer emits 27 typed event categories covering:

- coordinator classification and routing decisions
- agent retrieval steps and outputs
- sufficiency evaluation scores and feedback
- re-retrieval triggers during iterative refinement
- proposal, critique, and revision rounds in planning mode

Events are emitted asynchronously by parallel agents and ordered through a queue-based synchronization mechanism before being sent to the client interface. The frontend renders these events into structured interface components that allow users to observe the system's reasoning workflow in real time.

B Query Complexity Classification

TartanMaroon routes queries to different execution modes based on their predicted complexity tier (T1–T5). We train a lightweight LLM-based classifier on labeled advising queries to estimate the difficulty of incoming questions.

The classifier predicts both the tier label and the execution mode: **Chat Mode** for informational queries (T1–T3) and **Planning Mode** for multi-constraint planning queries (T4–T5). Training data consists of manually labeled advising queries spanning five tiers of reasoning complexity described in Section 3.

At inference time the classifier operates in two stages. A fast-path prompt handles straightforward queries with minimal latency. If confidence is low, the coordinator performs a full LLM analysis incorporating the student's academic profile and conversation history. The predicted tier determines whether the system executes the iterative evaluation pipeline or activates the negotiation protocol.

C Knowledge Stores and Retrieval

Each domain-specialized agent operates over a dedicated ChromaDB collection containing institutional documents embedded using OpenAI `text-embedding-3-large`. Separating knowledge by domain reduces retrieval noise and allows each agent to reason over documents relevant to its expertise.

The system organizes knowledge into four domains:

- **Courses**: course catalog entries including codes, units, descriptions, and prerequisite chains.
- **Programs**: degree requirements, major and minor structures, and sample curricula.
- **Policies**: institutional rules including academic standing, registration limits, and grading policies.
- **Planning**: semester offerings and scheduling data.

At query time, each agent retrieves the top-k documents from its collection using embedding simi-

larity and uses these documents as context for reasoning.

Deterministic validation tools. In addition to retrieval-based reasoning, the Courses Agent invokes four programmatic validation tools operating directly on course data: `check_prereqs_satisfied` verifies prerequisite chains; `check_courses_conflict` detects scheduling conflicts; `validate_semester_plan` validates constraints within a single semester; `validate_full_plan` verifies multi-semester plans, including prerequisite ordering.

These tools are used both in *Chat Mode* and during the critique phase of *Planning Mode*, where symbolic validation complements LLM-based reasoning.

Institutional scope. The current deployment covers Carnegie Mellon University Qatar, including four undergraduate programs (Computer Science, Information Systems, Business Administration, and Biological Sciences), 17 minors, and six semesters of scheduling data. Adapting the system to a new institution requires replacing the indexed documents and updating agent domain descriptions; the coordination architecture and negotiation protocol remain unchanged.

D Blackboard State Representation

Agents coordinate through a shared **blackboard** state maintained by the coordinator. The blackboard stores intermediate reasoning artifacts and structured outputs generated by the agents.

The state object contains several key fields:

- `agent_outputs`: responses generated by each domain-specialized agent.
- `constraints`: extracted institutional constraints, including prerequisites, policy rules, and credit limits.
- `risks`: potential issues identified by agents, such as missing prerequisites or scheduling conflicts.
- `conflicts`: detected inconsistencies across domain outputs.
- `plan_options`: candidate course plans produced during planning queries.

Agents do not communicate directly with each other; instead, all coordination occurs through updates to the blackboard and decisions made by the coordinator. This design simplifies synchronization and enables deterministic control over the multi-agent workflow.

Concurrent writes and conflict handling. The blackboard is not subject to race conditions because agents do not write to it directly during execution. Each agent returns a completed, immutable `AgentOutput` object to the coordinator upon finishing; there are no partial or mid-execution writes to shared state. The coordinator collects all agent outputs after parallel execution completes and performs a single sequential write to the blackboard, eliminating any concurrent write window.

Mutually exclusive constraints, for example, two agents identifying incompatible policy interpretations, are therefore not a concurrency problem. They are surfaced as structured entries in the `conflicts` field of the blackboard state, which the coordinator explicitly inspects before triggering the negotiation loop. Resolving such conflicts is the purpose of the proposal–critique protocol rather than a symptom of a synchronization failure.

We note that the SSE event queue described in Appendix E handles ordering of *streaming events* for the transparency layer and is a separate concern from blackboard state writes; the two mechanisms do not interact.

E Transparency Layer Implementation Details

Event Streaming Infrastructure *TartanMaroon* streams system activity to the frontend via Server-Sent Events (SSE), emitting typed events for coordinator routing decisions, agent retrieval and outputs, sufficiency evaluation scores, re-retrieval triggers, and Planning Mode rounds. Each event includes structured metadata describing the current stage, responsible agent, and relevant outputs.

Concurrency and Event Ordering Since agents execute in parallel, a shared coordinator-managed queue with lightweight lock synchronization serializes events before transmission, preserving logical ordering across asynchronous agent completions.

Interface Design Principles The interface follows three principles: **progressive disclosure** (high-level progress by default, detailed traces on demand); **real-time streaming** (events rendered immediately to reduce perceived latency); and

structured presentation (typed components such as confidence indicators and critique cards rather than raw LLM output).