AgentMaster: A Multi-Agent Conversational Framework Using A2A and MCP Protocols for Multimodal Information Retrieval and Analysis

Callie C. Liao*
Stanford University
Stanford, CA, USA
ccliao@stanford.edu

Duoduo Liao*
George Mason University
Fairfax, VA, USA
dliao2@gmu.edu

Sai Surya Gadiraju*
George Mason University
Fairfax, VA, USA
sgadira3@gmu.edu

Abstract

The rise of Multi-Agent Systems (MAS) in Artificial Intelligence (AI), especially integrated with Large Language Models (LLMs), has greatly facilitated the resolution of complex tasks. However, current systems are still facing challenges of inter-agent communication, coordination, and interaction with heterogeneous tools and resources. Most recently, the Model Context Protocol (MCP) by Anthropic and Agent-to-Agent (A2A) communication protocol by Google have been introduced, and to the best of our knowledge, very few applications exist where both protocols are employed within a single MAS framework. We present a pilot study of AgentMaster, a novel modular multi-protocol MAS framework with self-implemented A2A and MCP, enabling dynamic coordination and flexible communication. Through a unified conversational interface, the system supports natural language interaction without prior technical expertise and responds to multimodal queries for tasks including information retrieval, question answering, and image analysis. Evaluation through the BERTScore F1 and LLM-as-a-Judge metric G-Eval averaged 96.3% and 87.1%, revealing robust inter-agent coordination, query decomposition, dynamic routing, and domain-specific, relevant responses. Overall, our proposed framework contributes to the potential capabilities of domain-specific, cooperative, and scalable conversational AI powered by MAS.

1 Introduction

Recent advances in artificial intelligence (AI) have increasingly focused on Multi-Agent Systems (MAS), in which multiple intelligent agents collaborate, communicate, and share contextual information to address complex tasks (Li et al., 2025; Qian et al., 2024; Yao et al., 2023). The integration of Large Language Models (LLMs) into MAS frameworks has significantly broadened their applicability, enabling general-purpose collaboration, natural language interaction, and open-ended reasoning (Hu et al., 2025; Luo et al., 2024; Huang et al., 2024; Guo et al., 2024). This makes LLM-based MAS particularly well-suited for dynamic, unstructured tasks such as multimodal data analysis, research automation, and intelligent assistance (Dong et al., 2024; Islam et al., 2024; Lin et al., 2025). By distributing intelligence across agents, LLM-based MAS offer a promising approach to overcoming the limitations of standalone LLMs (Gemini, 2025; OpenAI, 2024; Touvron et al., 2023).

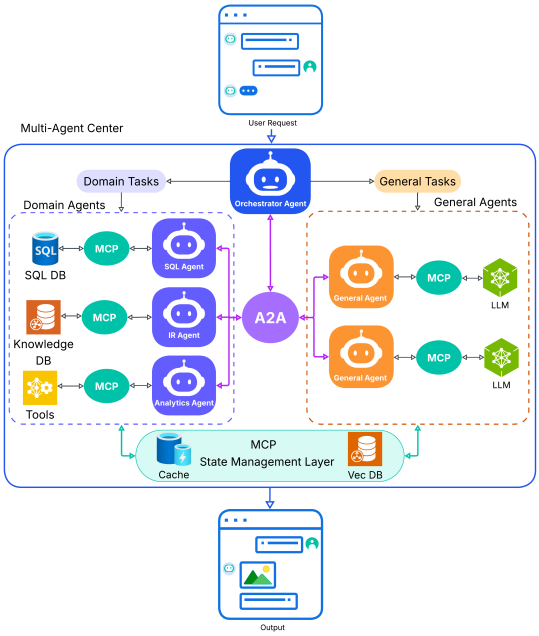


Figure 1: The general MAS framework of AgentMaster.

Despite their potential, current LLM-based MAS face critical challenges that limit their scalability, robustness, and effectiveness. These challenges span technical, architectural, and practical dimensions, including agent coordination, communication, interaction with heterogeneous tools and data

*Equal contribution.

sources, knowledge representation and reasoning, modularity, and integration of domain-specific expertise (Du et al., 2025; Shen et al., 2023). Especially, in domain-specific contexts where specialized agents are increasingly essential (Yu et al., 2025; Mathur et al., 2024; Gadiraju et al., 2024), these systems often require substantial domain-specific knowledge and the capability to process diverse data modalities, posing additional challenges for effective automation and coordination (Yu et al., 2025; Aminian-Dehkordi et al., 2025; Haase and Pokutta, 2025; Zhang et al., 2025b).

Most recently, two new open standards, Anthropic's Model Context Protocol (MCP) (Anthropic, 2024) and Agent-to-Agent (A2A) communication protocol introduced by Google (Surapaneni et al., 2025), aim to address these challenges. MCP, announced in May 2024, streamlines the process by providing a standardized interface for accessing various tools and resources, enhancing the modularity, interoperability, and statefulness of multi-agent and tool-augmented systems. A2A, announced in May 2025, complements MCP by facilitating structured inter-agent communication, which allows multiple AI agents to exchange messages, distribute subtasks, and build shared understanding to solve problems collectively. These protocols offer a systematic alternative to the fragmented, ad hoc integration approaches common in current MAS implementations. Both A2A and MCP can be developed using existing SDKs or fully implemented by users as needed. These protocols offer a systematic alternative to the fragmented, ad hoc integration approaches common in current MAS implementations (Jeong, 2025; Yang et al., 2025).

Existing LLM-based multi-agent systems that do not incorporate A2A or MCP often suffer from static coordination, limited memory, and rigid communication mechanisms. By leveraging these emerging standards, systems can support structured inter-agent communication, maintain shared contextual understanding, and seamlessly interface with external tools, developing more capable, scalable, and cooperative AI systems (Yang et al., 2025; Ehtesham et al., 2025).

To date, both industry and academia have conducted limited research on the application of A2A and MCP within LLM-based MAS. While a few research efforts have explored the independent use of A2A (Habler et al., 2025) and MCP (Krishnan, 2025; Qiu et al., 2025; Sarkar and Sarkar, 2025), there are, to the best of our knowledge, very few applications in which both protocols have been jointly employed within a single MAS framework.

To address these gaps, this paper introduces AgentMaster, a novel modular multi-protocol MAS framework that integrates A2A protocol and MCP. AgentMaster decomposes user queries into specialized workflows executed by dedicated agents, coordinated through A2A and supported by a centralized MCP backend for tool and context management. Users interact with the system through a unified conversational interface, enabling natural language interaction without prior technical expertise. The framework supports complex task decomposition, dynamic routing, and agent-to-agent orchestration. By isolating agents and provisioning separate API keys, the system can manage resource utilization and enforce the separation of concerns between components.

A fully functional prototype through self-developed A2A and MCP demonstrates AgentMaster's capabilities in domain-specific multimodal tasks, including information retrieval, image analysis, database querying, question answering, and content summarization. The system is deployed both locally and on Amazon Web Services (AWS) as a set of Flask-based microservices, and exhibits consistent performance across varied task types in a pilot study.

Our main contributions are as follows:

- This paper introduces AgentMaster, a modular multi-agent MAS framework that integrates Anthropic's MCP and Google's A2A protocol to enable flexible inter-agent communication, intelligent coordination, and retrieval-augmented generation.
- A unified system architecture is designed to support query decomposition, dynamic routing, and orchestration across specialized retrieval agents and multimodal data sources.
- The pilot study explores the implementation of self-developed A2A and MCP protocols specifically designed for AgentMaster without relying on existing libraries such as Google's A2A SDK.
- A fully functional prototype is implemented as a set of Flask-based microservices, demonstrating applicability to information retrieval, image analysis, database querying, question answering, and summarization.

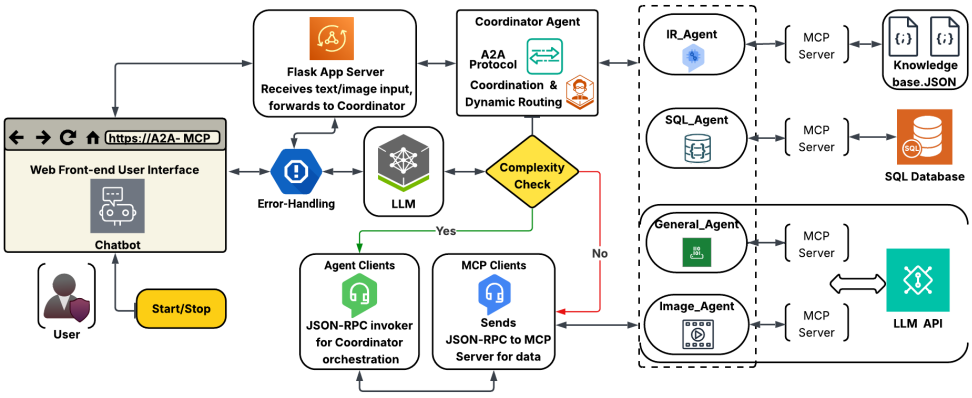


Figure 2: The system architecture of the case study.

- Comprehensive evaluation is conducted using G-Eval, BERTScore, and related metrics to validate correctness, completeness, and semantic fidelity across diverse query types.

2 The General System Framework

Figure 1 illustrates the general multi-protocol MAS architecture of the AgentMaster. The framework comprises four core components: a unified conversational interface, a multi-agent center, multi-agent AI protocols, and a state management layer.

2.1 Unified Conversational Interface

The unified conversational interface resembles a chatbot, receiving user input in various multimodal formats, including text, charts, images, and audio, and generating corresponding output in modalities such as text, images, and structured data tables.

2.2 Multi-Agent Center

The Multi-Agent Center consists of three hierarchical layers of agents: the orchestrator agent, domain agents, and general agents. At the top of the hierarchy, the orchestrator agent is responsible for decomposing tasks and coordinating execution across agents. Domain agents specialize in specific functionalities and may be either LLM-based or non-LLM-based. General agents operate independently, each paired with a dedicated LLM to handle general-purpose reasoning tasks. All agents communicate through the A2A protocol, which enables structured, language-based message exchange. Additionally, each agent is integrated with the MCP protocol, which standardizes interactions with external tools, APIs, and contextual resources.

2.2.1 Orchestrator Agent

The orchestrator agent serves as the central coordinator, identifying available tasks and delegating them to appropriate agents based on their capabilities. To optimize efficiency and accuracy, it may further decompose complex user requests into subtasks for parallel or sequential execution across agents. As a pivotal hub, it not only translates high-level user goals into manageable tasks aligned with agent capabilities, but also facilitates inter-agent communication, handles error management across protocols, and synthesizes outputs into a coherent, unified response.

2.2.2 General Agents

The general agent is designed to handle broad tasks that do not require access to domain-specific datasets. The orchestrator agent determines whether to delegate a task to a general agent or a domain agent, selecting the most appropriate agent based on the nature and complexity of the task.

2.2.3 Domain Agents

Domain agents are specialized for specific domains and designed to interface with domain-relevant functions, datasets, and tools. Each domain agent may internally manage sub-agents to further decompose and process tasks in a modular fashion. These agents communicate not only with each other, but also with general agents, enabling collaborative task execution across domains.

In AgentMaster, domain agents are designed to specialize in common domain-specific tasks, including Structured Query Language (SQL) querying, Information Retrieval (IR), and multimodal data analytics. The framework is extensible, allowing for the integration of additional agents as

required to support diverse application needs.

2.3 Multi-Agent AI Protocols

AgentMaster employs A2A for structured communication between agents, enabling coordination, delegation, and orchestration through standardized JSON-based message exchange. MCP complements this by providing a unified interface for tool access, long-term memory, and context management, enhancing modularity, interoperability, and statefulness in LLM-based agents.

Depending on the application and requirements, the framework leverages the A2A protocol via Google's A2A SDK (Surapaneni et al., 2025), or fully implements it as needed. MCP is developed in a similar manner.

2.4 State Management Layer

The State Management Layer in AgentMaster leverages vector databases and context caches to maintain the MCP state, enabling agents to be context-aware and memory-augmented for efficient handling of multistep, user-specific, and domain-specific tasks. This layer utilizes the vector database to provide persistent semantic memory for retrieving relevant past interactions and documents, while the context cache offers fast, temporary storage for session data and intermediate results during active workflows.

3 System Architecture of the Case Study

Figure 2 illustrates the architecture of a conversational MAS, an example implementation of the AgentMaster framework for multimodal information retrieval and analysis. The system integrates modular components to enable robust, retrieval-augmented question answering through dynamic agent orchestration.

The architecture comprises a web-based user interface, a Flask server acting as the main entry point, a Coordinator Agent (i.e., the Orchestrator Agent within the framework) implementing the A2A protocol, and multiple specialized retrieval agents (i.e., domain agents within the framework). User queries are submitted via the chatbot front end and processed asynchronously by the backend components.

3.1 Coordinator Agent and Complexity Assessment

The Coordinator Agent is responsible for query analysis, routing, and orchestration (Zhang et al., 2025a). A key function is the complexity assessment module, which determines whether a query requires multi-agent collaboration or can be handled by a single retrieval agent. For simple queries, the Coordinator dispatches requests directly to an appropriate MCP Client. In contrast, complex queries trigger Agent Clients that dynamically coordinate multiple retrieval workflows.

3.2 Agent Clients and MCP Clients

Agent Clients serve as JSON-RPC invokers for orchestrating distributed workflows among retrieval agents. MCP Clients manage communication with retrieval backends, dispatching JSON-RPC requests to MCP Servers that encapsulate domain-specific retrieval logic (Kumar et al., 2025). This division enables the system to support compositional retrieval and fallback handling without manual routing configuration.

3.3 Retrieval Agents

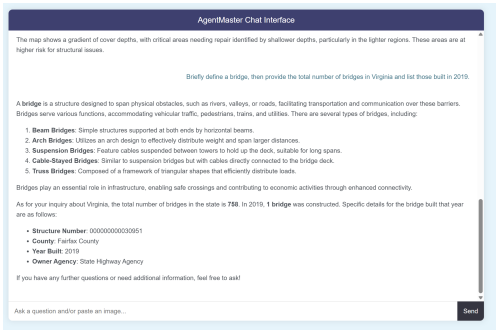
The system incorporates four primary specialized agents: (i) an IR Agent that retrieves unstructured content from knowledge bases; (ii) a SQL Agent that generates and executes SQL queries over relational databases; (iii) an Image Agent that processes image inputs through external vision APIs; and (iv) a General Agent that handles open-domain queries and fallback cases. Each agent exposes an MCP Server endpoint for standardized invocation.

3.4 LLM Integration and Error Handling

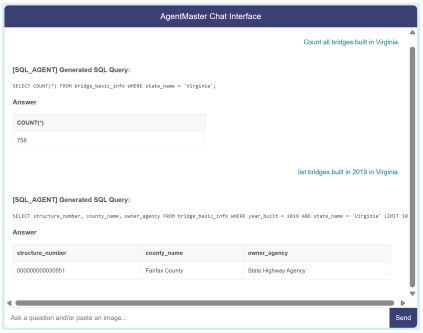
The architecture integrates a local or external LLM for language generation, reasoning, and summarization. The LLM module aggregates partial outputs returned by retrieval agents and formulates the final response. The Flask server and Coordinator Agent include error-handling mechanisms that detect and recover from failures in retrieval workflows and model inference (Williams, 2025).

3.5 End-to-End Workflow

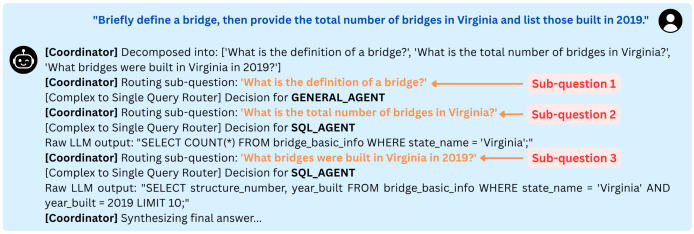
End-to-end query resolution proceeds as follows. The user submits a text or image query via the front end. The Flask server forwards the request to the Coordinator Agent, which performs complexity assessment and routes the query to the appropriate retrieval pathway. Specialized retrieval agents return results via MCP Clients. The LLM module synthesizes the final output, which is delivered to the user interface for presentation.



(a) Frontend example.



(b) Verification of the generated response to the complex query.



(c) Backend example.

Figure 3: AgentMaster demonstration example and verification.

3.6 Design Considerations

The A2A-MCP design emphasizes modularity, extensibility, and reproducibility. New retrieval agents can be integrated without modifying the orchestration logic. The standardized JSON-RPC interfaces facilitate consistent communication across agents (Zhang et al., 2025a). This architecture provides a flexible foundation for retrieval-augmented conversational systems and supports future research into multi-agent LLM collaboration.

4 Experimental Results and Evaluation

In this case study, the AgentMaster system is deployed locally as well as on AWS to facilitate internet access. Each agent leverages OpenAI's GPT-4o mini model. Three domain agents, derived from our prior research, focus on Structured Query Language (SQL) (Gadiraju et al., 2025), Information Retrieval (IR) (Gadiraju et al., 2024, 2025), and image analysis (Darji et al., 2024), utilizing the Federal Highway Administration (FHWA) public datasets (Federal Highway Administration, 2025).

Experiments were conducted to evaluate both individual agents and agent-to-agent collaborations using simple and complex queries. Multiple evaluation metrics are employed to assess the multi-agent system, including agentic metrics, LLM-as-a-Judge (Zheng et al., 2023), and human evaluation. Agentic metrics assess AI agents' autonomy and effectiveness in complex tasks. LLM-as-a-Judge uses a large language model to evaluate another LLM's outputs for correctness, relevance, and coherence. Human evaluation remains the gold standard for validating these assessments in this pilot study.

4.1 Individual Agent Evaluation

Three domain agents (SQL, IR, and Image) were previously evaluated independently in our past research and demonstrated high reliability and accuracy (Gadiraju et al., 2024, 2025; Darji et al., 2024). Additionally, due to the robustness of the GPT model, individual queries or single tasks have consistently yielded correct results in our testing. However, there are occasional instances of misclassifying single queries as complex queries for query decomposition, resulting in incorrect responses.

4.2 Complex Task Evaluation

To evaluate the quality and accuracy of AgentMaster's responses, sub-questions decomposed from complex queries were individually submitted to AgentMaster. The outputs generated for these simpler sub-questions were then compared to the corresponding segments within AgentMaster's responses to the overall complex queries. Since the sub-questions are simple queries, it would not require mutli-agent collaboration and thus can serve as a valid verification method for AgentMaster's output.

Figure 3 presents the front-end and back-end of the demonstration, as well as the verification of

ID	Num of Sub-Questions	Assigned Agents
Q1	3	General, SQL, SQL
Q2	3	IR, SQL, SQL
Q3	5	IR, SQL, IR, SQL
Q4	3	SQL, SQL, IR
Q5	2	SQL, General
Q6	8	8 IRs

Table 1: The number of query decompositions and the corresponding path for each complex query.

Table 2: Evaluation Metrics by Query Type

Query Type	G-Eval (%)	BERT Precision (%)	BERT F1 (%)
SQL Queries	92.0	98.8	98.7
IR Queries	90.2	97.6	97.8
General QA	84.0	95.7	96.8
Image/Complex QA	82.0	90.1	91.9
Average	**87.1**	**95.6**	**96.3**

AgentMaster's generated response. As shown in Figure 3a, AgentMaster responds with a domain-specific full response to a complex user query by providing a combination of relevant specific information from the database and general information. Figure 3c displays the coordinator agent decomposing the complex query into sub-questions before assigning each sub-question to the appropriate agents. In the example, the general agent and the SQL agent were employed to generate partial responses, which are sent back to the coordinator to integrate them into a cohesive final response. Additionally, in Figure 3b, the corresponding sub-questions were submitted to AgentMaster to validate the complex query results, and the simple query results were found to be consistent with the information in the complex query responses. AgentMaster was queried for the total number of bridges built in Virginia and those built in Virginia in 2019, and correct information was provided, indicating accurate routing of the complex query and successful SQL database retrieval. Similarly, Figure 12 in Appendix A displays a complex query evaluation, verifying the reliability of AgentMaster.

As shown in Table 1, six complex queries were submitted to AgentMaster for evaluation. For each question, the number of sub-questions from the coordinator agent's query decomposition and the respective agents are automatically assigned to tasks according to their capabilities. The specific wording of the queries is provided in Appendix A. Human evaluation, based on the agentic metrics comprised of task completion and correction, revealed that each complex query was correctly decomposed, with most agent task paths correctly assigned.

4.3 Overall Evaluation

The overall A2A-MCP framework was evaluated across multiple dimensions, including factual correctness, relevance, completeness, and semantic similarity. Metrics included Answer Relevancy, Hallucination detection, G-Eval (LLM-based assessment) (Liu et al., 2023), and BERTScore (Zhang et al., 2024). The test set comprised diverse queries spanning SQL retrieval, IR, general knowledge, and summarization.

Table 2 reports the aggregated metrics across all query types. Overall, the system demonstrated strong correctness and semantic alignment, with Answer Relevancy and Hallucination metrics indicating high reliability across domains. The average G-Eval score for complex queries exceeded 87.1%, while BERTScore F1 averaged 96.3%, reflecting high semantic fidelity to reference outputs.

During individual agent evaluation, the SQL Agent and IR Agent produced consistently accurate results, while the General Agent and Image Agent showed minor variability due to open-ended generation. Evaluation of complex queries confirmed effective decomposition and integration by the Coordinator Agent, with most sub-questions yielding outputs consistent with the composite responses.

5 Conclusions

This paper presents AgentMaster, a novel modular conversational framework leveraging A2A-MCP protocols for retrieval-augmented question answering across structured, unstructured, and multimodal data sources. By interacting with AgentMaster using natural language communication, users can receive domain-specific information regardless of expertise. Experimental results demonstrate the system's ability to effectively coordinate specialized agents and produce accurate, semantically faithful responses. The BERTScore F1 and LLM-as-a-Judge metric G-Eval averaged 96.3% and 87.1%. The proposed architecture highlights the potential of agent-based orchestration for scalable, domain-adaptive conversational AI.

6 Limitations

While the framework achieved strong performance across diverse query types, some limitations remain. The accuracy of retrieval and generation is partly constrained by the underlying LLM and retrieval corpus. Occasional misclassification of query complexity can lead to unnecessary decomposition or incomplete responses. Limited inter-agent collaboration and the constrained size of the database occasionally led to responses with minimal informational depth. The LLM-based reasoning process may also encounter challenges in synthesizing complex information. While LLM-as-a-judge evaluation offers scalability and efficiency, it remains limited by potential biases, lack of task-specific expertise, and alignment with human judgment. Finally, the current framework lacks established security safeguards for information storage and usage. These limitations can be addressed in future work.

7 Acknowledgments

The authors would like to thank the Federal Highway Administration (FHWA) for providing public datasets used to build knowledge databases for the case study.

References

Javad Aminian-Dehkordi, Mohammad Parsa, Mohsen Naghipourfar, and Mohammad R.K. Mofrad. 2025. Koda: An agentic framework for kegg orthology-driven discovery of antimicrobial drug targets in gut microbiome. *bioRxiv*.

Anthropic. 2024. Introducing the model context protocol.

Viraj Nishesh Darji, Callie C. Liao, and Duoduo Liao. 2024. Automated interpretation of non-destructive evaluation contour maps using large language models for bridge condition assessment. In *2024 IEEE International Conference on Big Data (BigData)*, pages 3258–3263.

Yubo Dong, Xukun Zhu, Zhengzhe Pan, Linchao Zhu, and Yi Yang. 2024. Villageragent: A graph-based multi-agent framework for coordinating complex task dependencies in minecraft. *Preprint*, arXiv:2406.05720.

Hung Du, Srikanth Thudumu, Rajesh Vasa, and Kon Mouzakis. 2025. A survey on context-aware multi-agent systems: Techniques, challenges and future directions. *Preprint*, arXiv:2402.01968.

Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. 2025. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *Preprint*, arXiv:2505.02279.

Federal Highway Administration. 2025. Fhwa infotechnology portal. Accessed: 2025-7-4.

Sai Surya Gadiraju, Zijie He, and Duoduo Liao. 2025. A conversational agent framework for multimodal knowledge retrieval: A case study in fhwa infohighway web portal queries. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025) Workshop for Research on Agent Language Models (REALM 2025)*, Vienna, Austria. ACL.

Sai Surya Gadiraju, Duoduo Liao, Akhila Kudupudi, Santosh Kasula, and Charitha Chalasani. 2024. InfoTech Assistant: A Multimodal Conversational Agent for InfoTechnology Web Portal Queries . In *2024 IEEE International Conference on Big Data (BigData)*, pages 3264–3272, Los Alamitos, CA, USA. IEEE Computer Society.

Gemini. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *Preprint*, arXiv:2402.01680.

Jennifer Haase and Sebastian Pokutta. 2025. Beyond static responses: Multi-agent llm systems as a new paradigm for social science research. *Preprint*, arXiv:2506.01839.

Idan Habler, Ken Huang, Vineeth Sai Narajala, and Prashant Kulkarni. 2025. Building a secure agentic ai application leveraging a2a protocol. *Preprint*, arXiv:2504.16902.

Li Hu, Guoqiang Chen, Xiuwei Shang, Shaoyin Cheng, Benlong Wu, Gangyang Li, Xu Zhu, Weiming Zhang, and Nenghai Yu. 2025. Compileagent: Automated real-world repo-level compilation with tool-integrated llm-based agent system. *Preprint*, arXiv:2505.04254.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. 2024. QueryAgent: A reliable and efficient reasoning framework with environmental feedback based self-correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5014–5035, Bangkok, Thailand. Association for Computational Linguistics.

Md. Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. MapCoder: Multi-agent code generation for competitive problem solving. In

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4912–4944, Bangkok, Thailand. Association for Computational Linguistics.

Cheonsu Jeong. 2025. A study on the mcp x a2a framework for enhancing interoperability of llm-based autonomous agents. *Preprint*, arXiv:2506.01804.

Naveen Krishnan. 2025. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *Preprint*, arXiv:2504.21030.

Sonu Kumar, Anubhav Girdhar, Ritesh Patil, and Divyansh Tripathi. 2025. Mcp guardian: A security-first layer for safeguarding mcp-based ai system. *Preprint*, arXiv:2504.12757.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2025. Agent-oriented planning in multi-agent systems. *Preprint*, arXiv:2410.02189.

Hongzhan Lin, Yang Deng, Yuxuan Gu, Wenxuan Zhang, Jing Ma, See-Kiong Ng, and Tat-Seng Chua. 2025. Fact-audit: An adaptive multi-agent framework for dynamic fact-checking evaluation of large language models. *Preprint*, arXiv:2502.17924.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Qinyu Luo, Yining Ye, Shihao Liang, Zhong Zhang, Yujia Qin, Yaxi Lu, Yesai Wu, Xin Cong, Yankai Lin, Yingli Zhang, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024. RepoAgent: An LLM-powered open-source framework for repository-level code documentation generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 436–464, Miami, Florida, USA. Association for Computational Linguistics.

Puneet Mathur, Alexa Siu, Nedim Lipka, and Tong Sun. 2024. MATSA: Multi-agent table structure attribution. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 250–258, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1088–1113, Bangkok, Thailand. Association for Computational Linguistics.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, Shilong Liu, Xun Jiang, Liu Leqi, and Mengdi Wang. 2025. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *Preprint*, arXiv:2506.14728.

Anjana Sarkar and Soumyendu Sarkar. 2025. Survey of llm agent communication with mcp: A software design pattern centric review. *Preprint*, arXiv:2506.05364.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Preprint*, arXiv:2303.17580.

Rao Surapaneni, Miku Jha, Michael Vakoc, and Todd Segal. 2025. Announcing the agent2agent protocol (a2a). Google Developer Blog.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Dean Lawrence Williams. 2025. Multi-agent communication protocol in collaborative problem solving: A design science approach.

Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, Weiwen Liu, Ying Wen, Yong Yu, and Weinan Zhang. 2025. A survey of ai agent protocols. *Preprint*, arXiv:2504.16736.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan W. Suchow, Zhenyu Cui, Rong Liu, Zhaozhuo Xu, Denghui Zhang, Koduvayur Subbalakshmi, Guojun Xiong, Yueru He, Jimin Huang, Dong Li, and Qianqian Xie. 2025. Fincon: a synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Wentao Zhang, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. 2025a. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *Preprint*, arXiv:2506.12508.

Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Yao Zhang, Zaixi Shang, and Silpan Pateland Mikel Zuniga. 2025b. From unstructured communication to intelligent rag: Multi-agent automation for supply chain knowledge bases. *Preprint*, arXiv:2506.17484.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A Appendix

The appendix contains more demonstrations of multimodal inputs and outputs, single and complex queries, and different usages of the domain-specific agents.

A.1 AgentMaster Demonstration Displays and Examples

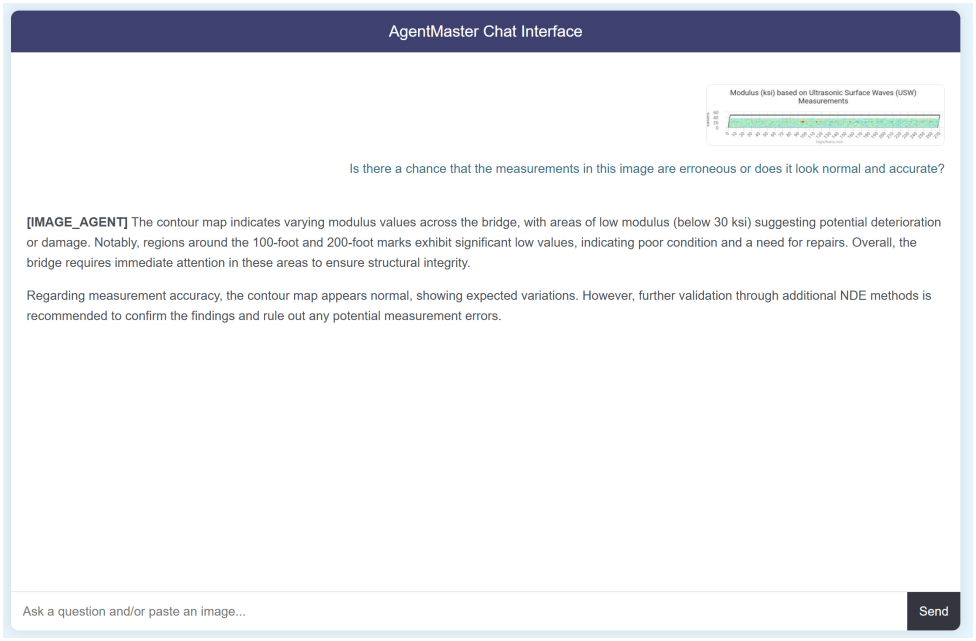


Figure 4: Frontend example of a single query for an image agent.

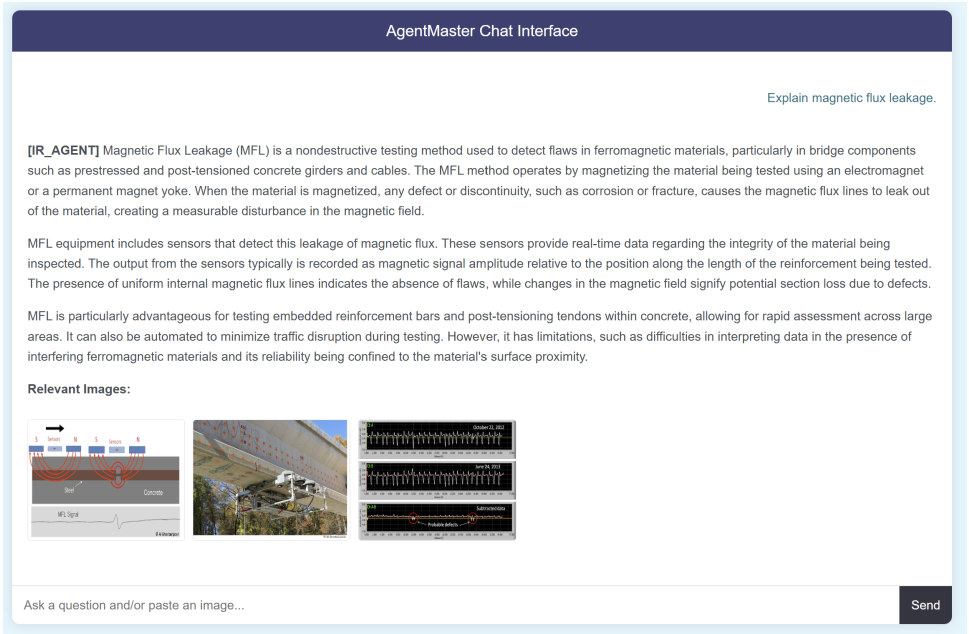
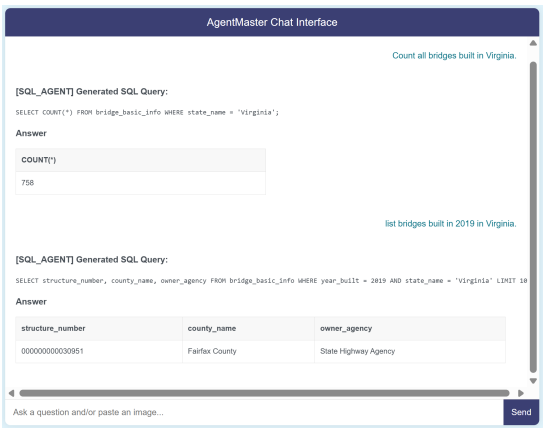
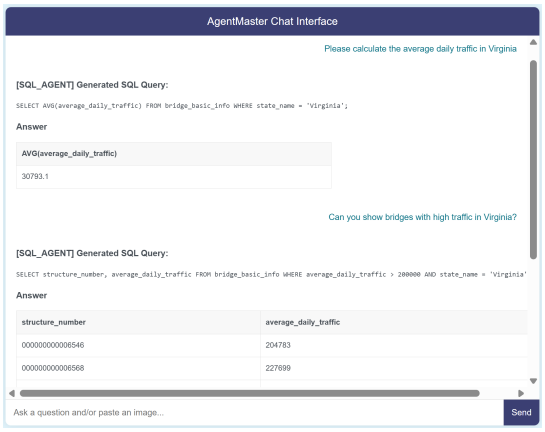


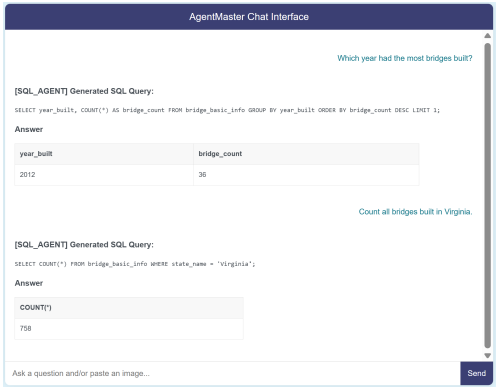
Figure 5: Frontend example of a single query for an IR agent.



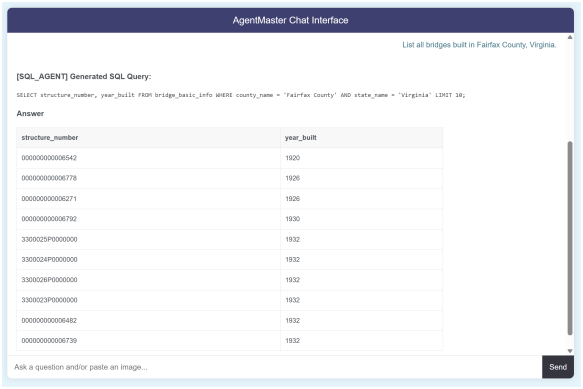
(a) Frontend example of two single queries for the SQL agent.



(b) Frontend example of two single queries for the SQL agent.

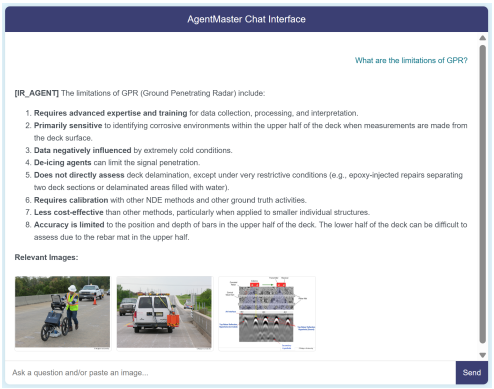


(c) Frontend example of two single queries for the SQL agent.



(d) Frontend example of one single query for the SQL agent.

Figure 6: More demonstration examples of a single query for the SQL agent.



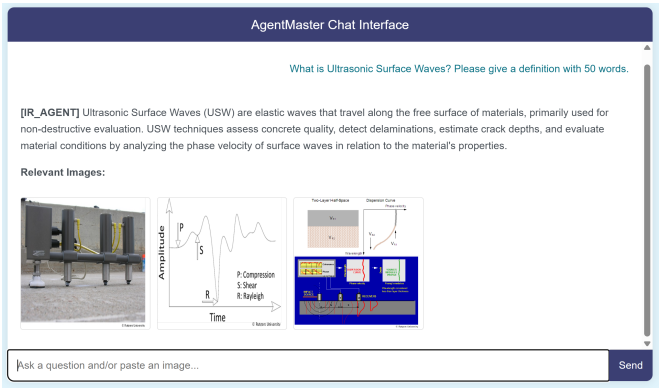


Figure 7: Two examples of single verification queries for the IR agent.

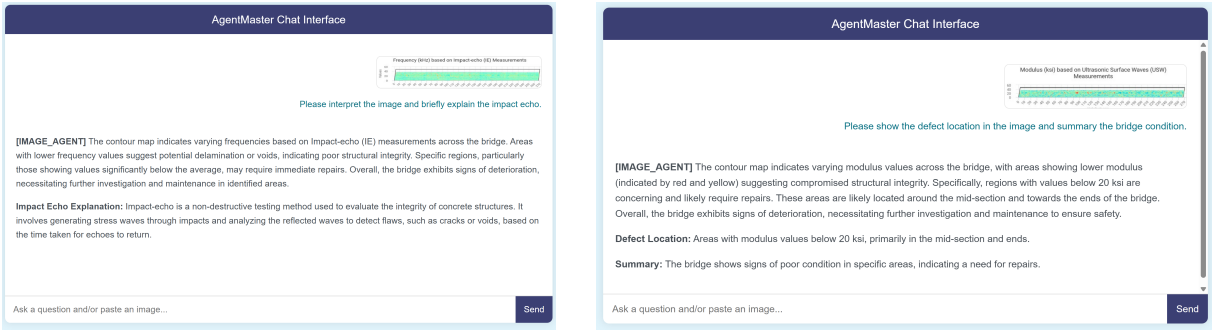
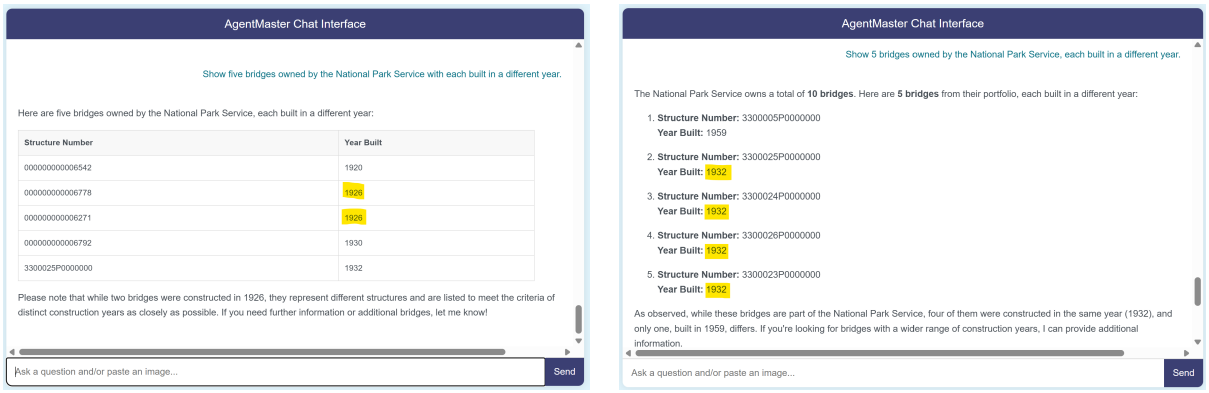


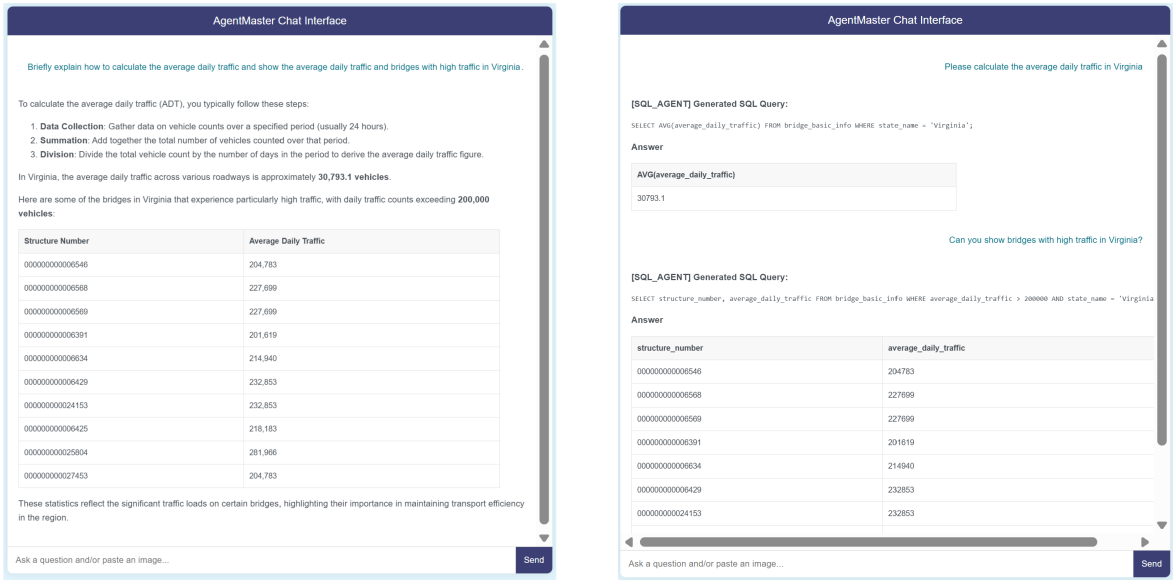
Figure 8: Two examples of single verification queries for the image agent.



(a) A single query for verification.

(b) A single query for verification.

Figure 9: SQL agent. The highlighted text represent errors in SQL information retrieval. There is duplicate content in them, indicating that AgentMaster can still be erroneous.



(a) Example of a complex query.

(b) Single queries for verification.

Figure 10: An additional AgentMaster demonstration example and verification.

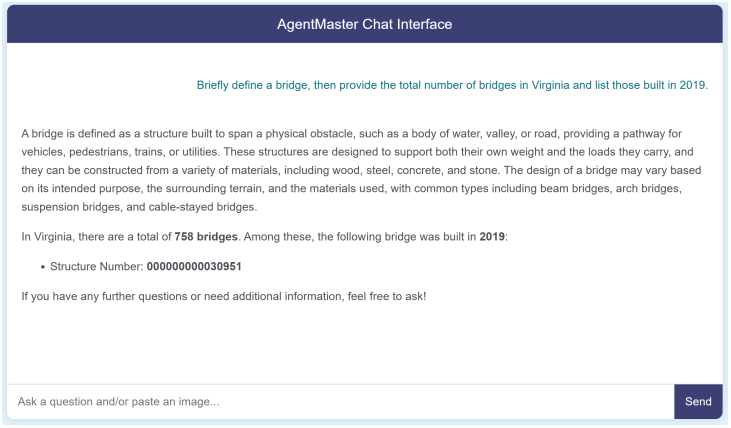
A.2 Agent Task Decomposition and Routing for Complex Queries

A.2.1 Complex Queries

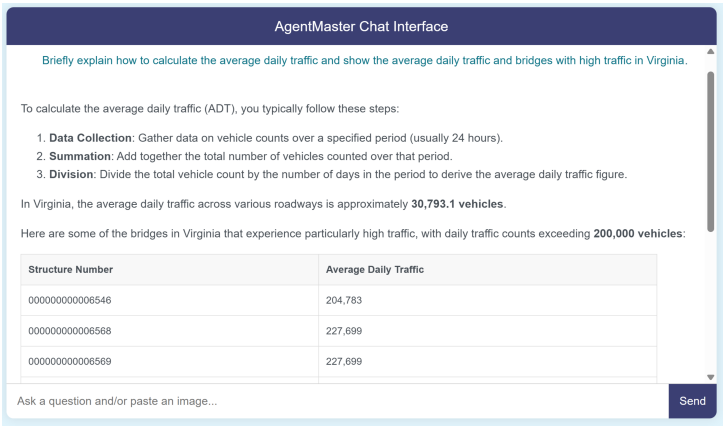
ID	Complex Query
Q1	Briefly define a bridge, then provide the total number of bridges in Virginia and list those built in 2019.
Q2	Briefly explain how to calculate the average daily traffic and show the average daily traffic and bridges with high traffic in Virginia.
Q3	What is the typical service lifespan of a bridge? Identify and display all bridges that exceed this average age.
Q4	List the three oldest bridges in Virginia, show their year built from the database, and briefly explain why their maintenance costs tend to be higher according to engineering guidelines with 50 words.
Q5	List five of the oldest bridges in the United States still in use today, and briefly describe their historical significance.
Q6	Compare the advantages and disadvantages of concrete arch bridges and steel truss bridges in terms of maintenance, lifespan, and load capacity.

Table 3: The complex questions and the corresponding IDs, matching Table 1.

A.2.2 AgentMaster Outputs of Complex Queries in the Chat Interface

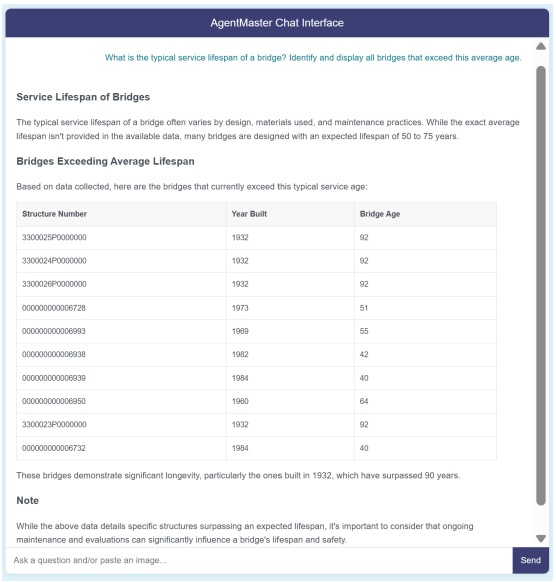


(a) Complex query Q1.

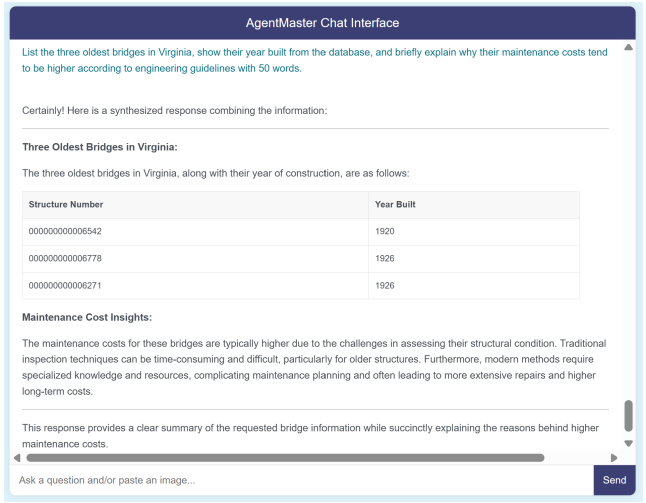


(b) Complex query Q2.

Figure 11: The display of the frontend response for each complex query.

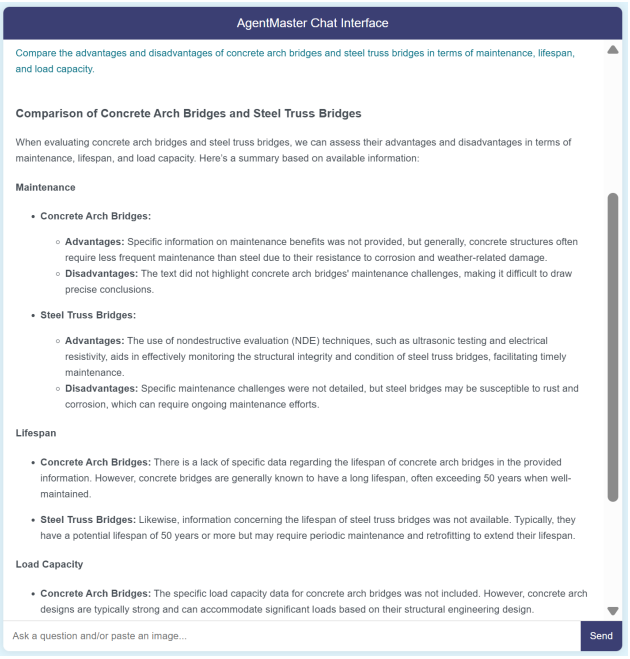


(a) Complex query Q3.



(b) Complex query Q4.

(c) Complex query Q5.



(d) Complex query Q6.

Figure 12: The display of the frontend response for each complex query continued.

A.2.3 Decomposition and Routing of Complex Queries in the Backend

ID	Decomposition and Routing Information
Q1	[Agent Server] Received complex query for coordinator: 'Briefly define a bridge, then provide the total number of bridges in Virginia and list those built in 2019.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What is a brief definition of a bridge?', 'What is the total number of bridges in Virginia?', 'Which bridges in Virginia were built in 2019?'] [Coordinator] Routing sub-question: 'What is a brief definition of a bridge?' [Complex to Single Query Router] Decision for GENERAL_AGENT [Coordinator] Routing sub-question: 'What is the total number of bridges in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT COUNT(*) FROM bridge_basic_info WHERE state_name = 'Virginia';" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'Which bridges in Virginia were built in 2019?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, year_built FROM bridge_basic_info WHERE state_name = 'Virginia' AND year_built = 2019 LIMIT 10;" Detected valid SQL, executing... [Coordinator] Synthesizing final answer...
Q2	[Agent Server] Received complex query for coordinator: 'Briefly explain how to calculate the average daily traffic and show the average daily traffic and bridges with high traffic in Virginia?' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['How do you calculate the average daily traffic?', 'What is the average daily traffic in Virginia?', 'Which bridges in Virginia have high traffic?'] [Coordinator] Routing sub-question: 'How do you calculate the average daily traffic?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the average daily traffic in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT AVG(average_daily_traffic) FROM bridge_basic_info WHERE state_name = 'Virginia';" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'Which bridges in Virginia have high traffic?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, average_daily_traffic FROM bridge_basic_info WHERE state_name = 'Virginia' AND average_daily_traffic > 200000 LIMIT 10;" Detected valid SQL, executing... [Coordinator] Synthesizing final answer...
Q3	[Agent Server] Received complex query for coordinator: 'What is the typical service lifespan of a bridge? Identify and display all bridges that exceed this average age.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What is the typical service lifespan of a bridge?', 'What is the average age of bridges?', 'What bridges exceed the typical service lifespan?', 'How can I display the bridges that exceed the average age?'] [Coordinator] Routing sub-question: 'What is the typical service lifespan of a bridge?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the average age of bridges?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: 'The average age of bridges cannot be determined with a single SQL query as it requires an aggregate function.' No SQL detected, returning text answer. [Coordinator] Routing sub-question: 'What bridges exceed the typical service lifespan?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'How can I display the bridges that exceed the average age?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: 'SELECT structure_number, year_built, bridge_age FROM bridge_basic_info WHERE bridge_age > (SELECT AVG(bridge_age) FROM bridge_basic_info) LIMIT 10;' Detected valid SQL, executing... [Coordinator] Synthesizing final answer...

Table 4: The decomposition and routing information in the backend for each complex question, matching Table 1.

ID	Decomposition and Routing Information
Q4	[Agent Server] Received complex query for coordinator: 'List the three oldest bridges in Virginia, show their year built from the database, and briefly explain why their maintenance costs tend to be higher according to engineering guidelines with 50 words.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What are the three oldest bridges in Virginia?', 'What is the year built for each of the three oldest bridges in Virginia?', 'Why do the maintenance costs of these bridges tend to be higher according to engineering guidelines?'] [Coordinator] Routing sub-question: 'What are the three oldest bridges in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, year_built FROM bridge_basic_info WHERE state_name = 'Virginia' ORDER BY year_built ASC LIMIT 3;" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'What is the year built for each of the three oldest bridges in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, year_built FROM bridge_basic_info WHERE state_name = 'Virginia' ORDER BY year_built ASC LIMIT 3;" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'Why do the maintenance costs of these bridges tend to be higher according to engineering guidelines?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Synthesizing final answer...
Q5	[Agent Server] Received complex query for coordinator: 'List five of the oldest bridges in the United States still in use today, and briefly describe their historical significance.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What are the five oldest bridges in the United States still in use today?', 'What is the historical significance of each of the five oldest bridges in the United States?'] [Coordinator] Routing sub-question: 'What are the five oldest bridges in the United States still in use today?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: 'SELECT structure_number, year_built FROM bridge_basic_info WHERE year_built < 1970 ORDER BY year_built ASC LIMIT 5;' Detected valid SQL, executing... [Coordinator] Routing sub-question: 'What is the historical significance of each of the five oldest bridges in the United States?' [Complex to Single Query Router] Decision for GENERAL_AGENT [Coordinator] Synthesizing final answer...
Q6	[Agent Server] Received complex query for coordinator: 'Compare the advantages and disadvantages of concrete arch bridges and steel truss bridges in terms of maintenance, lifespan, and load capacity' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What are the advantages of concrete arch bridges in terms of maintenance?', 'What are the disadvantages of concrete arch bridges in terms of maintenance?', 'What is the lifespan of concrete arch bridges?', 'What is the load capacity of concrete arch bridges?', 'What are the advantages of steel truss bridges in terms of maintenance?', 'What are the disadvantages of steel truss bridges in terms of maintenance?', 'What is the lifespan of steel truss bridges?', 'What is the load capacity of steel truss bridges?'] [Coordinator] Routing sub-question: 'What are the advantages of concrete arch bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What are the disadvantages of concrete arch bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the lifespan of concrete arch bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the load capacity of concrete arch bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What are the advantages of steel truss bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What are the disadvantages of steel truss bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the lifespan of steel truss bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the load capacity of steel truss bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Synthesizing final answer...

Table 5: The decomposition and routing information in the backend for each complex question continued, matching Table 1.