Explainable Natural Language Query Interface for Relational Databases Using a Multi-Agent System

Group 9 Proposal - CITS5553 (Client: Sirui Li, Siwen Luo)

1. BACKGORUND & VALUE PROPOSITION

1.1 Aim

Develop an explainable, multi-agent tool that allows user to explore databases via natural language queries, providing visualisations and justification for the query output.

1.2 Background & Value Proposal

Relational databases remain central to enterprise data management, offering structured storage and reliable performance. Recent advancements in large language models (LLMs) have enabled tools that allow users to query databases in natural language, alleviating the need for technical SQL and data warehousing expertise, such capabilities are increasingly integrated into platforms like Power BI and Tableau [1]. However, these approaches often lack transparency and explainability, limiting their suitability in enterprise contexts where trust and accountability are essential. In addition to functioning as "black boxes," these end-to-end models often struggle to adapt to different database scenarios, limiting their flexibility and practical applicability in enterprise settings [2].

For the project, we will use the Spider v1 dataset [3], a large-scale complex and cross-domain dataset annotated by 11 Yale students. We will focus on the easy-median questions, and use 'dataset split', ensuring that all question for a given database remain in the same split. This setup attempts our system's capability to handle not only new questions but also generalise to new databases and domains.

Key Challenges

- Query results must be transparent, enabling users to understand how each answer is generated rather than receiving only the final output.
- The framework for producing explainable results should be modular, allowing easy adaptation to different scenarios. Where adjustments are required, it should support fine-tuning of individual models rather than retraining the entire system.

To address these challenges, we propose an explainable Natural Language Interface for Database Exploration. The system would take a natural language query as input and invoke multiple agents to perform specialised tasks, making it easy to adapt to different database schemas by fine-tuning or replacing individual modules. By providing explanations for each agent's results and presenting visualisations of relevant tables and records rather than only a final answer, the framework ensures transparency, adaptability, and actionable insights for enterprise users.

Key Benefits

- A multi-Agent framework that is easily adaptable to different database scenarios, requiring only fine-tuning or replacement of individual modules instead of retraining the whole system.
- Interpretable outputs, presenting results through visualisations of relevant tables and records rather than only returning a final answer.
- A clear focus on adaptability and explainability as the core strengths of the proposed approach.

---

2. DELIVERABLES & TIMELINE & COSTS

2.1 Deliverables

1. Deliver an intelligent and explainable database exploration tool that allows users to input natural language queries and receive not only accurate results, but also a transparent reasoning process. This includes highlighting relevant tables and records and providing insights into why specific records were selected and presented.

2. Deploy an interactive web-based dashboard where users can input natural language queries and receive relevant tables with highlighted records and explanations for table/record selection.
3. Produce a comprehensive GitHub repository containing modular implementations of all agents, complete with data processing scripts, training logs, and evaluation reports, ensuring that any developer can reproduce results.
4. Deliver an oral presentation and live demo showcasing system functionality, explainability, and user interface, along with a summary of the technical approach and achieved outcomes.

2.2 Timeline*

Week(s)	Task Name	Description
W3-W4	Research & Scoping	Investigate MAS architectures (e.g., LangChain, custom pipelines), define communication/orchestration approach, select tools. Collaborate with client to refine objectives, scope, deliverables, and submit proposal.
W5	Agents Development	Individually train and test agents (Schema Intelligence, NL Understanding, DB/Table Selection, SQL Generation, Record Discovery) and link schema embeddings to DB Selector.
W6	Pipeline Integration	Connect agents into an initial sequential pipeline with basic data flow.
W7	CLI Prototype	Deliver a working command-line version handling basic queries.
W8	UI Integration	Build and connect a web dashboard to the backend pipeline.
W8–W10	Iterative	Improve accuracy and implement query memory.
W10–W12	Finalisation	Polish code/UI, prepare presentation, and rehearse demonstration.

**Please refer to Appendix for Deliverable Gantt Chart.*

2.3 Costs

Monetary: There will be no costs incurred by the team. The only expense will be running our agents, for which the client has already provided an API key. We will be using GPT-5 mini, which costs USD $0.25 per 1M input tokens and USD $10.00 per 1M output tokens.

Time: The project is estimated to require a total of 400 hours of work from data scientists. This work is spread out among team members and weeks 5 to 12.

---

3. METHODS

The project source code and documentation, including design documents, proposals, and final paper, will be managed using GitHub version control system, with built-in product managing features like tasks - issues, and virtual co-workspace.

The Spider dataset contains 200 SQLite databases across 138 domains. Each database folder contains a single SQLite file with its corresponding SQL creation script. On average, each database in Spider has 28 columns and 9 foreign keys [3].

The dataset also provides multiple JSON files describing all database table schemas, along with 10,181 questions and 5,693 complex SQL queries.

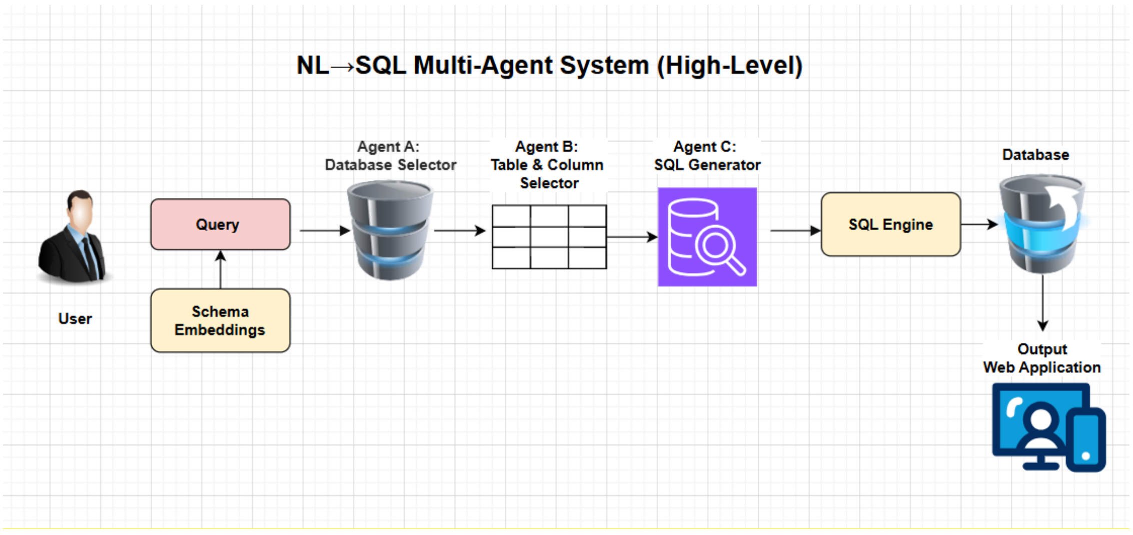
Task 1: Design and deploy the core modules of the Agentic AI application
The project will experiment with multi-agent system designs to optimise the performance of an AI application that converts natural language queries into SQL and provides explainable reasoning.

Multiple strategies will be explored, including but not limited to:

1. Few-shot prompting, database-aware (DAIL) [1].
2. Schema linking with explicit linking prompts and foreign-key surfacing [1], [4].
3. Decomposition prompts [4].
4. Self-correction prompts [4].
5. Agentic planning and tooling [2], [5].

To implement these strategies, the core system may follow a prototype architecture consisting of:

- Database Schema Embedding: All available database schemas are embedded to enable semantic understanding and similarity-based selection.
- Agent A: Database Selector: Chooses the most relevant database based on the user's natural language query.
- Agent B: Table & Column Selector: Identifies the appropriate tables and columns within the selected schema that are needed to answer the query.
- Agent C: SQL Generator: Generates an SQL query tailored to the selected tables and columns.
- Execution & Explanation: Executes the generated SQL query and consolidates the outputs from all agents. The user interface visualises and explains the reasoning process, showing both the data results and the steps behind the selection and generation process.


NL→SQL Multi-Agent System (High-Level)
User
Query
Schema Embeddings
Agent A: Database Selector
Agent B: Table & Column Selector
Agent C: SQL Generator
SQL Engine
Database
Output Web Application

Optional editable reasoning will be implemented to allow users to modify the reasoning steps of the AI agents, enhancing both the explainability and the performance of the system [6], [7].

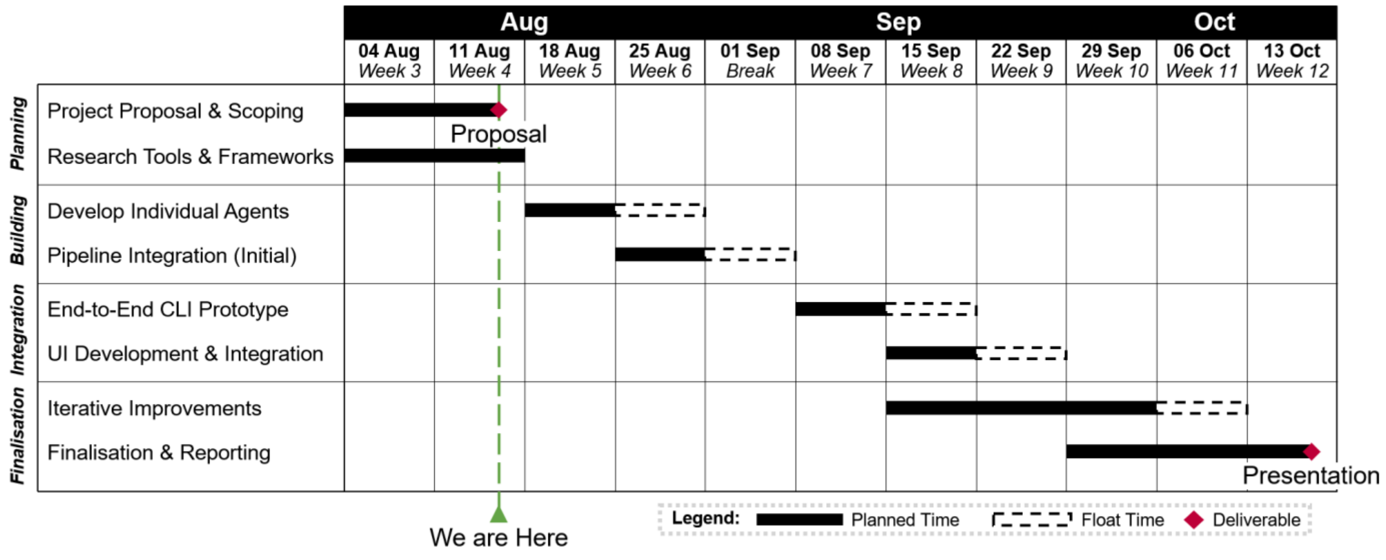
Task 2: Evaluate and optimise the performance of the designs
The performance of the Agentic AI application will be evaluated through a series of combinations by models and agentic AI techniques based on the Spider version 1 dataset's metrics that been proposed in the dataset's paper [3], [1]. Metrics that will be used are Exact-Match, Execution Accuracy, and other relevant metrics [1], [4].

Task 3: Design and deploy the explainable functionality for the Agentic AI application in the GUI of the application
To overcome the limited clarity of the reasoning process in the Agentic AI application, we will implement explainable function in the GUI of the application. These explanations will provide users with insights into the reasoning process of the AI agents, enhancing transparency and user understanding. The design may explore various visualisation techniques, such as reasoning paths and tree, along with result presentation like chart, data table, SQL tables or other formats that can effectively increase the explainable capacity of the AI agents [6], [7].

Task 4: Evaluate the effectiveness of the explanation in the Agentic AI application in enhancing the AI agents' performance and user satisfaction.
The effectiveness of the explanation will be evaluated through user studies and performance metrics using surveying techniques. User studies will involve gathering feedback from users on the clarity and usefulness of the explanations provided by this interactive tool as well as their impact on user satisfaction and trust in the AI agents. Each of the metrics will be evaluated and the overall feedback also collected. The space of evaluation samples will include a diverse range of user interactions and scenarios, based on the Spider data set and SQL potential tasks to ensure comprehensive evaluation [6], [7].

APPENDIX

A. Deliverable Gantt Chart



B. Team Members

- Cedrus Dang (24190901)
- Franco Meng (23370209)
- Aswathy Mini Sasikumar (24331072)
- Laine Mulvay (22708032)
- Nirma Rajapaksha Senadherage (24268122)
- RuiZhe Wang (24260749)

REFERENCES

[1] D. Gao *et al.*, “Text-to-SQL empowered by large language models: a benchmark evaluation,” *arXiv.org*, Aug. 29, 2023. https://arxiv.org/abs/2308.15363

[2] S. Xue *et al.*, “Demonstration of DB-GPT: Next generation Data Interaction System empowered by large language models,” *arXiv.org*, Apr. 16, 2024. https://arxiv.org/abs/2404.10209

[3] T. Yu *et al.*, “Spider: a Large-Scale Human-Labeled dataset for complex and Cross-Domain semantic parsing and Text-to-SQL task,” *arXiv.org*, Sep. 24, 2018. https://arxiv.org/abs/1809.08887

[4] M. Pourreza and D. Rafiei, “DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction,” *arXiv.org*, Apr. 21, 2023. https://arxiv.org/abs/2304.11015

[5] B. Wang *et al.*, “MAC-SQL: a Multi-Agent collaborative framework for Text-to-SQL,” *arXiv.org*, Dec. 18, 2023. https://arxiv.org/abs/2312.11242

[6] T. Wu, M. Terry, and C. J. Cai, “AI chains: Transparent and controllable Human-AI interaction by chaining large language model prompts,” *arXiv.org*, Oct. 04, 2021. https://arxiv.org/abs/2110.01691

[7] R. Y. Pang *et al.*, “Interactive Reasoning: Visualizing and controlling Chain-of-Thought reasoning in large language models,” *arXiv.org*, Jun. 30, 2025. https://arxiv.org/abs/2506.23678