LLM-Powered Multi-Agent Framework for Quality Improvement in Healthcare

BT5153 Applied Machine Learning in Business Analytics : Group 02
Benjamin Zhou Ming Xi (A0091098X), Kim Yeontae (A0280486W),
Neo Yuan Khai (A0143408L), Roy Yeo Fu Qiang (A0280541L), Yang Fan (A0139307E)

Abstract

This project presents an LLM-powered system designed to enhance healthcare quality improvement (QIP) workflows by bridging gaps in data analysis and evidence synthesis among clinicians. Leveraging a synthesized hospital SQL database and a vectorized archive of historical QIP posters, two specialized agents were developed: a Poster RAG agent for document retrieval and summarization, and a Text-to-SQL agent for structured data querying. Both agents are orchestrated through LangGraph, a free and open-source framework for building graph-based, multi-agent workflows with persistent state management. Rigorous evaluation using LangSmith observability tools, RAGAS retrieval metrics, and SPIDER-based SQL benchmarks demonstrate the system's capability to provide accurate, context-grounded insights, while highlighting areas for future improvement in scalability, multimodal retrieval, and faithfulness optimization

1. Introduction

1.1 Background

Within Singapore's dynamic healthcare ecosystem, hospitals churn out dozens of quality improvement projects (QIPs) annually as part of medical residency training or institutional performance initiatives. The culmination of these dedicated efforts involves the presentation of a proposal or a written report at a healthcare conference.

However, it is known in healthcare that most QIPs fail to deliver sustainable results (Ivers et al., 2014) with strong implementation – with one key reason being the poor execution of identified methodology (Larson et al., 2020). Clinicians are required to define and measure the problem before proceeding with their initiatives, but many personnel lack data expertise (Celi et al., 2016) to continue with effective implementations. Coupled with heavy clinical and administrative workloads, projects often begin without sufficient preparation leading to a high failure rate.

This project aims to streamline data-driven decision-making by leveraging Large Language Model (LLM)-powered solutions to bridge the afore-mentioned skill gaps in data analysis and literature review among healthcare professionals. By improving access to critical data and research, the proposed solution minimizes redundant efforts, reduces costs, and enhances the efficiency of quality improvement initiatives.

1.2 Proposed Solution

The goal is to develop an intuitive LLM-powered chat interface that facilitates clinicians to streamline and optimize their research workflow by providing 2 key outputs: summaries of past similar projects based on the input text queries related to specific QIPs and historical hospital performance metrics. The integration of dual functionality streamlines the user workflow by consolidating research tasks within a unified platform, eliminating the need to navigate between different tools and promotes a more cohesive research experience.

Leveraging on a vector database indexed with embeddings of past QIP documents, the Poster RAG agent summarizes a list of similar internal projects to inform and contextualize tried and tested improvement efforts; while the Text-to-SQL agent provides data-driven support with the retrieval and summarization of historical hospital performance metrics relevant to the user's query extracted from the hospital's database.

While the two specialized agents are designed to perform individual workflows, the user can also choose to enable a sequential workflow that enables interaction between the Poster RAG agent and the Text-to-SQL agent. Leveraging the wealth of data contained within historical QIP documents, this connected workflow enables users to perform follow-up research by posing nuanced questions that connect the information from relevant posters with specific hospital data insights. Our solution is designed to facilitate generation of insightful connections between ideas and data findings, prioritizing transformation of abstract concepts into concrete, data-driven insights.

Considering our emphasis on leveraging internal reference materials together with the current workflow to integrate new QIP ideas with existing hospital data, the originally proposed web search agent – intended for the retrieval of

external information has been determined to fall outside the boundaries of the current project scope. The existing solution infrastructure is deemed sufficient for our target audience.

2. Dataset Description

2.1 SQL Tables

The hospital database utilized in this project consists entirely of synthesized data generated through SQL scripts based on mathematical distributions. All data was randomly and artificially created; no real patient information is used at any stage of this project. This synthetic dataset serves as the foundation for the Text-to-SQL component, demonstrating how natural language queries can be translated into SQL and executed against a structured database.

A total of 5 table schemas have been curated for the SQL database: `inpatient_cases`, `outpatient_cases`, `patient_incidents`, `patient_wait_times`, and `surgery_metrics`. These schemas capture critical data related to patient experiences and operational efficiency within a hospital system, focusing on the details for each patient encounters, adverse events, waiting times, and surgical procedures.

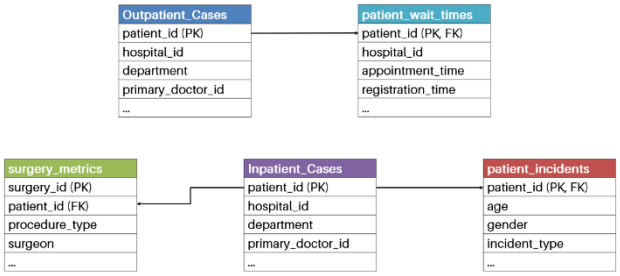


Figure 2.1.1. Sample Entity-Relationship (ER) Diagram of table schemas available in a hospital SQL database.

The `inpatient_cases` and `outpatient_cases` detail the specifics of each patient's admissions or visits. Both tables track essential medical diagnosis, treatment plans, attending physician details, and billing information with the notable difference of `inpatient_cases` recording hospital admission details while `outpatient_cases` records the visit date.

The `patient_incidents` table is specifically designed to record and track adverse events involving patients, capturing the type of incident, the date-time of occurrence, and information regarding the affected patients. This allows for the analysis of incidents in relation to inpatient stays, potentially identifying patterns or factors associated with specific patient populations or conditions relevant to quality improvement.

The `patient_wait_times` table provides a detailed breakdown of the time spent by outpatients at various stages of their visit – appointment, registration, triage, consultation, to billing and pharmacy. The captured timestamps at each key step along with the total time spent enables a granular analysis of patient flow and bottlenecks in the outpatient process.

Finally, the `surgery_metrics` table focuses on capturing a comprehensive set of data related to surgical procedures. Beyond surgery procedure information, it also includes critical surgery metrics such as blood loss, patient complications, and readmission rates.

An SQLite database file (`bt5153_gp.db`) was populated with generated data based on the defined schema structures. This database serves as a centralized repository that closely represents a hospital database, for data retrieval and analysis purposes to be used for the Text-to-SQL agent.

2.2 Medical Posters

The medical QI posters used in this project are publicly available from the Singapore Healthcare Management Congress: Singapore Healthcare Management Congress - Poster Exhibition (Singapore Health Services, n.d.).

31 medical posters from 2021 to 2024 across various hospitals have been systematically extracted and prepared for in-depth analysis in PDF format. These posters possess key attributes such as the originating hospital, thematic category, medical department, implementation results and key findings. The rich amount of metadata within each poster that can be extracted ensures that analysis can be focused on specific areas of interest, enabling comparative studies and identification of best practices within the specific area of interest in quality improvement initiatives across medical institutions.

Pre-processing of the documents involve a multi-stage approach to extract the structure and relevant information prior to the content conversion into vector embeddings. Recognizing that the posters are present in a variation of format, this poses a limitation to pre-process all posters with a standardized. As such, distinct methodologies are developed for more robust content extraction to accommodate the structural variations of the posters ensuring comprehensive and accurate chunking of information.

Following the pre-processing stage that identifies information segmented into crucial portions, the resulting metadata and structured textual content undergoes an embedding process and stored into a vector database for retrieval usage by the Poster RAG agent.

3. LLM overview & techniques

3.1 LLM Overview

Large Language Models (LLMs) have emerged as transformative tools for information processing and

decision support across numerous domains. Recent advances in mid-sized models like Llama 3 (Meta, 2024) and Mistral (Le Scao et al., 2023) have demonstrated that models with 7-13B parameters can achieve performance comparable to much larger counterparts when properly optimized. These developments have made sophisticated NLP capabilities more accessible for specialized applications like healthcare, where computational efficiency must be balanced with performance (Thirunavukarasu et al., 2023). LLMs offer particular value in healthcare quality improvement by automating information synthesis, standardizing knowledge extraction, and supporting decision-making—tasks that traditionally require extensive manual effort by clinicians with limited data analysis expertise (Singhal et al., 2023).

For the Poster RAG agent, we implement a Retrieval-Augmented Generation (RAG) architecture that grounds LLM outputs in domain-specific knowledge, reducing hallucinations while enhancing factual accuracy in healthcare contexts (Lewis et al., 2020). RAG systems have shown particular promise in healthcare applications where factual correctness is critical (Yang et al., 2025). The Poster RAG agent employs a dual chunking strategy—utilizing both full-poster and section-level chunks—to balance comprehensive context with precise information retrieval, following established practices in RAG system design (Gao et al., 2023).

For the Text-to-SQL agent, our implementation focuses on effective query generation with built-in error handling mechanisms. The agent first analyzes user input for relevance to the database schema, then generates SQL queries appropriate to the detected information needs. Recent studies highlight the importance of schema awareness in text-to-SQL systems (Liu et al., 2023), which we incorporate through explicit database structure analysis. A recursive refinement process improves query accuracy by providing error messages back to the query generation component for up to three retry attempts (Zeng et al., 2022). This approach significantly enhances the robustness of database interactions without requiring users to possess SQL expertise, addressing known challenges in text-to-SQL parsing robustness.

Agentic AI refers to advanced artificial intelligence systems where autonomous agents-often powered by large language models-can independently interpret context, make decisions, plan and execute multi-step actions, and adapt their behavior based on feedback and changing objectives, all with minimal human oversight. (IBM, 2025; Domo, 2025). Unlike traditional rule-based or generative AI, agentic AI systems are goal-oriented, capable of decomposing complex queries, iteratively refining their outputs, and integrating data from multiple sources or tools to achieve specific outcomes. This adaptability and autonomy allow agentic AI to handle ambiguous, multi-faceted, or evolving tasks, making it ideal for domains like healthcare, where evidence synthesis and dynamic problem-solving are crucial (Weights & Biases, 2025).

To operationalize agentic AI in our project, we adopt LangGraph — a specialized framework designed to build stateful, graph-based, multi-agent workflows for LLMs. In LangGraph, each agent or function is represented as a node within a directed graph, enabling flexible, non-linear execution with persistent memory and dynamic decision-making across the workflow. This architecture supports modularity, robust error handling, and human-in-the-loop interventions where needed. By leveraging LangGraph's capabilities, our system orchestrates specialized agents to deliver a highly adaptive, transparent, and resilient research support tool for healthcare quality improvement initiatives. The next section details the design and development of these agents within our platform.

3.2 LLM Application- Technical Stack

This project adopts a fully local, privacy-preserving architecture tailored for healthcare environments where data confidentiality is critical. The system integrates several core components. Weaviate, deployed via Docker, serves as the local vector database for storing and retrieving medical poster embeddings, utilizing the text2vec-ollama and generative-ollama modules. LangGraph is employed to build stateful, multi-agent workflows that coordinate the Retrieval-Augmented Generation (RAG) and Text-to-SQL agents. All language models are hosted locally through Ollama, enabling the use of models such as Llama 3 and Qwen2.5 (7B, 14B, and 32B variants) without making external API calls. This guarantees that no sensitive data leaves the local environment.

The user interface is powered by Streamlit, providing an intuitive, browser-based conversational interface that consolidates both poster retrieval and hospital data querying workflows. Hospital operational data is stored in a local SQLite database containing entirely synthetic information to simulate realistic query tasks without exposing real patient data. This fully localized design ensures end-to-end control, security, and compliance with strict healthcare data privacy requirements.

4. Agent Design & Development

4.1 Poster Search Agent

The Poster Search Agent is designed to provide users with relevant information retrieved from the database of Quality Improvement Posters (QIPs), leveraging a structured Langgraph workflow for efficient and accurate retrieval.

Traditional linear RAG workflows, which perform passive single-shot retrieval based on the user's query, are insufficient for healthcare QIP contexts characterized by ambiguous, multi-faceted, and under-specified questions. Our agentic RAG architecture instead introduces dynamic decision points: initial relevance validation, metadata-driven pre-filtering, retrieval quality grading, and iterative

query rewriting, enabling active correction and contextual refinement during retrieval. This agentic design sharply improves retrieval precision, semantic alignment, and robustness against query failure modes – critical in clinical research environments where irrelevant outputs erode user trust, waste clinician time, and undermine project outcomes. Although agentic flows increase system complexity and marginally impact latency, the ability to reason, adapt, and self-correct is essential for sustaining user confidence and delivering reliable support to QIP initiatives.

To optimize retrieval performance in our RAG system, we implemented a dual chunking strategy that balances semantic coherence with retrieval granularity. Recognizing variability in posters layouts (Appendix A2), two chunking methods were applied:

1) **Full-poster-level chunking**: Preserve complete project narratives, supporting complex cross-sectional queries, and at the
2) **Section-level chunking**: Enhance precision for fine-grained information needs (e.g., Introduction, Methods, Results). Layout parsing models (e.g. PDF layout models) and semantic segmentation techniques were employed to accurately detect section boundaries for section-level chunking, ensuring that critical information remains intact and contextually coherent for accurate and complete information retrieval.

To align retrieval with user intent, we implemented a two-tier RAG workflow: initial queries retrieve from section-based chunks for targeted relevance, and from the relevant projects identified, a secondary retrieval pulls the full posters to provide holistic context for deeper exploration. This dynamic adjustment of retrieval granularity mirrors natural clinical reasoning patterns—moving from focused evidence to broader case understanding—and enhances both semantic alignment and user trust. Although it introduces additional system complexity, this architecture significantly improves retrieval resilience, precision, and the overall quality of interaction in high-stakes, healthcare-focused RAG environments.

The process begins with the user posing a question, where the initial relevance_check node determines if the user's query aligns with the established scope and relevance to a medical theme and focus on quality improvement initiatives. This ensures that relevant information can be retrieved within the system, else the user will be prompted to clarify their question.

Upon confirming query relevance, the agent proceeds to the retrieve_poster node, where metadata filtering is applied to identify relevant attributes within the vectorized document database. This metadata-driven approach enhances retrieval precision by prioritizing documents that more directly address the user's intent. At this stage, a loose semantic similarity threshold (e.g., 0.5) is applied, with final relevance scoring and filtering delegated to the LLM for deeper semantic evaluation beyond surface-level similarity.

After the retrieval, all relevant chunks are collectively evaluated by a secondary LLM to assess their combined relevance to the user's query. This holistic evaluation considers the collective informational strength of the retrieved set. By analyzing how the chunks interact and complement each other, the system ensures that the final response is not only individually relevant but contextually coherent and fully aligned with the user's intent. This collective assessment significantly enhances the fidelity and completeness of the output, especially for complex clinical queries requiring multi-source synthesis.

Subsequently, a structured response is returned to the user, comprising the titles of the top-ranked posters, concise summaries, and direct links to the original documents for easy access. This design promotes traceability, enables independent verification of the summarized content, and reinforces user confidence in the system's transparency and reliability.

Shall the user's initial query not result in relevant content within the existing QIP knowledge base, the agent initiates a rewriting process, up to a maximum of 4 tries.

To enhance the efficiency of poster filtering, users are also provided with active controls to refine their search upfront by the year and hospital. This feature enables quicker access to relevant posters based on predefined contextual parameters with a narrower search window. (Appendix A3)

4.2 Data Search Agent

The Data Search Agent is designed to process user-provided text prompts and return summarized results from SQL queries for the user. The underlying process, detailed in the LangGraph (Appendix A4) outlines the agent's architecture and operational steps for user interaction. An agentic approach was found to provide better results in terms of generated answers and fewer errors, as the agent can analyse user inputs in details and evaluate if it needs to clarify with the user further when there is not sufficient information to perform the query. Also, it can learn from any failed query executions to improve its subsequent SQL query generation.

The flow begins when the user provides input being passed through the sequence of nodes in the agent:

1. The format_question node performs a two-step process in which it extracts key information from the user input question, then evaluates if the key points are relevant to the SQL database:
    a. Firstly, the user input is analysed and contextualised to the SQL database schema, to extract the key answers that the user is looking for in the data.
    b. Secondly, these key points are evaluated against the database schema and sample data from the

tables to determine if the SQL database contains the relevant information to answer the user. This step is crucial to cover the cases whereby the information the user is searching for may be within a column value and cannot be found in the column header.

2. The write_query node attempts to generate a SQL query based on the analysed user input from the previous node and the SQL database schema information. The LLM is prompted with good practices such as SQL syntax, especially multi-table query related ones such as JOIN and UNION which it struggles with more, along with reminder to follow SQLite database specific syntax, to help it generate SQL codes more accurately. The resulting SQL query is subsequently cleaned and pre-processed to ensure compatibility with existing SQL server.
3. The execute_query node attempts to run the generated SQL query against the database. Shall the query executes successfully, the results are passed to the generate_answer node; else if the query fails, the error message is passed back to the previous write_query node for it to retry its SQL query generation with the error message providing additional context. A max limit of 3 retries is set to avoid infinite loops of retries, when it occurs, the error message will be passed forward instead, indicating SQL code generation is not possible.
4. Finally, generate_answer node compiles the user's original question, the SQL query, and the returned results to generate a natural language answer. Additionally, the prompt is also used to guide the generated answer to provide more explainability to the generated answer by including information on the data sources and thought process of the LLM.
5. Meanwhile, if the agent determines that the user's question could not be answered with the database at the format_question node, the flow is routed to the clarify_with_user node where the user is prompted for clarification.
6. The final answer at the end of the loop is formatted for the user in the format_response node before being presented.

4.3 Dual Agent Orchestration

Within the main Streamlit UI, 2 app modes are available for selection that calls the agents for poster retrieval and data generation. The user can seamlessly transition between the modes vis an intuitive toggle mechanism, optimizing their workflow to suit their specific research tasks.

One key feature of our solution is the seamless integration between poster-based retrieval and structured SQL data exploration. After answering a user's query by retrieving relevant medical QIP posters, the system intelligently suggests three follow-up data questions tailored to the user's original intent (Figure 4.3.1). These database insights guide clinicians to dive deeper into structured databases, allowing them to validate findings, compare interventions, or quantify outcomes. This tight coupling between unstructured poster insights and structured data analytics accelerates clinical decision-making, reduces manual effort, and promotes a more rigorous, data-driven approach to quality improvement research.

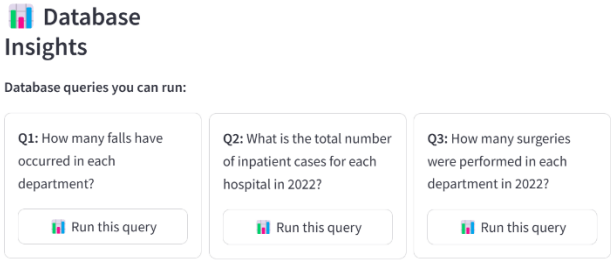


Figure 4.3.1. Suggested questions generated by the LLM for the user as to follow-up after the Poster Search Agent.

5. Model Evaluation and Discussion

The Qwen large language model has been chosen as the foundational model due to its consistent top performance among open-source models and its built-in support for tool calling functionality. To determine the optimal model configuration, a comparative analysis was conducted across several Qwen variants, specifically evaluating the impact of different parameter sizes from the 7B, 14B, and 32B parameter models.

Qwen2.5's selection as our foundational model is further validated by its top-tier performance on the Hugging Face Open LLM Leaderboard. Specifically, Qwen2.5-7B, 14B, and 32B variants consistently rank as the best-performing open-source models within their respective parameter classes, outperforming peers across key benchmarks such as MMLU, BBH, and GSM8K. This external validation underscores Qwen2.5's robustness and suitability for our healthcare quality improvement tasks, ensuring that our comparative analysis is grounded in models that are not only theoretically sound but also empirically superior.

By deploying a uniform underlying Qwen model throughout all iteration controls, we ensure that the experiment setup guarantees any observed performance differences are attributed to the model size and not confounded by the inherent differences across the base language model itself. This approach enhances the validity of our comparative analysis, enabling us to draw more accurate and reliable conclusions regarding the effectiveness of the development techniques in conjunction with the models deployed of varying parameter scales.

5.1 LangSmith Evaluation

LangSmith is a unified observability & evaluation tool for LLMs. Our group selected LangSmith for this project as it is considered an industry-standard tool for LLM monitoring and provides integration with LangChain. Using LangSmith, we were able to trace the performance of different LLMs used throughout the development process and evaluate the trade-offs between latency and token usage. In general, higher token consumption correlates with increased model latency. While using larger models with extended context windows can enhance the quality of responses, it also typically results in longer runtimes.

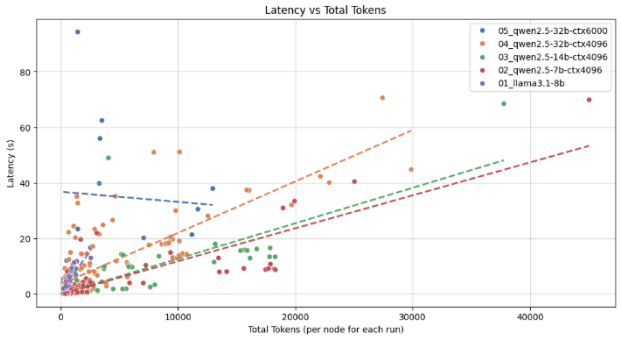


Figure 5.1.1. Figure of latency vs. total tokens used.

An exception to this trend was observed with the `qwen2.5-32b-ctx6000` model, which specifically process cases requiring longer input contexts. As a result, its total token usage typically remained within the 15,000-token range. This more specialized use case naturally limited the variation in output length compared to more general-purpose models, which may have to handle a wider range of inputs and therefore sometimes generate significantly longer responses.

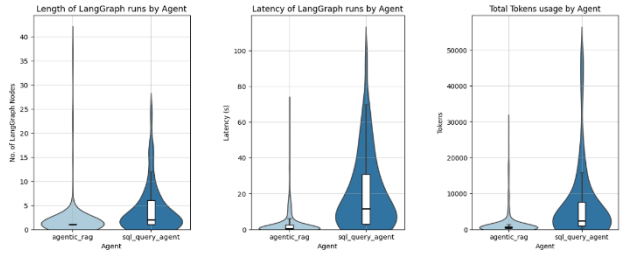


Figure 5.1.2. Figure of Length of LangGraph runs by Agent.

Although most agent runs completed within 20 seconds and consumed fewer than 10,000 tokens, violin plots of graph length, latency, and token usage revealed the presence of outliers. These outlier runs consumed substantially more resources than intended, resulting in longer runtimes and increased computational costs.

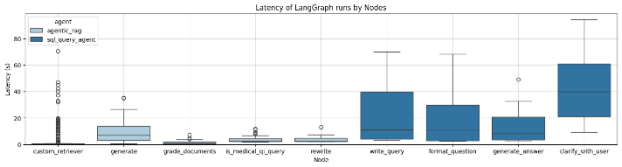


Figure 5.1.3. Latency of LangGraph runs by nodes for Poster Search Agent.

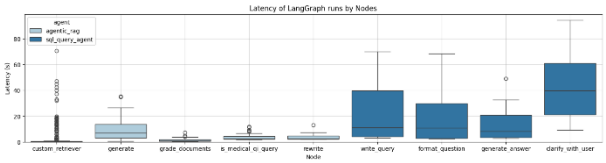


Figure 5.1.4. Latency of LangGraph runs by nodes for Data Search Agent.

A detailed analysis at the node level indicated that the majority of these issues were caused by:

- `custom_retriever` node (RAG agent): This node retrieves relevant poster documents from the database. The number of documents retrieved can vary depending on input relevance. In certain cases, highly relevant inputs triggered the retrieval of a large number of documents, significantly increasing token consumption.
- `format_question` node (SQL query agent): This node extracts key questions from user input and checks for relevance against the database schema. Since the schema itself can consume over 3,000 tokens, additional tokens are also required for the LLM to reason through and formulate the final question, resulting in higher token usage.
- `write_query` node (SQL query agent): This node generates SQL queries using the database schema as part of its prompt. Furthermore, it incorporates any error messages from previous query execution attempts into its input, aiming to refine subsequent queries. However, this error-handling mechanism also increases the overall token consumption.

By identifying these bottlenecks, we are better able to understand resource usage patterns and can explore more safeguards in future iterations to control costs and improve performance.

5.2 Evaluation of Agentic Poster RAG

The Poster RAG system adopts an agentic retrieval-augmented generation (RAG) architecture to address the complexity inherent in medical quality improvement (QIP) queries. Traditional linear RAG systems follow a reactive, single-shot retrieval process based solely on the user's query, which is effective for straightforward information needs but limited in handling multi-faceted, underspecified, or evolving queries typical in QIP contexts. In contrast, the agentic RAG design enables dynamic decision-making: the system can reason about query

relevance before retrieval, decompose complex questions into sub-tasks, iteratively refine search queries, and validate the quality of retrieved documents (Fluid.ai, 2024). Leveraging the LangGraph framework, the Poster RAG agent performs step-by-step planning, proactively screens for query relevance, and grades retrieved contexts for alignment, ensuring a higher baseline quality of information feeding into the final answer.

Rigorous evaluation of system performance was conducted using the RAGAS framework, assessing five key dimensions: context precision, context recall, faithfulness, response relevancy, and response similarity.

To rigorously evaluate our Poster Search Agent under the RAGas framework, we curated a test set comprising 28 queries spanning a gradient of difficulty levels. Queries were stratified into three tiers — Level 1 (easiest, targeted single-hop), Level 2 (moderate), and Level 3 (difficult, broad/multi-hop) — based on anticipated retrieval and reasoning complexity (Appendix A5). In addition to standard queries, we included four edge-case queries specifically designed to test the system's resilience against ambiguous, multi-hop, or low-context prompts. This deliberate design ensured a comprehensive evaluation across common, challenging, and adversarial user scenarios, offering a more realistic assessment of system robustness and generalization performance.

Table 5.2.1. RAGAS evaluation metrics across queries of different levels of difficulties.

Query difficulty	Context _precision	Context _recall	Faithful -ness	Response _relevancy	Res _similarity	Avg _score
1 (easy)	0.733	0.850	0.661	0.770	0.781	0.759
2	0.690	0.846	0.577	0.752	0.666	0.706
3 (difficult)	0.717	0.831	0.513	0.601	0.524	0.637
Average	**0.713**	**0.842**	**0.584**	**0.708**	**0.657**	**0.701**

The table summarizes RAGAS evaluation metrics across queries of varying difficulties. As expected, Level 1 queries achieved the highest overall scores across all metrics, with strong context precision (0.733) and response relevancy (0.770). Performance slightly declined for Level 2 queries, reflecting greater semantic complexity. Level 3 queries, representing the most difficult prompts, showed marked drops in faithfulness (0.513) and response similarity (0.524), highlighting challenges in grounding answers under limited or fragmented evidence conditions. Nevertheless, the system maintained a respectable average score of 0.701 across all queries, demonstrating robust retrieval, strong contextual reasoning, and resilience even under adversarial conditions.

Table 5.2.2. RAGAS evaluation metrics comparison between Qwen2.5 7B vs 14B vs 32B.

Qwen2.5B _paramsize	Context _precision	Context _recall	Faithful -ness	Response _relevancy	Response _similarity	Average_ score
32B	0.706	0.844	0.580	0.713	0.658	0.700
14B	0.666	0.776	0.603	0.645	0.557	0.649
7B	0.615	0.698	0.485	0.619	0.610	0.605

Results demonstrate strong retrieval capabilities, particularly at larger model scales. Using Qwen2.5-32B, the system achieved a context precision of 0.705 and a context recall of 0.844, confirming its ability to recover most of the critical information from past QIP projects. In the medical QIP domain, where comprehensive situational awareness across interventions, methodologies, and outcomes is vital for project design, high recall is especially critical to avoid redundant efforts and missed learning opportunities.

Model scaling effects were clearly observed: Qwen2.5-32B consistently outperformed 14B and 7B variants across all RAGAS metrics, highlighting the advantages of greater model capacity for semantically rich domains like QIP. However, faithfulness — the strict grounding of generated responses in retrieved evidence — remains a notable challenge, with the 32B model achieving a score of 0.524.

The suboptimal faithfulness score highlights the 32B model's limitations in consistently grounding complex clinical answers to retrieved evidence. However, the significant performance gains observed between the 7B and 32B models suggest that scaling to larger models could further enhance grounding reliability.

Recognizing the clinical importance of answer fidelity, our solution mitigates this risk by providing direct links to the original project posters, allowing users to independently verify summarized information and reinforcing transparency.

Addressing this challenge fully beyond raw model size will require architectural innovations such as dynamic context pruning, retrieval-aware generation scaffolding, and answer faithfulness verification mechanisms.

Overall, the agentic Poster RAG system demonstrates a robust foundation for supporting medical QIP research workflows. High retrieval recall enables comprehensive evidence synthesis, while thoughtful agentic design mitigates many common failure modes of traditional RAG. Strategic future enhancements aimed at tightening answer grounding and expanding multimodal extraction capabilities will further optimize the system's fidelity, completeness, and clinical utility.

5.3 Evaluation of Data Search Agent

To evaluate the effectiveness of the Text-to-SQL agent, we adopt a robust evaluation framework grounded in the SPIDER benchmark methodology. A set of 10 diverse natural language questions was manually curated by our team, with corresponding gold-standard SQL queries constructed and validated using SQLite to ensure the correctness of ground truth answers.

Each question was then automatically categorized into one of four difficulty levels—easy, medium, hard, or extra hard—based on the SPIDER-defined complexity of the

gold SQL components. This stratified setup ensures a comprehensive assessment of the model's ability to generate queries across a range of complexities.

The predicted SQL queries are evaluated using two primary metrics: exact match and component-level (partial) match.

The exact match requires the predicted query to be structurally identical to the gold-standard, resolving issues such as clause ordering. In contrast, Partial match decomposes the SQL into clause-level components (e.g., SELECT, WHERE, GROUP BY, JOIN), treated as bags of several sub-components, and measures overlap between the predicted and gold components. This allows recognition of partial correctness even when full exactness is not achieved.

Given the complexity of real-world SQL constructs, including nested queries, aliasing, and aggregation functions (e.g., ROUND, CAST, CASE WHEN, IS NULL), we observed several parsing failures using SPIDER's original evaluation script. To address these issues, we introduced targeted modifications to the SPIDER process_sql.py parser to improve resilience against:

1. Invalid alias detection during parsing 2. Functions not originally supported (e.g., ROUND, CAST, TIMESTAMPDIFF) 3. Ambiguous JOIN patterns and long subqueries 4. Improper tokenization of NULL-related conditions

These enhancements enabled more accurate parsing of complex SQL expressions, which are common in user-generated prompts for enterprise use cases.

While these parser adjustments improve practical coverage, they may slightly deviate from the strict original evaluation semantics of SPIDER. However, they also mitigate known limitations, such as the penalization of logically equivalent queries that differ only in alias naming or clause ordering—issues previously raised in recent work (Richard et al., 2024).

In conclusion, our modified evaluation pipeline, informed by SPIDER's methodology and refined for real-world SQL structures, provides a more faithful reflection of the model's strengths and weaknesses in text-to-SQL generation.

Table 5.3.1. Text to SQL evaluation metrics comparison between Qwen2.5 7B vs 14B vs 32B.

Question/ Difficulty	Evaluation	Qwen-7B	Qwen-14B	Qwen-32B
1-Easy	Exact Match	True	True	True
	Execution Value	True	True	True
2-Easy	Exact Match	True	True	True
	Execution Value	True	True	True
3-Medium	Exact Match	True	False	True
	Execution Value	True	True	True
4-Easy	Exact Match	False	False	True
	Execution Value	False	True	True
5-Extra	Exact Match	False	False	False
	Execution Value	True	True	True
6-Hard	Exact Match	False	True	False
	Execution Value	False	True	False
7-Extra	Exact Match	False	False	False
	Execution Value	False	False	False
8-Extra	Exact Match	False	False	False
	Execution Value	False	False	False
9-Extra	Exact Match	False	False	False
	Execution Value	False	False	False
10-Extra	Exact Match	False	False	False
	Execution Value	False	False	False

Table 5.3.2. Text to SQL evaluation metrics comparison between Qwen2.5 7B vs 14B vs 32B.

	easy	medium	hard	extra	all
count	3	1	1	5	10
========== EXACT MATCHING ACCURACY ==========					
exact match	1.000	1.000	0.000	0.000	0.400
----------PARTIAL MATCHING ACCURACY----------					
select	1.000	1.000	0.000	0.200	0.500
select(no AGG)	1.000	1.000	0.000	0.200	0.500
where	1.000	0.000	0.000	0.500	0.500
where(no OP)	1.000	0.000	0.000	0.500	0.500
group(no Having)	0.000	1.000	0.000	0.667	0.750
group	0.000	1.000	0.000	0.667	0.750
order	0.000	0.000	0.000	0.000	0.000
and/or	1.000	1.000	1.000	1.000	1.000
IUEN	0.000	0.000	0.000	0.000	0.000
keywords	1.000	1.000	1.000	0.750	0.857

As part of the evaluation, we compared model performance using Qwen-7B, Qwen-14B, and Qwen-32B on a custom SPIDER-style SQL generation benchmark. Across both exact match and execution accuracy (exec match) metrics, Qwen-32B consistently outperformed Qwen-7B and Qwen-14B. While Qwen-14B achieved execution scores comparable to Qwen-32B, we observed that it occasionally produced SQL queries with complex expressions or syntax (e.g., nested functions, aliases) that the SPIDER evaluation parser could not process correctly, leading to some parsing failures during exact match evaluation. Nevertheless, the underlying execution results remained strong, suggesting that Qwen-14B generated semantically correct but

syntactically non-standard SQL in some cases (Appendix A6).

For exact matching and partial matching scores, the results clearly showed that larger models generally achieved higher SPIDER scores, reinforcing the idea that model capacity correlates with SQL generation quality in structured query tasks.

Overall, the evaluation confirmed that SPIDER-style exact match and partial match scores are effective indicators of a model's ability to generate correct SQL text, even when combined with execution-based evaluation for robustness.

6. Limitation

6.1 Poster Search Agent Limitations

1. While maximizing retrieval recall by including broader relevant chunks enriches answer completeness, it introduces minor faithfulness risks, as larger context windows increase cognitive load on the language model and elevate the potential for semantic drift. In the medical QIP domain, where project fidelity and evidence-based reporting are critical, even minor hallucinations can mislead subsequent improvement efforts. To mitigate this, future work will explore dynamic context pruning strategies, selectively narrowing context inputs based on relevance scoring and query intent, thereby balancing the trade-off between coverage and groundedness.
2. Current text-only extraction pipelines omit a significant portion of critical information embedded within tables, flow diagrams, and outcome charts, which are pervasive in medical QIP posters. This blind spot results in incomplete contextual understanding and retrieval gaps, particularly for metrics-driven projects. Future system iterations will integrate multimodal extraction pipelines, combining OCR, structured table parsing, and diagram captioning, enabling comprehensive semantic indexing of both textual and visual elements to significantly enhance retrieval quality and answer fidelity.
3. As the volume and semantic richness of QIP posters expand, the risk of embedding collapse in low-dimensional vector spaces intensifies, leading to semantic blurring and degraded retrieval precision. Without adequate retrieval sophistication, fine-grained project nuances critical to QIP outcomes may be lost. To address this, future development will implement higher-dimensional embeddings, multi-vector retrieval architectures such as CoLBERT, and hierarchical retrieval frameworks that preserve semantic separability while maintaining scalable, efficient retrieval over expanding corpus.

6.2 SQL Query Agent Limitations

1. Scalability Challenges with Increasing Database Size – the SQL agent relies heavily on the database schema to provide essential context when generating queries. As the size of the SQL database increases, the schema naturally becomes more complex, with complicated relationships between tables. Currently, Qwen2.5-32B is used to handle a SQL database consisting of only five tables, and even so, it requires at least 3,000 tokens — a relatively large context size for a non-commercial solution. As database complexity scales up, more powerful LLM will be necessary to process the expanded context effectively and accurately interpret the increasingly complicated relationships and structures.

2. Limitations in Handling Execution Errors and Retry Mechanisms – another limitation lies in how the agent manages execution errors resulting from incorrect SQL code generation. To prevent infinite, retry loops (where the agent repeatedly attempts to regenerate SQL queries without resolving the underlying error) the current solution imposes a maximum limit of three retries before passing the error to the user. While this safeguard is necessary with the current capabilities, a more advanced LLM could generate correct SQL queries on the first attempt, significantly reducing the need for retry limits and error-handling mechanisms.

7. Conclusion

This project advances healthcare quality improvement by demonstrating how LLM-powered, agentic systems can close critical data-literacy gaps among clinicians. Through the orchestration of Poster RAG and Text-to-SQL agents with LangGraph, we built a modular, resilient workflow that enables context-grounded evidence synthesis and structured data exploration. Rigorous evaluation confirms strong retrieval, reasoning, and robustness, while highlighting pathways for future innovation in multimodal retrieval, faithfulness optimization, and scalable deployment. Our work establishes a strong foundation for agentic AI as a catalyst for more rigorous, efficient, and data-driven clinical improvement efforts.

References

AWS. (2025, March 6). Build a Multi-Agent System with LangGraph and Mistral on AWS.

Celi LA et al. Bridging the Health Data Divide. J Med Internet Res. 2016;18(12):e325. doi: 10.2196/jmir.6400.

Chase, H. (2023, December 14). LangGraph: Multi-agent workflows for LLMs. LangChain Blog. https://blog.langchain.dev/langgraph/

Dev.to. (2025, February 10). LangGraph Uncovered: Building Stateful Multi-Agent Applications with LLMs.

Domo. (2025, February 26). Agentic AI Explained: Definition, Benefits, and Use Cases.

Fluid.ai. (2024, December 18). *Agentic RAG vs. Traditional RAG: The Future of AI Decision-Making*. Retrieved from https://www.fluid.ai/blog/agentic-rag-vs-traditional-rag-the-future-of-ai-decision-making

Gan, Y., Chen, X., Xie, J., Purver, M., Woodward, J. R., Drake, J., & Zhang, Q. (2021). Towards robustness of text-to-SQL models against synonym substitution. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (pp. 2505-2515). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.195

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997. https://doi.org/10.48550/arXiv.2312.10997

Han FK (2024, February 14) *Commentary: What to do about rising medical costs in Singapore.* Channel News Asia.https://www.channelnewsasia.com/commentary/singapore-healthcare-medical-costs-rising-expensive-solution-insurance-coverage-4119801

IBM. (2025, February 14). What is LangGraph?

IBM. (2025, February 24). What Is Agentic AI?

Ivers NM et al. Growing literature, stagnant science? J Gen Intern Med. 2014;29(11):1534-41. doi: 10.1007/s11606-014-2913-y.

LangChain. (2024). *LangGraph: Building stateful multi-agent workflows*. LangChain. Retrieved April 27, 2025, from https://langchain-ai.github.io/langgraph/

Larson DB et al. Recognizing and Avoiding Common Mistakes in Quality Improvement. J Am Coll Radiol. 2021;18(3 Pt B):511-513. doi: 10.1016/j.jacr.2020.09.053.

Le Scao, T., Tunstall, L., Poesia, G., Silva, A. D., Rozière, B., Chowdhery, A., Geiger, A., Lavril, T., Zhou, C., Lebret, R., Vetterli, M., Wolf, T., Bhagia, A., Santilli, A., von Werra, L., & Luccioni, A. S. (2023). Mistral 7B. arXiv preprint arXiv:2310.06825. https://arxiv.org/abs/2310.06825

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459-9474. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Liu, Q., Guo, B., Chen, Z., Zhu, J., & Tang, J. (2023). A survey of text-to-SQL parsing. arXiv preprint arXiv:2208.13629. https://arxiv.org/abs/2208.13629

Meta. (2024, April 18). Introducing Llama 3: The most capable openly available AI model. https://ai.meta.com/blog/meta-llama-3/

Milvus. (n.d.). Stop using outdated RAG: DeepSearcher & Agentic RAG approaches—changes everything. Milvus. https://milvus.io/blog/stop-use-outdated-rag-deepsearcher-agentic-rag-approaches-changes-everything.md

Richard R, Gowtham K, Ashutosh T. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. 2424Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2000.

Singapore Health Services. (n.d.). Poster exhibition. Singapore Healthcare Management Congress. Retrieved April 27, 2025, from https://www.singaporehealthcaremanagement.sg/Pages/Poster-Exhibition-2022.aspx

Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., Payne, P., Seneviratne, M., Gamble, P., Kelly, C., Scharli, N., Ridgeway, K., Chen, I., Uminski, M., Pfister, H., ... Nori, H. (2023). Large language models encode clinical knowledge. Nature, 620(7972), 172-180. https://doi.org/10.1038/s41586-023-06291-2

Tao Y et al. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. 2018.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., & Tan, T. (2023). Large language models in medicine. Nature Medicine, 29(8), 1930-1940. https://doi.org/10.1038/s41591-023-02448-8

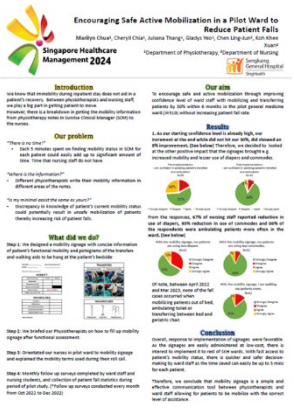
Weights & Biases. (2025, March 10). Agentic RAG: Enhancing retrieval-augmented generation with AI agents.

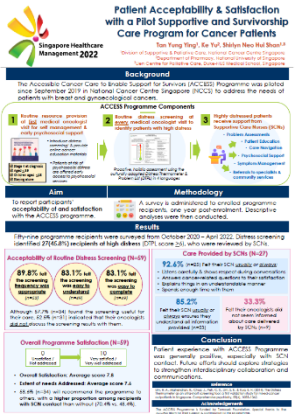
Appendix

A1. Source code on GitHub

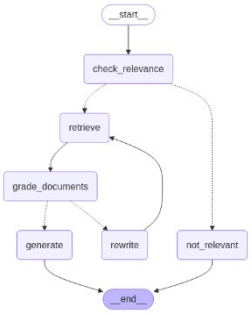
https://github.com/BZ269/5153-2025-final-proj-gp2

A2. Sample QIP Posters

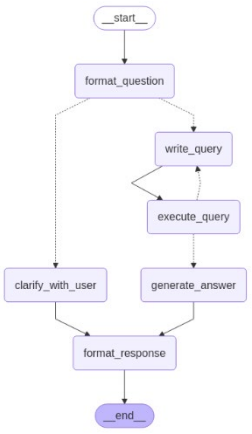




A3. LangGraph for Poster RAG Agent



A4. LangGraph for Text-to-SQL Agent



A5. Sample Test Queries for RAGas (Poster RAG)

[{ "query": "What specific PDSA interventions were tested to reduce prescription error rates at Seng Kang Polyclinic?",

"ground_truth": "SHM_RM054_SHP tested three PDSA cycles: (1) standardizing medication charts (ineffective), (2) bi-weekly prescription error data and interviews with high-error prescribers, and (3) pharmacy technicians flagging fragmented prescriptions to reduce omissions."},

{"query": "Show me the projects on falls from 2022 by SKH.",

"ground_truth": "The project SHM_RM001_SKH, titled 'Reducing Inpatient Falls Related to Toileting Needs', conducted at SKH, describes work carried out during 2022. It focused on targeted toileting interventions, visual reminders, and nursing staff training to reduce falls specifically related to toileting needs."},

{"query": "Are there QIPs that used environmental or physical layout interventions (e.g. ward design, signage, chair placement) to reduce patient risks?",

"ground_truth": "Projects: SHM_CO005_SKH, SHM_RM022_CGH, SHM_RM055_SHP, SHM_RM053_SCH, SHM_RM044_SCH — used signage, bin placement, ward redesigns, and visual reminders."}]

A6. Sample Test Queries for SQL Evaluation (Data Search)

[{“query”: “What is the average billing amount from each hospital's outpatient cases?”,

“ground_truth”:

“SELECT hospital_id, AVG(bill_amount)

FROM outpatient_cases

GROUP BY hospital_id;”},

{“query”: “How many inpatient cases that required surgery also had follow-ups?”,

“ground_truth”:

“SELECT COUNT(*)

FROM inpatient_cases

INNER JOIN surgery_metrics ON inpatient_cases.patient_id = surgery_metrics.patient_id

WHERE inpatient_cases.follow_up_date IS NOT NULL;”},

{“query”: “Which departments have the longest average surgery durations and also the highest average length of stay? Are these concentrated in specific hospitals?”,

“ground_truth”:

“SELECT sm.hospital, sm.department, AVG(sm.duration) AS avg_surgery_duration, AVG(sm.length_of_stay) AS avg_length_of_stay, COUNT(sm.surgery_id) AS surgery_count

```
FROM surgery_metrics sm
GROUP BY sm.hospital, sm.department
ORDER BY avg_surgery_duration_minutes DESC,
avg_length_of_stay_days DESC;"}]
```