IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Natural Language Sql Querying Via Langchain And Ai Agent

[1]Prof. Vijay Shanker, [2]Mohd Anas Khan, [3]Iliyaan Karovalia, [4]Hamza Ali Shaikh, [5]Aadya Jha,

Department of Artificial Intelligence and Data Science,
Rizvi College of Engineering, Mumbai, India

Abstract: In today's data-centric landscape, effective database interaction is vital for informed decision-making. However, traditional querying methods require technical expertise in Structured Query Language (SQL), creating a barrier for non-technical users. This project proposes a user-friendly solution that enables natural language interaction with SQL databases, thereby democratizing access to structured data.

We present a Streamlit-based web application that integrates LangChain agents with a large language model (LLM). This architecture allows users to input conversational prompts, which are interpreted and translated into executable SQL queries. The system removes the need for SQL proficiency, offering a simplified, scalable, and intuitive interface for data retrieval. Amid rapid advancements in AI and natural language processing (NLP), this solution exemplifies practical application by transforming complex database interactions into accessible tasks. It serves both technical and non-technical users, such as data analysts, decision-makers, and business professionals, enabling them to derive insights independently and efficiently.

By reducing reliance on IT specialists and facilitating rapid data access, the proposed system contributes to more agile and data-driven business processes. Its design prioritizes usability, flexibility, and real-world applicability, addressing a growing need for intelligent, inclusive database tools in modern digital environments.

Keywords - Natural Language to SQL, Conversational Database Interface, AI-powered Query Generation, SQL Automation, Intelligent Database Interaction, Natural Language Query Processing.

I. INTRODUCTION

In the era of data-driven decision-making, the ability to efficiently interact with databases is a fundamental requirement across diverse organizational roles. However, traditional database querying necessitates a strong command of Structured Query Language (SQL), which poses a barrier to non-technical users seeking to extract actionable insights from data. This project addresses this challenge by developing a web-based application that enables natural language interaction with SQL databases, thereby facilitating seamless access to structured data regardless of the user's technical background.

The proposed system is implemented using the Streamlit framework and integrates LangChain's agent architecture with large language model (LLM). This combination allows for the interpretation of user queries expressed in natural language and their real-time conversion into syntactically valid SQL commands. As a result, users can perform complex data retrieval operations without explicit knowledge of database query syntax.

The application supports both **SQLite** and **MySQL** databases, offering flexibility and adaptability for various deployment contexts. Through a user-friendly interface, individuals can connect to a database of choice, authenticate securely, and submit queries in a conversational format. The system dynamically processes these inputs and retrieves relevant results, effectively mimicking a natural dialogue between the user and the database.

This work aims to bridge the gap between sophisticated database systems and the demand for intuitive, human-centered data access tools. By eliminating the dependency on SQL proficiency, the proposed solution democratizes data interaction, streamlines analytical workflows, and enhances productivity in modern business environments.

II. LITERATURE SURVERY

2.1 Survey of Existing Systems:

Numerous tools and platforms have emerged to simplify interaction with SQL databases. However, most still require users to possess a foundational understanding of SQL or database structures. Existing solutions fall into several categories:

- **SQL Query Builders**: Tools such as SQLizer and SQL Fiddle enable users to generate SQL queries via graphical interfaces. While these reduce the need to write code manually, users must still comprehend database schema and relational logic. They also lack natural language support, limiting accessibility.
- **Business Intelligence Tools**: Platforms like Tableau and Power BI offer powerful data visualization and query generation through drag-and-drop interfaces. Although they abstract some complexity, they often demand users to grasp the underlying data model and SQL-like operations, posing challenges for non-technical users.
- **Natural Language Query Interfaces**: Tools such as AskData and Qlik enable users to pose natural language questions that are translated into SQL queries. However, their applicability is typically confined to predefined query types and domains, limiting flexibility and robustness.
- **AI-Powered Chatbots**: Some organizations deploy AI chatbots for data-related inquiries. These systems utilize machine learning to interpret queries, but often lack the precision required to generate accurate and complex SQL statements consistently. Their capabilities are generally restricted to simple interactions.
- **Natural Language Processing Libraries**: Frameworks like Rasa and Dialogflow allow developers to build custom NLP-powered interfaces. While powerful, these require extensive development efforts and technical knowledge, rendering them impractical for widespread adoption by general users.

2.2 Research Gap:

Despite progress in query abstraction and AI-based solutions, a significant gap persists in systems that can **reliably translate natural language into SQL across various query types**, with minimal user training or setup. Most current solutions either compromise on flexibility, require customization, or lack sufficient accuracy. This gap underscores the need for a platform that provides **real-time, natural language-based access to databases** in a way that is both **intuitive and technically robust**.

2.3 Problem Definition and Objectives

2.3.1 Problem Statement:

Accessing and analyzing relational data remains a technical hurdle for non-specialists due to the necessity of mastering SQL. This limits data accessibility for many stakeholders who could otherwise contribute to data-driven decision-making. The core challenge lies in developing a solution that allows users to interact with databases using **natural language**, while maintaining query accuracy and system performance.

2.3.2 Objectives:

The primary objective of this project is to design and implement a **Streamlit-based web application** that enables users to interact with SQL databases through **natural language queries**. Key goals include:

1. Creating an intuitive interface that abstracts SQL complexity.
2. Integrating advanced NLP capabilities to translate natural language into SQL.
3. Supporting efficient and accurate data retrieval and insights generation.

4. Enhancing database usability for non-technical users across various domains.

2.4 Scope of the Project

The scope of this project encompasses the following components:

- ***User Interface Development**:* Design an accessible, conversational UI using Streamlit, suitable for users with minimal technical expertise.
- ***Natural Language Processing**:* Utilize Llama Model model through LangChain to interpret user inputs and convert them into structured SQL queries.
- ***Database Interaction**:* Support dynamic querying for multiple databases (e.g., MySQL, SQLite), ensuring secure connections and efficient query execution.
- ***Insights Generation**:* Provide meaningful outputs such as summaries, visualizations, or metrics from retrieved data, along with support for iterative querying.
- ***Testing and Evaluation**:* Conduct accuracy assessments of the NLP-to-SQL translation and gather user feedback to refine the application.
- ***Documentation and Support**:* Deliver a comprehensive user guide and troubleshooting resources to assist users in operating the system effectively.

III. PROPOSED SYSTEM

3.1 System Overview

The proposed system is a **Streamlit-based web application** that enables users to query SQL databases using **natural language**. It is built upon the **LangChain framework**, leveraging **Meta Llama 3 language model** Using **Groq's Api** for translating user inputs into executable SQL queries. This approach allows users with little or no knowledge of SQL to interact seamlessly with databases, democratizing data access and enabling efficient decision-making.

3.2 System Architecture

Key Components:

- **LangChain**: Serves as the core framework for handling natural language processing. It manages query parsing, model interaction, and agent coordination.
- **Groq API (LLaMA3-8B-8192)**: Powers the language model backend, enabling accurate translation of natural language into SQL queries.
- **Streamlit**: Provides an interactive web UI for user input, database selection, and result display.
- **SQLAlchemy**: Abstracts database operations, handling secure, dynamic connections to both **SQLite** and **MySQL** databases.

Operational Flow:

1. **Database Selection & Authentication**: Users choose a database (SQLite/MySQL) and provide credentials for remote access (if needed).
2. **API Key Validation**: The system verifies the Groq API key to enable secure model access.
3. **Query Processing**:
    - Natural language input is collected via the Streamlit interface.
    - LangChain processes the input and uses Groq's LLM to generate a valid SQL query.
4. **Execution & Response**:
    - The query is executed using SQLAlchemy.
    - Results are retrieved, formatted, and displayed to the user in a readable format.

Current Limitation:

- The system is currently **read-only**. It supports **SELECT** queries but intentionally **excludes data manipulation operations** (INSERT, UPDATE, DELETE) to prevent unintentional data changes.

3.3 Hardware and Software Requirements

3.3.1 Hardware Requirements:

Below are the Hardware Requirements for the Model to run Locally

Table 1 Hardware Requirements

Component	Specification
Processor	Multi-core CPU (Intel i5 / AMD Ryzen 5) or higher
Ram	Minimum 8 GB
Storage	256 GB SSD (preferred) or HDD
Graphics	Integrated GPU sufficient (no dedicated GPU needed)
Operating System	Windows 11 (development), compatible with macOS and Linux

3.3.2 Software Requirements:

For the software stack, various tools, libraries, and frameworks were utilized to ensure smooth development and functioning of the system. But if the Model is deployed and the end user is using it he/she does not have to install any software. The software is up on the web which can be easily accessible via internet

Below is the Software requirement for the Model to run Locally:

Table 2 Software Requirements

Category	Description
IDE	Visual Studio Code – lightweight and extensible Python development
Language	Python – primary language for logic, APIs, and database interaction
Web Framework	Streamlit – for building and deploying interactive web UIs
NLP Framework	LangChain – for natural language to SQL processing
Database Libraries	SQLAlchemy and SQLite3 – ORM and local database interface
LLM Integration	Groq API (LLaMA3-8B-8192) – for natural language interpretation
Databases Supported	SQLite (lightweight, local) and MySQL (scalable, remote-capable)
Package Manager	pip – for managing and installing Python dependencies
Version Control	GitHub – for collaborative development and version tracking

IV. SYSTEM DESIGN AND IMPLEMENTATION DETAILS

4.1 Overview

The proposed system is designed to enable users to interact with relational databases using natural language, eliminating the need for direct SQL knowledge. The architecture emphasizes modularity, scalability, and ease of use, allowing seamless integration of various databases and language models. Key components include a Streamlit-based UI, a Groq-powered LLM, a database connector layer, and an agent framework based on LangChain.

4.2 Architectural Components

4.2.1 User Interface

The front end is developed using Streamlit, offering an intuitive chat-based interface. Users can submit queries in natural language, select between supported databases (SQLite or MySQL), and view results directly. Sensitive inputs like API keys and credentials are handled securely through the sidebar.

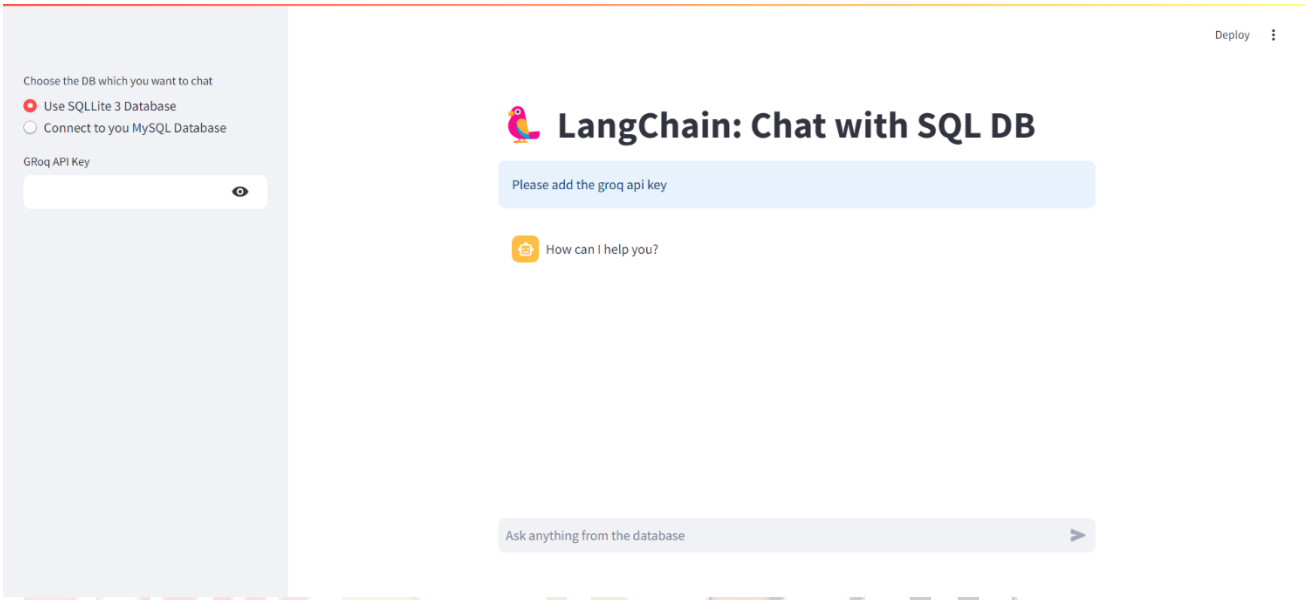

Fig. 1: Shows Streamlit Web Interface UI

4.2.2 Language Model Integration

At the core of the system is Groq's Llama3-8b-8192 language model. It translates user input into syntactically valid and semantically relevant SQL queries, based on contextual understanding and schema-specific awareness. This eliminates the need for users to learn SQL.

4.2.3 Database Connector

Database interaction is abstracted using SQLAlchemy, which supports both SQLite (for local data) and MySQL (for external connections). Users can dynamically switch between databases through the UI. SQLAlchemy simplifies ORM operations and enhances portability across database types.

4.2.4 Agent Framework

LangChain's SQL Agent acts as the middleware between the language model and the database. It leverages SQLDatabaseToolkit to manage the flow from user query to SQL execution and result retrieval. This component ensures smooth coordination and execution of the pipeline.

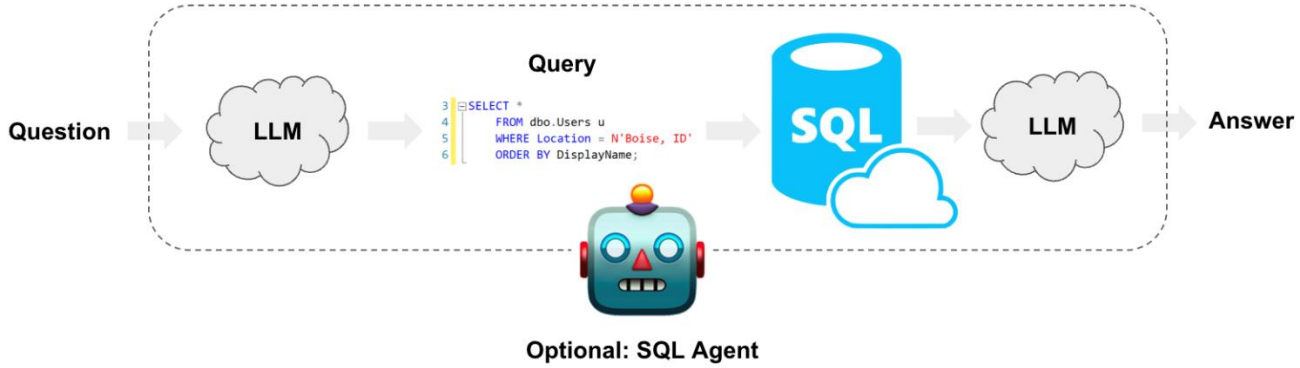

Fig.2: Process of Natural Language to Sql Conversion, [1]

4.3 Functional Workflow

Users interact via the Streamlit interface, selecting a database and entering natural language queries. API keys and MySQL credentials are securely submitted. The LLM interprets the user query and translates it into an SQL command, considering database schema and intent.SQLAlchemy establishes a connection with the selected database. The SQL query is executed, and relevant results are fetched.Retrieved results are processed and displayed in a readable format on the web interface. LangChain's agent ensures consistency and error-handling throughout the flow.

4.4 Design Considerations

Each component (UI, LLM, agent, DB connector) operates independently, allowing easier debugging, maintenance, and future upgrades. Components can be swapped or upgraded (e.g., replacing LLMs or adding PostgreSQL support) without impacting the overall architecture. The system is designed to integrate additional databases and model endpoints as needed with minimal structural changes. Credentials are not exposed in plain text. The system currently supports read-only database operations, mitigating risks associated with write access.

4.5 Implementation Methodology

The system was developed in iterative phases:

4.5.1 Requirement Analysis:
Identified the need for a no-code database querying system targeting non-technical users.

4.5.2 Design:
Architected the system using a client-server model with modular boundaries between the frontend, backend, and language model integration.

4.5.3 Implementation:
User input is parsed using tokenization techniques to extract relevant keywords. A Lang Chain based agent interfaces with the LLM and database layers. SQL Alchemy handles query execution and result fetching. Results are rendered clearly via Stream lit, maintaining conversational flow.

V. TESTING AND RESULTS

After the query is executed, the results are returned to the system, processed by the LangChain agent, and then presented back to the user through the Streamlit interface. The user receives a clear, readable output from the database, allowing them to interact with and explore the data without any direct SQL knowledge.

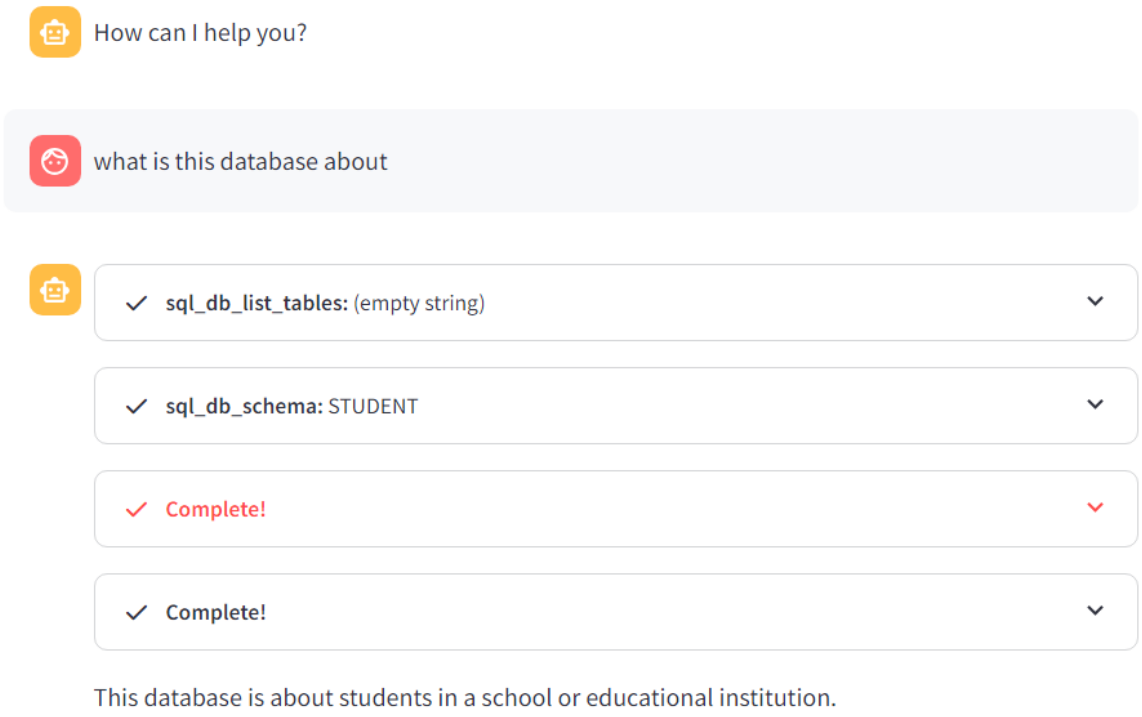

Fig.3. Result in Natural Language

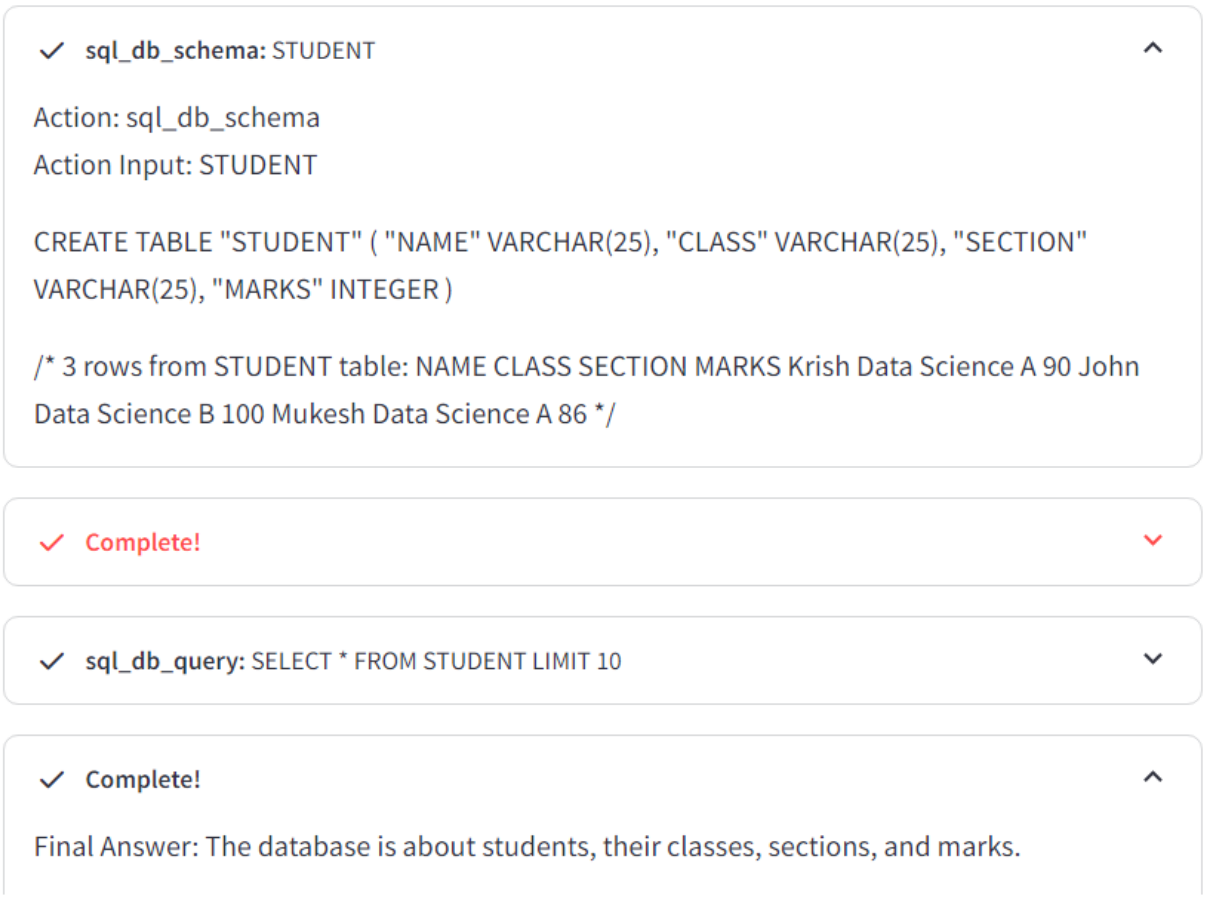

Fig.4. Query Execution and Response Result

VI. CONCLUSION

In conclusion, the project successfully demonstrates the capabilities of a Streamlit-based web application that enables users to interact with SQL databases through natural language queries. By leveraging advanced technologies such as LangChain and Groq's LLM, the system provides a user-friendly interface that abstracts the complexities of SQL, allowing users to retrieve insights from databases without needing extensive knowledge of query syntax. The architecture of the application, with its robust encoder-decoder framework, ensures accurate interpretation of user queries and effective generation of SQL commands. The design prioritizes data retrieval while maintaining security, as the system is engineered solely for reading data, thus preventing unintended modifications to the database.

Ultimately, this project not only showcases the potential of AI-driven solutions in simplifying database interactions but also opens avenues for further research and development in the realm of natural language processing and database management. Through ongoing efforts, we aspire to build a more versatile tool that empowers users with deeper insights and greater accessibility to data.

VII. FUTURE WORKS

While the integration of LLMs into text-to-SQL systems has achieved significant strides in natural language querying, a number of challenges remain. These will be overcome with future research that will further improve the performance, efficiency, and usability of these systems. As seen from the elaboration on some key challenges and their possible solutions in the following, it will shape the evolution of this paradigm

A. Scalability and Computational Efficiency

Enhancing LLM-based text-to-SQL systems for large and complex databases without losing computational efficiency is an important challenge. The processing and generation cost of SQL queries remains high, especially with longer sequences and larger datasets. Future solutions will likely focus on model optimizations, more efficient retrieval and storage mechanisms, and specialized indexing techniques to streamline query generation.

B. Dynamic Adaptation to Schema Changes

Real-world databases are dynamic, constantly evolving with schema changes and added data, necessitating adaptation of LLM-based systems without full retraining. Techniques like incremental learning and flexible architectures will enable seamless updates of both LLMs and KGs, maintaining up-to-date query accuracy, particularly for rapidly changing databases.

C. Contextual Accuracy and Disambiguity

Many LLM-based text-to-SQL systems face challenges in handling complex and ambiguous queries where context is not explicitly given. Improving contextual accuracy will require research into how LLMs use structured information from KGs. Enhancing semantic links between user queries and the database schema will be critical, and more advanced semantic parsing and disambiguation techniques will help resolve ambiguity.

D. Ethics, Data Privacy, and Interpretability

The application of LLMs in critical domains like healthcare, finance, and education raises ethical concerns regarding data privacy and model interpretability. It is essential for such systems to be transparent, reliable, and respectful of user privacy. Future work will need to establish clear explainability protocols, safe data handling practices, and transparent AI procedures to build trust in LLM-based text-to-SQL systems

E. Knowledge Graph Integration and Maintenance

While knowledge graphs improve schema awareness and enhance query precision, their construction and maintenance are complex and resource-intensive. A scalable implementation will require efficient automation of KG creation and optimized integration with LLMs. Additionally,

developing dynamic updating techniques for KGs without degrading performance is crucial for ensuring that systems remain effective as data and schemas evolve.

REFERENCES

[1] LangChain, “LangChain Documentation,” [Online]. Available: https://python.langchain.com/.

[2] Groq, “Groq,” [Online]. Available: https://groq.com/.

[3] OpenAI, “ChatGPT: A Conversational AI Model,” [Online]. Available: https://openai.com/chatgpt.

[4] Streamlit, “Streamlit: The Fastest Way to Build Data Apps,” [Online]. Available: https://streamlit.io/.

[5] A. Auffarth, Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT, and other LLMs, Packt Publishing, 2023.

[6] A. Kotiyal, P. G. J, G. P. M. S, R. M. Devadas, V. Hiremani, and P. Tangade, "Chat With PDF Using LangChain Model," 2024 Second International Conference on Advances in Information Technology (ICAIT), Chikkamagaluru, Karnataka, India, 2024, pp. 1-4, doi: 10.1109/ICAIT61638.2024.10690817.

[7] T. Oguzhan Topsakal and T. Cetin Akinci, "Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast," International Conference on Applied Engineering and Natural Sciences, vol. 1, no. 1, 2023.

[8] M. Khorasani, M. Abdou, Hernández Fernández, and J. Streamlit, "Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework," Apress, 2022.

[9] "Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks and Implications," pp. 4-5.

[10] O. Akinci Topsakal and T. C. Creating, "Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast," In Proceedings of the International Conference on Applied Engineering and Natural Sciences, vol. 1, pp. 1050-1056, 10–12 July 2023.

[11] K. Pandya and M. Holia, "Automating customer service using LangChain: Building custom open-source GPT chatbot for organizations," 2023.

[12] "Chat2VIS: Generating Data Visualizations via Natural Language Using ChatGPT," odex and GPT-3 Large Language Models.

[13] A. Kate, S. Kamble, A. Bodkhe, and M. Joshi, "Conversion of Natural Language Query to SQL Query," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2018, pp. 488-491, doi: 10.1109/ICECA.2018.8474639.

[14] G. V. R. Ram, K. Ashinee, and M. Anand Kumar, "End-to-End Space-Efficient Pipeline for Natural Language Query based Spacecraft Health Data Analytics using Large Language Model (LLM)," 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT), Kottayam, India, 2024, pp. 1-6, doi: 10.1109/ICITIIT61487.2024.10580129.

[15] H. Yang, Z. Yang, R. Zhao, X. Li, and G. Rao, "The implementation solution for automatic visualization of tabular data in relational databases based on large language models," 2024 International Conference on Asian Language Processing (IALP), Hohhot, China, 2024, pp. 175-180, doi: 10.1109/IALP63756.2024.10661162.

[16] V. Câmara, R. Mendonca-Neto, A. Silva, and L. Cordovil, "A Large Language Model approach to SQL-to-Text Generation," 2024 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2024, pp. 1-4, doi: 10.1109/ICCE59016.2024.10444148.

[17] F. Siasar Djahan, M. Norouzifard, S. H. Davarpanah, and M. H. Shenassa, "Using natural language processing in order to create SQL queries," 2008 International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, 2008, pp. 600-604, doi: 10.1109/ICCCE.2008.4580674.

[18] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," arXiv preprint arXiv:1711.04436, 2017.