University of Science & Technology, Bannu

Final Year Project Proposal

Title: Unified SQL Agent: LLM-Based Natural Language Interface for SQL Databases

Students Particulars

Name: Muhammad Ashan

Reg_no:________________

Supervisor

Name: Mr. Salam Ullah Khan

Department of Computer Science
UST Bannu

Signature:_________________

Department of Computer Science UST Bannu

Table of Contents

Abstract

This project aims to develop a web-based application that allows users to interact with relational databases using natural language. Non-technical users often struggle to write complex SQL queries to extract insights from databases. This system bridges that gap by converting user questions into accurate SQL queries using the large language model (LLM) via LangChain. It supports multiple databases, including SQL Server, MySQL, PostgreSQL, and SQLite. The backend dynamically connects to databases, fetches schema information, generates prompts, and validates/executed generated queries. The frontend provides a clean interface for users to input their credentials and questions. The expected outcome is a functional AI agent that increases database accessibility, especially for business users, enabling them to interact with structured data more efficiently.

1. Introduction

Background:
In the field of data analysis and decision-making, structured data stored in relational databases plays a vital role. However, accessing this data requires knowledge of Structured Query Language (SQL), which can be a technical barrier for non-technical users.The integration of artificial intelligence and natural language processing (NLP) into software systems provides an opportunity to simplify data interaction.

Problem Statement:
In today's data-driven world, the ability to extract meaningful insights from relational databases is essential for informed decision-making across industries. However, many business users, analysts, and even developers lack proficiency in Structured Query Language (SQL), creating a significant barrier to accessing and utilizing organizational data. And it creates a dependency on technical teams, delays in decision-making, and underutilization of valuable data resources.

Although there are tools available that attempt to simplify SQL querying through graphical interfaces or predefined templates, these solutions often suffer from limited flexibility, require custom training datasets, or are restricted to specific database management systems. This significantly restricts scalability and adaptability in real-world, multi-database environments.

Moreover, modern enterprises frequently use a variety of relational database systems including SQL Server, MySQL, PostgreSQL, SQLite, and Oracle to manage their data. Current solutions rarely support seamless, real-time interaction across this wide range of platforms, further increasing the accessibility gap for non-technical users.

Consequently, there is a pressing need for a general-purpose, AI-powered tool that can accurately translate natural language into executable SQL queries across different databases. Such a solution should allow users to intuitively query complex datasets without prior SQL knowledge, ultimately enhancing data-driven decision-making, increasing productivity, and democratizing data access across organizations.

Project Objectives:

1. To design and implement a user-friendly web interface that allows users to securely connect to various types of SQL databases such as SQL Server, MySQL, PostgreSQL, and SQLite for interactive query generation.

2. To develop a dynamic metadata extraction module that retrieves and formats database schemas in real-time to aid the AI model in contextual query generation.

3. To integrate a large language model (LLM) for translating natural language questions into valid and optimized SQL queries tailored to the schema of the connected database.

4. To execute the generated SQL queries and render the results in an interactive and readable format through the web interface for better decision-making and insight generation.

5. To implement robust logging functionality that captures user queries, generated SQL, and corresponding outputs, enabling future analysis and debugging.

6. To evaluate the accuracy, efficiency, and scalability of the LLM agent across different databases and real-world scenarios to ensure reliability and usability.

7. To ensure secure handling of database credentials and enhance the system's performance and scalability for potential future integration with enterprise systems.

Scope of the Project:
This project focuses on developing a web-based AI agent that enables natural language to SQL translation across multiple relational databases. It includes key components such as dynamic database connectivity, prompt engineering for effective query generation, SQL query validation, and a frontend interface for user interaction. The system will support popular databases including SQL Server, MySQL, PostgreSQL, SQLite, and Oracle.
The scope is limited to structured data stored in relational databases and does not include unstructured data or NoSQL systems. The investigation will cover system design, implementation, and testing within a limited academic timeframe.

The main deliverables will include a functional web application, integration with large language models (LLMs), a query logging mechanism, and documentation. This project will not cover advanced data analytics or predictive modeling beyond SQL query execution.

Significance: The system democratizes data access, empowering users across departments to retrieve insights without SQL knowledge.

2. Literature Review

Introduction
Structured Query Language (SQL) is essential for accessing relational databases, but it is often challenging for non-technical users. Natural Language Interfaces to Databases (NLIDBs) aim to convert natural language queries into SQL, and with the rise of Large Language Models (LLMs), this transformation has become more efficient [1].

Theoretical Framework
Initial NLIDBs used rule-based systems. Modern approaches leverage LLMs, enabling zero-shot and few-shot capabilities. Techniques such as Retrieval-Augmented Generation (RAG) have improved contextual understanding and SQL accuracy [2].

Historical Background

Text-to-SQL systems evolved from pattern-matching to statistical methods and now deep learning. LLMs have become a turning point in enabling better comprehension and output of SQL from text [3].

Current State of the Art

LLMs like ChatGPT show high execution accuracy in SQL generation but still face issues such as ambiguity and cross-domain generalization. New approaches aim to improve model performance and user experience [4].

Critical Analysis

Despite notable improvements, limitations such as handling nested queries, computational demands, and data privacy persist. Further research is required to address these gaps [5].

Emerging Trends

Schema-aware models, domain-specific LLMs, and lightweight architectures are being explored to enhance accuracy and accessibility of NLIDBs [6].

Synthesis and Integration

NLP and database research are converging through LLMs, providing intuitive access to data. Innovations in models and evaluation are shaping more effective NLIDBs [7].

3. Problem Analysis

In the modern data-driven world, many non-technical users, such as business analysts and managers, face challenges in retrieving insights from databases due to their lack of SQL knowledge. Traditional methods of interacting with databases require writing syntactically correct SQL queries, which presents a steep learning curve for those without technical backgrounds. This results in reduced accessibility to critical data insights that could inform decision-making processes. Moreover, organizations often operate with multiple databases such as SQL Server, MySQL, PostgreSQL, and SQLite. Manually interacting with each of these in isolation is inefficient and time-consuming. There is a clear need for a solution that allows real-time, cross-database interaction using natural language input, making data access seamless, efficient, and accessible to all levels of users.

Challenges

1. Users struggle to formulate SQL queries.
2. Limited accessibility to data insights for decision-making.
3. Need for cross-database, real-time interaction.

4. Proposed Solution

The system is powered by an intelligent AI agent using LangChain, SQLAlchemy, pyodbc, and the LLM. This integrated approach uses advanced LLMs and dynamic schema access to enable accurate, real-time query generation across multiple database systems. It includes the following core components:

- Schema Extractor: retrieves database structure to inform the LLM
- Validator & Executor: ensures SQL syntax is valid, runs the query, and returns results
- LLM Query Engine: processes natural language input and generates context-aware SQL queries

5. Research Methodology

The research methodology adopted for this project involves a systematic approach encompassing the identification of stakeholders, system requirement analysis, design, development, and validation of the proposed system. This methodology ensures that the system meets user needs and technical standards effectively.

The project follows a streamlined development process starting with requirement gathering and design, followed by schema connection and metadata extraction. Prompt engineering and AI integration enable natural language to SQL translation. Frontend and backend development ensures user interaction and system functionality, with final integration and testing to validate the solution.

Requirements Analysis:

To solve the problem effectively, the proposed system will dynamically connect to major relational databases (e.g., SQL Server, MySQL, PostgreSQL, SQLite), convert natural language queries into valid SQL, execute them, and present human-readable results. It will maintain a log of all interactions for traceability. Key non-functional requirements include secure credential management, low-latency responses, an intuitive interface for non-technical users, and a scalable architecture to support future extensions like voice input and business intelligence (BI) tool integration.

a) Functional Requirements:

i. **Language Interface:** Ability to input natural language queries.
ii. **Database Connectivity:** Connection to different databases (SQL Server, Oracle, PostgreSQL, SQLite).
iii. **Query Generation:** Conversion of user queries into valid SQL queries using LLMs.
iv. **Query Execution:** Execution of generated SQL queries and return of results to users.
v. **User Authentication:** User authentication and access control.
vi. **Error Handling:** Query validation and error handling.
vii. **Session Logging:** Logging of query sessions for audit and improvement.

b) Non-Functional Requirements:

i. **Performance:** System should respond to user queries within 5 seconds.
ii. **Scalability:** Capable of handling increased load by multiple users and databases.
iii. **Usability:** Clean, intuitive UI suitable for non-technical users.
iv. **Security:** Ensure database credentials are encrypted and access is restricted.
v. **Cross-platform Compatibility:** Should be accessible via modern web browsers.
vi. **Reliability:** System must maintain a high uptime and handle exceptions gracefully.

6. System Design

The system follows a client-server architecture where the frontend interacts with the backend via RESTful APIs. A typical data flow starts with the user submitting a query, which is passed through the backend to the LLM for SQL generation, then executed and returned to the user. The use case involves schema fetching, SQL generation, execution, and result display. The database design includes a log table to store user queries, generated SQL, and timestamps, while user credentials are handled through Backend permanent storage.

Architecture: Client-server architecture with RESTful API. Frontend communicates with backend for all operations.

Data Flow Diagram (DFD):

Level 0: User → UI → Backend API → Gemini → SQL Generator → Result Return

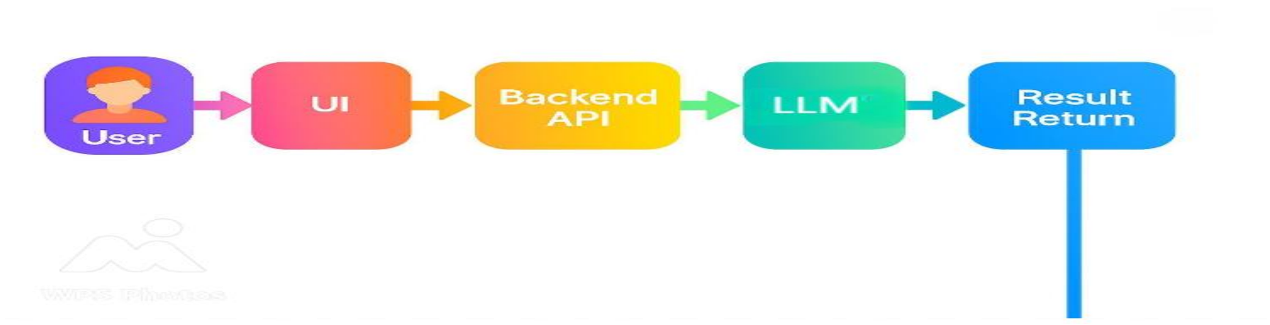

Figure 1: Data Flow Diagram of natural language to SQL LLM agent

Use Cases/Diagrams:

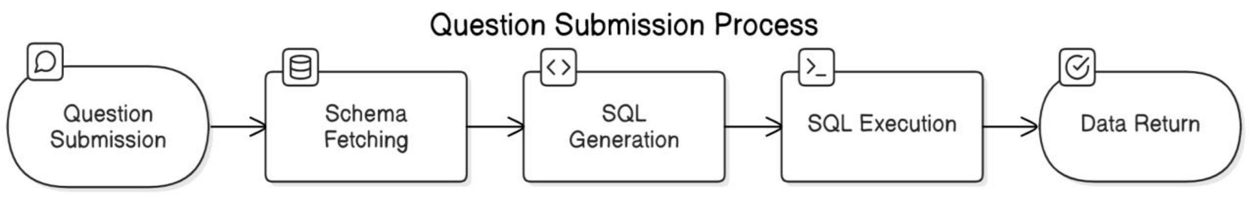

Figure 2: Use case diagram of natural language to SQL LLM agent

Database Design:

The database will be designed to support secure multi-user access, credential management for various databases, and logging of all query interactions. It ensures data integrity, scalability, and security for a smooth AI-driven experience.

Tables:

1. Users:

Stores user details including name, email, hashed password, and account creation timestamp.

2. Database_Credentials

Holds secure connection info for each user's databases, including type, host, encrypted password, and database name.

3. Query_Logs

Logs each user’s natural language query, the generated SQL, execution status, response time, and timestamp.

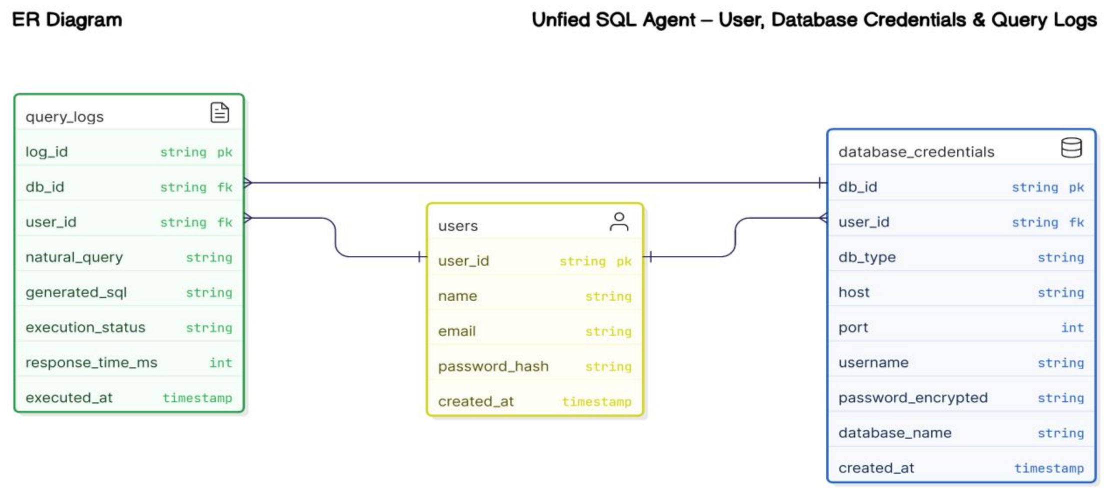

Figure 3: ER Diagram of natural language to SQL LLM agent

7. Tools and Technologies:

This project utilizes a combination of modern AI frameworks, backend technologies, and frontend tools to build a natural language-to-SQL query agent capable of interacting with multiple relational databases.

- **Programming Language & Backend Framework**:
  Python with FastAPI for building efficient and scalable APIs.

- **Database Interaction**:
  SQLAlchemy and inspection tools for dynamic schema extraction and multi-database connectivity (SQL Server, MySQL, PostgreSQL, SQLite, Oracle).

- **AI & Natural Language Processing**:
  LangChain for prompt engineering and pipeline management, integrated with a Large Language Model (LLM) API such as Google Gemini or OpenAI for converting natural language to SQL queries.

- **SQL Execution & Validation**:
  Backend utilities to validate and execute generated SQL securely and reliably.

- **Frontend Development**:
  Built using ReactJS to create a user-friendly interface tailored for non-technical users.

- **Development Process Overview**:

  - Requirement gathering and system design
  - Dynamic schema connection and metadata extraction
  - AI prompt integration and query generation
  - Frontend and backend implementation
  - System integration, testing, and evaluation

8. Expected Outcome

The proposed project will deliver a functional, web-based AI system that enables users to query and retrieve data from multiple relational databases (SQL Server, MySQL, PostgreSQL, SQLite) using natural language input. The system will dynamically connect to various schemas, translate queries using a schema-aware AI agent, execute the SQL, and return results in an easily readable format. It will also maintain logs of user interactions for tracking and future analysis.

Impact:

The system aims to democratize data access for non-technical users by eliminating the need to write SQL queries. This will empower business professionals, educators, and analysts to independently interact with databases, improving decision-making efficiency and reducing dependency on technical staff.

Evaluation Metrics:

The success of the system will be evaluated based on the following metrics:

- **SQL Accuracy**: Correctness of AI-generated SQL queries.

- **Response Time**: Speed of query generation and execution.
- **User Satisfaction**: Ease of use and perceived usefulness by non-technical users.
- **Query Success Rate**: Percentage of queries executed without errors and with meaningful results.

9. Timeline:

Month 1: Research and setup

Month 2: Backend: Schema and API setup

Month 3: Langchain and LLM integration

Month 4: SQL execution and logging

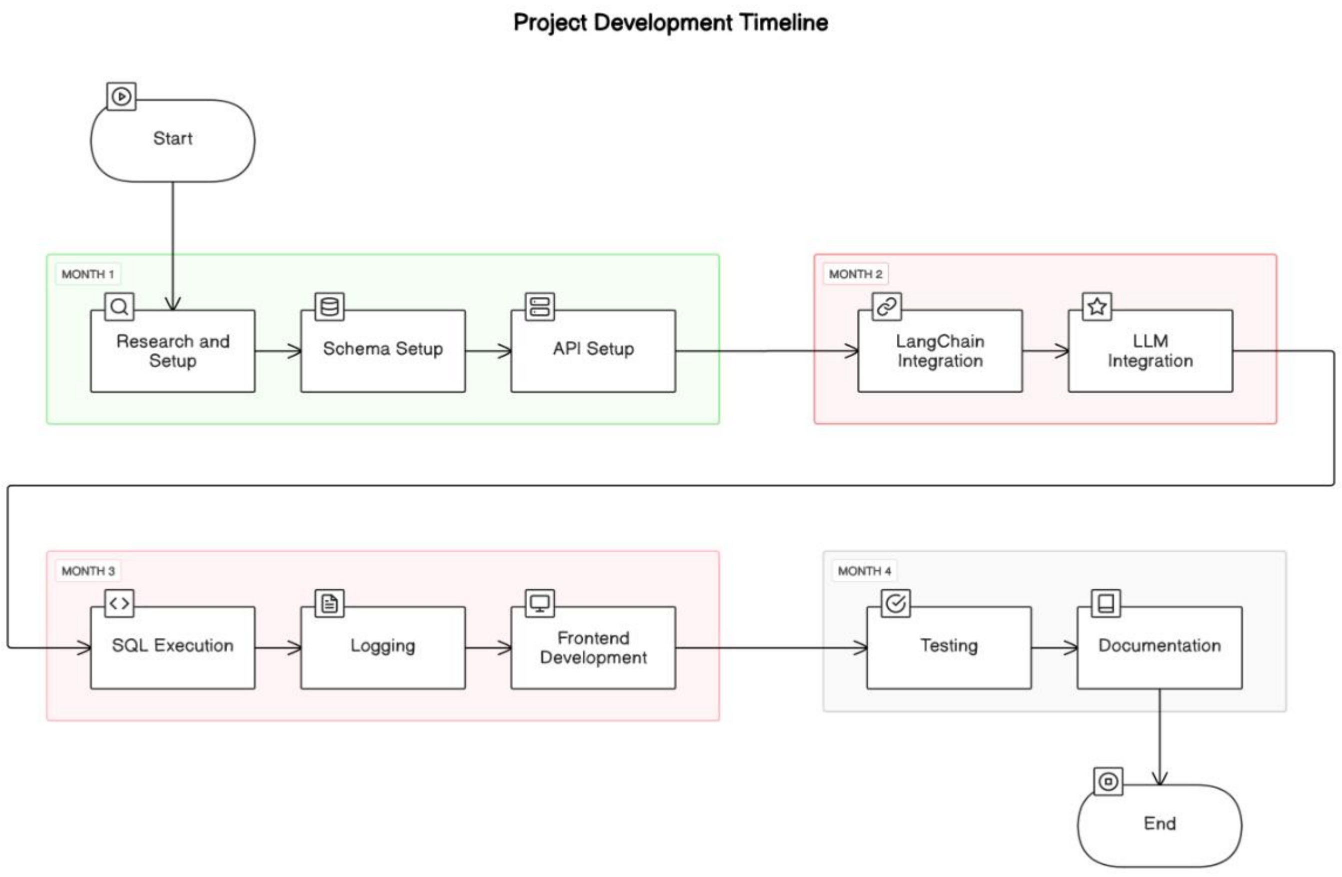

Figure 4: Timeline for development of natural language to SQL LLM agent

10. References

1. Mohammadjafari et al. (2024) reviewed the evolution of text-to-SQL systems, emphasizing LLMs and their efficiency and privacy challenges. [https://arxiv.org/html/2410.01066v1]
2. Liu et al. (2024) discussed data synthesis, prompt engineering, and evaluation metrics in LLM-powered NL2SQL systems. [https://arxiv.org/abs/2408.05109]
3. Shi et al. (2024) explored benchmark datasets, prompt strategies, and training methods. [https://arxiv.org/abs/2407.15186]
4. Xu et al. (2023) proposed schema-aware decoding to reduce errors in SQL generation. [https://arxiv.org/pdf/2105.07911]
5. Zeng et al. (2022) introduced value optimization methods to improve SQL generation from natural language. [https://arxiv.org/abs/2210.10668]
6. Li, F. & Jagadish, H. V. (2014). Constructing an Interactive Natural Language Interface for Relational Databases. VLDB.
7. LangChain Official Docs. LangChain.
8. SQLAlchemy Documentation. SQLAlchemy.