iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.71793

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Text to SQL Query

Kartik Sharma[1], Shubham Choudhary[2], Vaibhav Katyan[3], Assistant Prof. Gosiya Kaleem[4]
[1, 2, 3]*Student, Dept. Of Computer Science and Engineering, Meerut Institute of Technology, Meerut, India*
[4]*Assistant Professor, Dept. Of Computer Science and Engineering, Meerut Institute of Technology, Meerut, India*

Abstract: ***With the exponential growth of data in modern organizations, the ability to extract meaningful insights from databases has become crucial. However, interacting with structured databases often requires knowledge of SQL (Structured Query Language), which presents a barrier for non-technical users. To address this challenge, this project proposes a Text to SQL system that enables users to retrieve data from relational databases by simply expressing their queries in natural language. The goal is to bridge the gap between human language and machine-readable SQL commands through the use of Natural Language Processing (NLP) and machine learning techniques.***
The system is designed to accept a natural language input, process and understand its intent, and then convert it into an equivalent SQL query that can be executed on a target database. The core methodology involves text preprocessing, tokenization, semantic parsing, and SQL query generation. Modern NLP models, including transformer-based architectures, are explored to improve the understanding of context and the mapping between language and database schema.
Keywords: Text-to-SQL, large language model, Natural Language and understanding, Database querying, Prompt Engineering, NLP.

I. INTRODUCTION

The ability to access and analyze data has become a cornerstone of decision-making in modern organizations. Databases are the primary repositories of structured data, and SQL (Structured Query Language) is the standard tool for interacting with them. However, SQL requires technical proficiency, creating a barrier for non-technical users who need to query databases to retrieve information. This project focuses on developing a text-to-SQL conversion system that bridges the gap between natural language and SQL queries, making database interactions accessible to a broader audience. The choice of this topic stems from its relevance and impact in today's data-driven environment. In industries such as business, healthcare, education, and research, professionals often rely on data insights to guide their decisions. However, a lack of SQL knowledge forces them to depend on technical teams, leading to delays and inefficiencies. A tool that allows users to interact with databases using natural language could revolutionize workflows by eliminating these dependencies. The primary objectives of this project include designing a system that accurately converts natural language queries into SQL, ensuring compatibility with various database schemas, and maintaining a user-friendly interface. The system will leverage advanced Natural Language Processing (NLP) techniques, such as intent recognition and context understanding, to interpret user queries. Additionally, it will employ robust algorithms to handle complex queries and ambiguous inputs.

II. PROBLEM STATEMENT

- Despite the significant progress in NLP, building an effective Text to SQL system remains a complex task. Natural language is inherently ambiguous, and mapping it to a structured query requires deep understanding of both language and the underlying database schema.
- Many existing systems struggle with generalizing across multiple databases, handling complex queries involving joins, nested statements, or aggregations, and ensuring the generated SQL is syntactically correct and semantically meaningful.

III. METHODOLOGY

- The development of the The Text-to-SQL system is designed as an interactive web application where users input natural language queries, which are processed and translated into SQL using Google's Gemini Pro model. The system retrieves relevant data from a SQLite database and displays the results in a user-friendly format.
- The primary objectives of this project include designing a system that accurately converts natural language queries into SQL, ensuring compatibility with various database schemas, and maintaining a user-friendly interface. The system will leverage advanced Natural Language Processing (NLP) techniques, such as intent recognition and context understanding, to interpret user queries.

- However, a lack of SQL knowledge forces them to depend on technical teams, leading to delays and inefficiencies. A tool that allows users to interact with databases using natural language could revolutionize workflows by eliminating these dependencies.
- The ability to access and analyze data has become a cornerstone of decision-making in modern organizations. Databases are the primary repositories of structured data, and SQL (Structured Query Language) is the standard tool for interacting with them. However, SQL requires technical proficiency, creating a barrier for non-technical users who need to query databases to retrieve information.
- This project focuses on developing a text-to-SQL conversion system that bridges the gap between natural language and SQL queries, making database interactions accessible to a broader audience.
- For deployment, the fetched record are shown to the user on the Streamlit app using header and tables providing immediate feedback.

This development approach guarantees a smooth, intuitive, and enjoyable movie search experience while maintaining simplicity and efficiency in design.

IV. OBJECTIVES

1) The project aims to design an interface that accepts user queries in natural language and returns accurate SQL queries. This interface should be user-friendly, responsive, and applicable to a range of real-world database use cases.

2) At the core of the system lies the model responsible for interpreting user input and converting it into SQL. This involves identifying user intent, understanding query semantics, recognizing entities and conditions, and mapping them correctly to SQL syntax.

3) One of the primary technical objectives is to ensure that the generated SQL queries are both syntactically correct and semantically aligned with the user's input. The system should minimize errors in table/column selection, query structure, and filtering conditions.

V. FUTURE SCOPE

A. Support for Complex Queries

Extend the model to handle JOIN operations, nested queries, and aggregations. This would require a more detailed schema-aware prompt and possibly intermediate model fine-tuning.

- Instant search results with dynamic filtering
- Personalized recommendations based on browsing history

B. Schema Auto-Extraction:

Automatically extract the database schema and dynamically update the prompt to make the system adaptable to different datasets without manual configuration.

- Multi-database support
- Geo-targeted recommendations

C. Query Validation Module:

- Introduce a middleware that checks the SQL query syntax and logical structure before execution.
- Real-time availability on streaming platforms.
- This can prevent runtime errors and improve reliability.

D. User Feedback Integration:

Going forward, Users want intelligent discovery features. Your site can evolve to include:

- Implement a mechanism for users to give feedback on the generated query or correct it directly in the UI, enabling iterative improvement.
- Voice Incorporate translation tools or multilingual models to accept queries in languages other than English, expanding accessibility.

E. Visualization of Results:

- Enhance the output section by displaying results as charts, tables, or downloadable CSVs, improving usability for analytics.
- With future enhancements, it can evolve into a powerful tool applicable in education, business intelligence, healthcare, and many other domains.

VI. REQUIREMENT SPECIFICATION

Table 1: Software Requirements

OPERATING SYSTEM	WINDOWS OS/ ANY OS
IDE	VISUAL STUDIO CODE
SOFTWARES	PYTHON, STREAMLIT, GOOGLE-GEMINI, GEN-AI

Table 2: Hardware Requirements

CPU	MINIMUM 2 CORES AND 4 THREADS
RAM	MINIMUM 4 GB
MEMORY	MINIMUM 128 GB

VII. SYSTEM ARCHITECTURE

The system architecture of the "Text-to-SQL" model consists of four key layers: the frontend interface, application logic, LLM integration, and database backend. The frontend is built using Streamlit, which provides a simple and interactive user interface where users can enter natural language queries and view the results. Once the user submits a query, it is passed to the application logic layer, where the input is processed and structured into a prompt that includes the user's query and optionally the database schema. This prompt is sent to the Gemini Pro language model via the Google Generative AI API, which returns a corresponding SQL query.

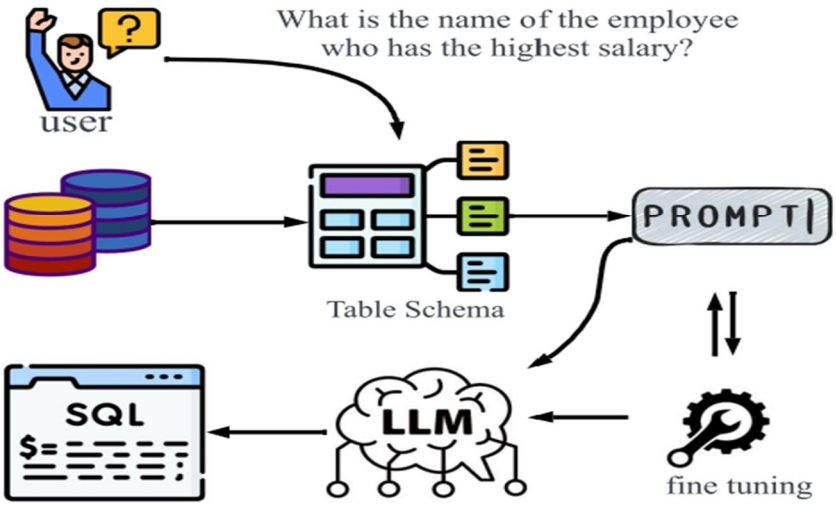


Fig.1 System Architecture

VIII. RESULT

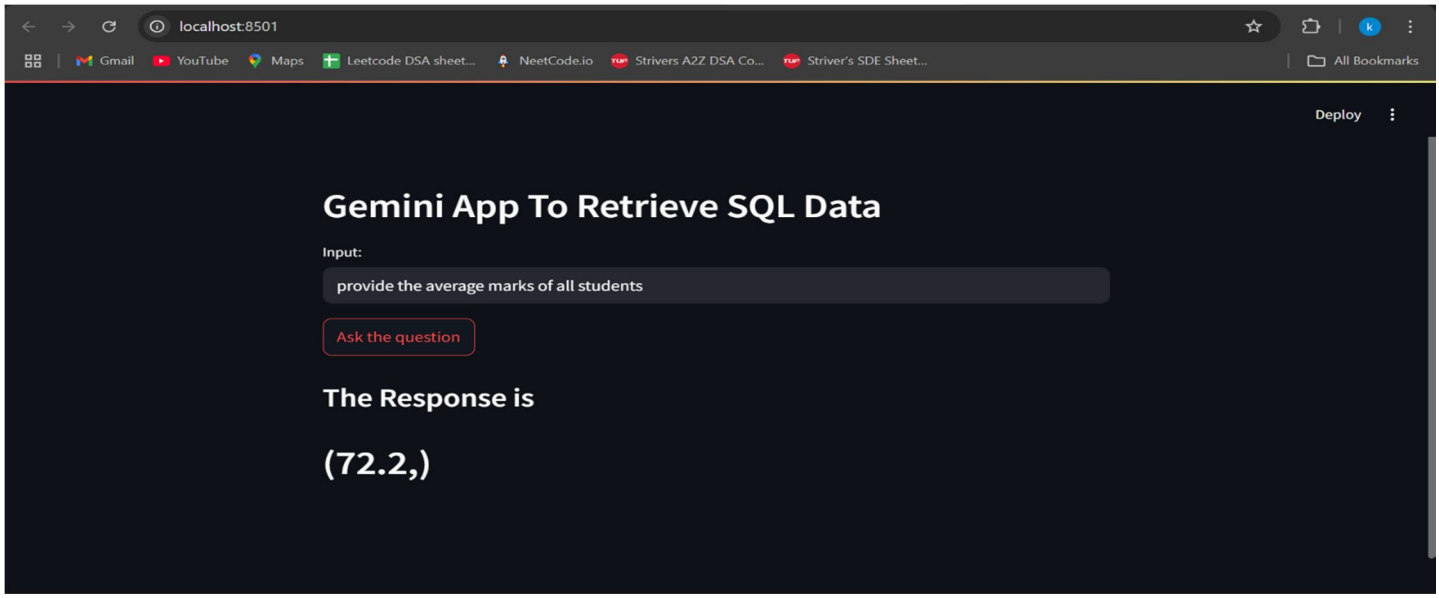


Fig. 2 Home Page

IX. CONCLUSION

1) This project successfully demonstrates the integration of a large language model (LLM) — Google's Gemini Pro — into a natural language interface that translates user queries into executable SQL commands.

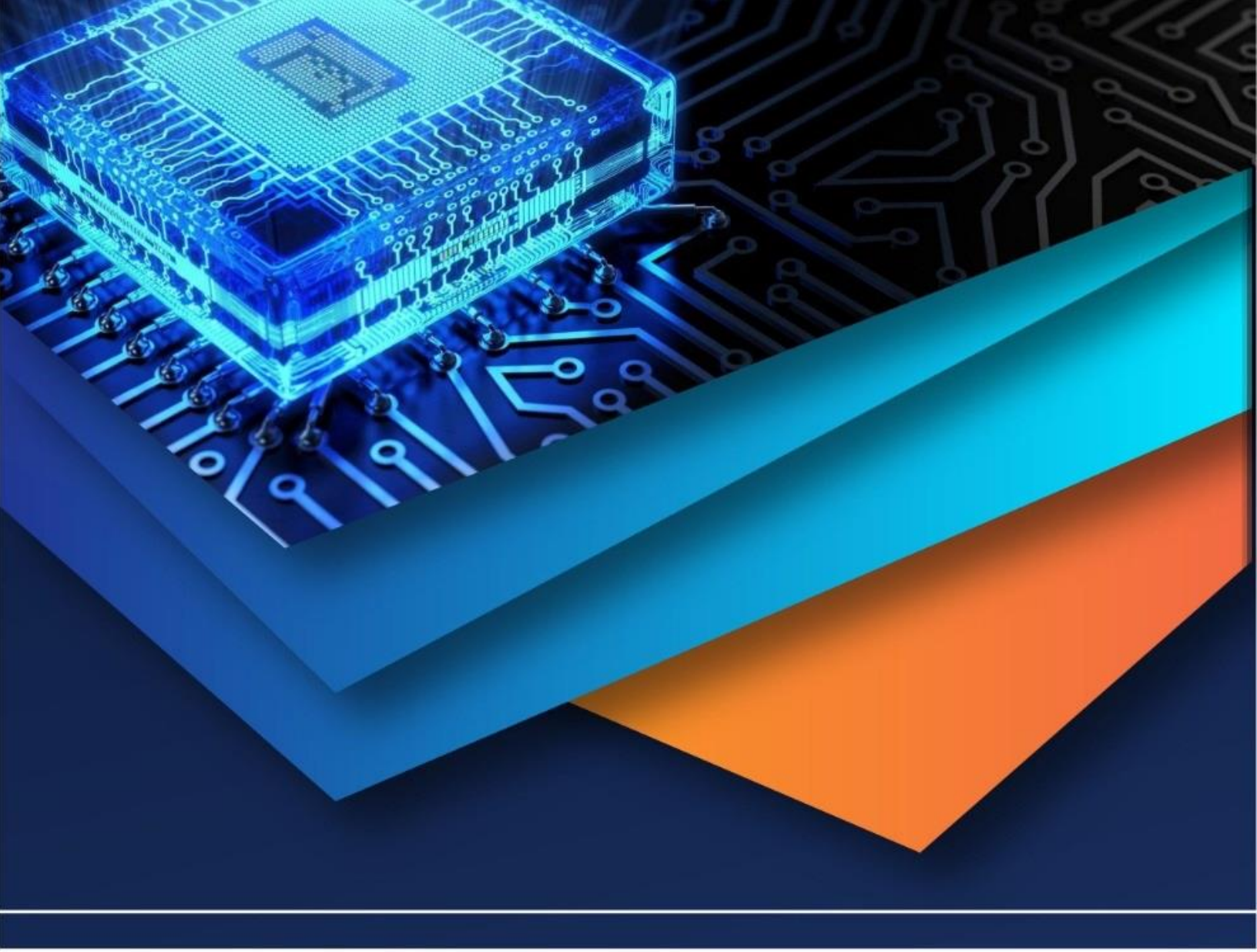
2) The key objective was to simplify data retrieval from structured databases using everyday English, making it accessible to users without technical or SQL knowledge. Using Streamlit for the front-end and SQLite as the backend database, we built a fully functional prototype that responds accurately to a wide range of natural language queries.

3) The application interface is intuitive, the processing time is minimal, and the architecture is scalable for larger and more complex datasets with minor modifications. Furthermore, the use of environment variables ensured secure API access, demonstrating good development practices in deploying LLM-based applications.

4) Overall, this project confirms that LLMs can bridge the gap between non-technical users and relational databases by removing the requirement for SQL expertise, offering a natural, conversational method of interacting with data.

REFERENCES

[1] Google Generative AI (Gemini Pro) API Documentation
https://ai.google.dev/docs
[2] Streamlit Documentation – An open-source app framework for Machine Learning and Data Science projects.
https://docs.streamlit.io/
[3] SQLite Documentation – Lightweight, disk-based database used in this project
https://www.sqlite.org/docs.html
[4] Rajkumar, S., & Liang, P. (2022). Evaluating Large Language Models for SQL Query Generation. arXiv preprint [arXiv:2204.00498].
https://arxiv.org/abs/2204.00498
[5] Finegan-Dollak, C., et al. (2018). Improving Text-to-SQL Evaluation Methodology. Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).
https://aclanthology.org/P18-1033/
[6] OpenAI Cookbook – Guides and code samples for building with LLMs
https://github.com/openai/openai-cookbook
[7] Python Official Documentation – Programming language used to build the application https://docs.python.org/3/
[8] Vaswani, A., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems.
https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[9] .env (dotenv) Python Library Documentation – For managing environment variables securely
https://pypi.org/project/python-dotenv/
[10] Yu, T., et al. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task.
https://arxiv.org/abs/1809.08887



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY