International Journal of Engineering Applied Sciences and Technology, 2025
Vol. 9, Issue 11, ISSN No. 2455-2143, Pages 39-43
Published Online March 2025 in IJEAST (http://www.ijeast.com)

AI POWERED TRANSLATOR: TRANSFORMING NATURAL LANGUAGE TO DATABASE QUERIES

G S N Murthy
Professor, Department of CSE
Aditya College of Engineering and Technology,
Surampalem, Andhra Pradesh, India – 533437

K Anshu, T Srilakshmi, CH Sumanth, M Naga Mounika
UG Scholar, Department of CSE
Aditya College of Engineering and Technology,
Surampalem, Andhra Pradesh, India – 533437

Abstract- **In the modern digital landscape, interacting with databases often requires structured query languages SQL or NoSQL syntax, which can be a barrier for non-technical users. This project introduces an intelligent AI-powered system that seamlessly converts natural language questions into executable database queries, bridging the gap between human communication and database management. Leveraging advanced natural language processing (NLP) models, large language models (LLMs), the system dynamically interprets user queries and translates them into optimized SQL or NoSQL commands. Our approach involves extracting schema details from both relational (MySQL) and non-relational (MongoDB) databases to generate a contextualized query prompt for AI processing. The model ensures precise query formation while adhering to database constraints and syntax rules. Additionally, security measures, including authentication and input validation, are integrated to prevent SQL injection and unauthorized access. This AI-driven solution enhances accessibility, enabling users to retrieve, insert, update, and delete data without technical expertise. It finds applications in data analytics, customer support, and business intelligence, streamlining database interactions and improving user efficiency.**

Keywords: **Natural language processing (NLP),SQL and NoSQL Query Generation, Flask Web Framework , MongoDB and MySQL Integration, CRUD Operations**

I.INTRODUCTION

The AI-Powered Natural Language to Database Query System is designed to simplify database interactions by enabling users to retrieve and manage data using everyday language. Many non-technical users face challenges in writing complex SQL or NoSQL queries, making data access difficult and inefficient. This project aims to bridge that gap by providing an AI-driven platform that translates natural language queries into executable database commands, enhancing accessibility and usability.

One of the key features of this system is its intelligent query interpretation engine, powered by advanced large language models (LLMs). The system dynamically understands user intent and converts it into structured queries for relational (MySQL) and non-relational (MongoDB) databases. By analyzing database schemas and context, it ensures accurate and optimized query generation while adhering to syntax constraints.

To improve efficiency, the platform incorporates query validation, optimization, and security measures. Features like authentication, SQL injection prevention ensure secure database interactions. Additionally, the system provides query analytics and recommendations, helping users refine their searches and improve data retrieval efficiency.

The project also includes a context-aware learning mechanism, allowing the AI to improve query understanding based on user interactions over time. Users can perform CRUD (Create, Read, Update, Delete) operations effortlessly without requiring database expertise, making it ideal for business analysts, customer support teams, and decision-makers.

Furthermore, the platform offers real-time query suggestions, streamlining the query-building process. By integrating AI-driven insights, real-time query execution, and secure database management, this project transforms how users interact with databases. It empowers individuals and organizations to access and manipulate data effortlessly, improving productivity and decision-making.

International Journal of Engineering Applied Sciences and Technology, 2025
Vol. 9, Issue 11, ISSN No. 2455-2143, Pages 39-43
Published Online March 2025 in IJEAST (http://www.ijeast.com)

II.PROPOESD SYSTEM

The proposed system introduces an AI-driven natural language interface that enables users to interact with databases effortlessly by converting human language into executable SQL and NoSQL queries. This system eliminates the need for technical expertise, allowing non-technical users to efficiently retrieve, insert, update, and delete data using simple natural language inputs.

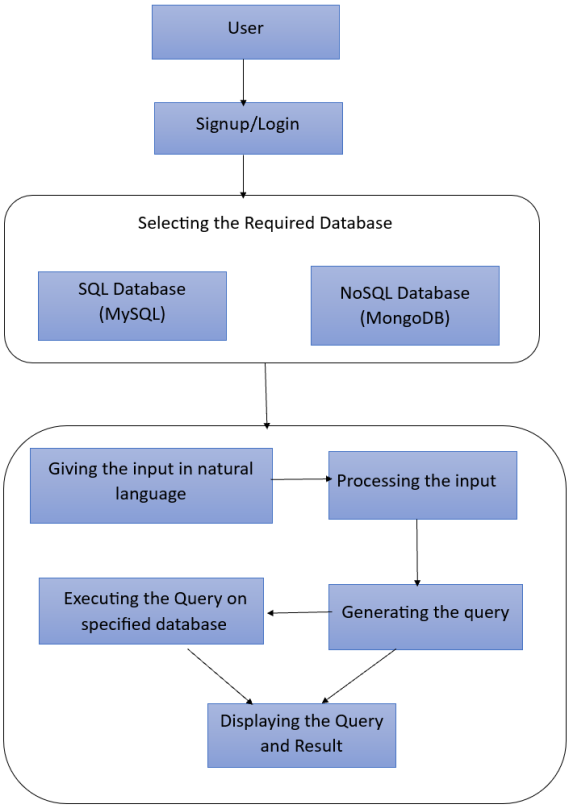
The core functionality of the system revolves around an advanced NLP model that dynamically interprets user queries and translates them into optimized SQL (for relational databases like MySQL) and NoSQL (for MongoDB) commands. To ensure accuracy, the system extracts schema details from databases to generate a context-aware query prompt, minimizing errors and enhancing query precision.

To enhance security and reliability, the system integrates authentication mechanisms and input validation to prevent unauthorized access. Additionally, an interactive query assistant helps users refine their queries by providing auto complete suggestions, error detection, and optimization tips.

By combining AI-powered NLP, intelligent query processing, security measures, and a user-friendly interface, this system provides a comprehensive, accessible, and efficient solution for database management, catering to users across diverse industries.

III.WORKING

1.System architecture:



The system architecture of the AI-powered natural language to database query translator is designed to enable seamless interaction between users and databases using plain English inputs. The process begins with the user, who is required to either sign up or log in to gain access to the system. This authentication step ensures secure and personalized usage. Once authenticated, the user selects the type of database they wish to interact with—either a traditional SQL database like MySQL or a NoSQL database such as MongoDB. This selection dictates the type of query that will be generated later in the process. Following the selection, the user provides their input in natural language, which is then processed using Natural Language Processing (NLP) techniques. This involves analyzing the user's intent, identifying key entities, conditions, and parsing the structure of the input sentence. The processed input is then passed to a query generation engine, which constructs a corresponding database query. For SQL databases, this results in a structured SQL query, while for NoSQL databases, it creates an equivalent command suitable for MongoDB or other NoSQL systems. Once the query is

International Journal of Engineering Applied Sciences and Technology, 2025
Vol. 9, Issue 11, ISSN No. 2455-2143, Pages 39-43
Published Online March 2025 in IJEAST (http://www.ijeast.com)

generated, it is executed on the selected database, performing the intended operation such as retrieving, inserting, updating, or deleting data. The final step involves displaying both the generated query and the results fetched from the database to the user. This transparent approach helps users understand how their natural language request was interpreted and executed. Overall, the architecture supports both SQL and NoSQL platforms, enhances user accessibility by removing the need to learn complex query languages.

2.User Authentication Module
This module is responsible for user registration and login. During registration, users provide details such as username, email, password, and other necessary information. Passwords are securely stored using encryption techniques (e.g., bcrypt or SHA hashing). When logging in, credentials are verified, granting access only to authenticated users. For example, if a user enters incorrect credentials, they receive a message like "Invalid username or password." After successful login, the system redirects them to the database selection module.

3.Database Selection Module
Once logged in, users must choose between SQL (MySQL) and NoSQL (MongoDB) for query execution. This module ensures that all queries are correctly structured based on the selected database. For example, if a user selects SQL, the system generates queries using MySQL syntax (SELECT * FROM users WHERE age > 25;). If NoSQL is chosen, queries are formatted in MongoDB syntax ({ age: { $gt: 25 } }). This module ensures that only valid queries are passed to the execution stage, preventing errors due to incorrect syntax.

4.Natural Language Query Processing Module
This module converts human-readable text into a structured database query AI-powered NLP models. The system analyzes user input, understands the intent, and generates the appropriate SQL or NoSQL query. For example, if a user enters "Show all students from the Computer Science department," the system generates:

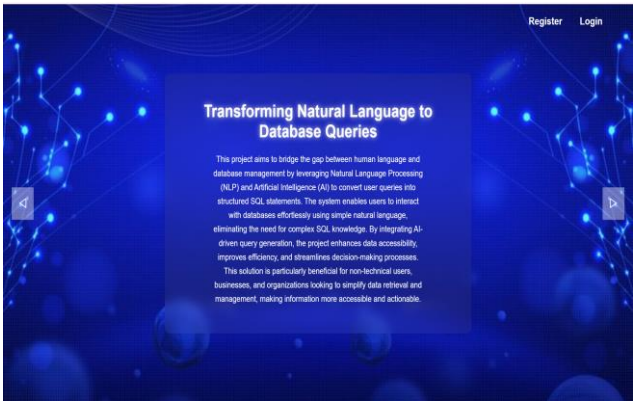
- SQL Query: SELECT * FROM students WHERE department = 'Computer Science';
- NoSQL Query: { "department": "Computer Science" }

It also handles variations in user queries, such as "List all CS students" or "Get students studying Computer Science", ensuring flexibility.

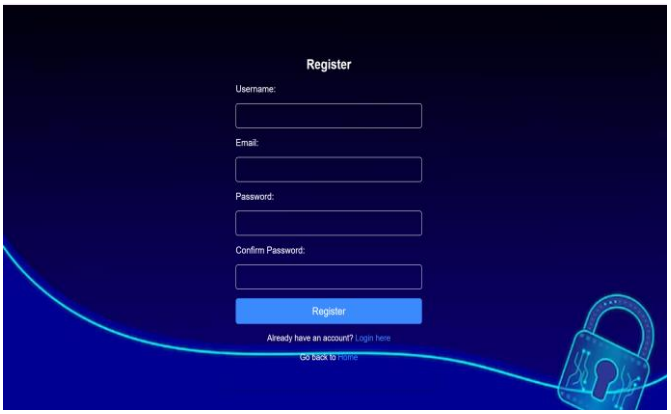
5.Query Execution Module
This module is responsible for executing the generated SQL or NoSQL queries on the respective database. If the query is valid, the system retrieves and processes the results. It also includes error handling mechanisms to manage incorrect queries, missing database tables, or syntax errors. For example, if a user mistakenly enters "Get users age 25", the module corrects it to SQL (SELECT * FROM users WHERE age = 25;) or NoSQL ({ age: 25 }). If an error occurs, an appropriate message like "Invalid column name: users" is displayed.

6.Result Display Module
The Result Display Module is responsible for presenting query results in a structured and user-friendly manner. After executing the generated SQL or NoSQL query, this module formats and displays the data in a way that is easy to read and understand. The results are shown in different formats based on the type of database selected.
Additionally, this module will also display errors if the schema does not match the expected structure. If the query references a non-existent table, column, or incorrect data type, the system provides clear error messages. For example, if a query attempts to access a column that is not present in the database schema, an error message such as "Column 'employee_age' does not exist in table 'employees'." will be displayed. This ensures that users can identify and correct issues efficiently, improving the reliability and accuracy of query execution.

IV.OUTPUTS

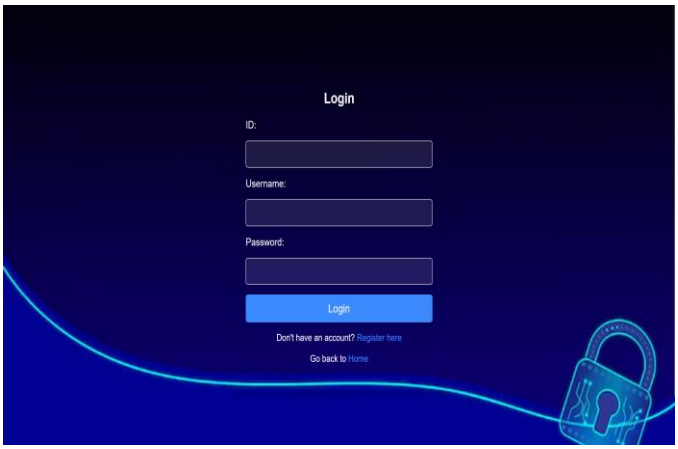


1. Home Page

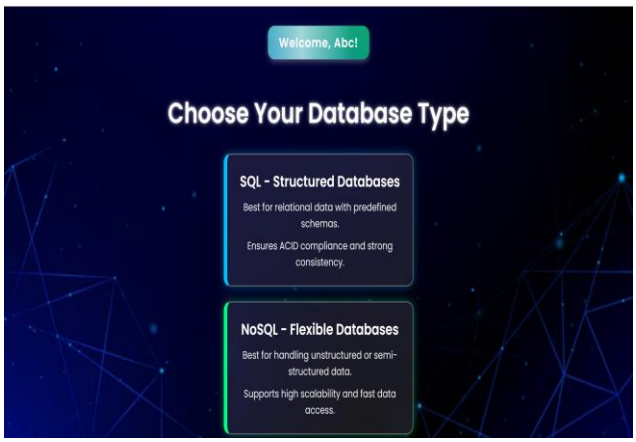


2. Registration Page

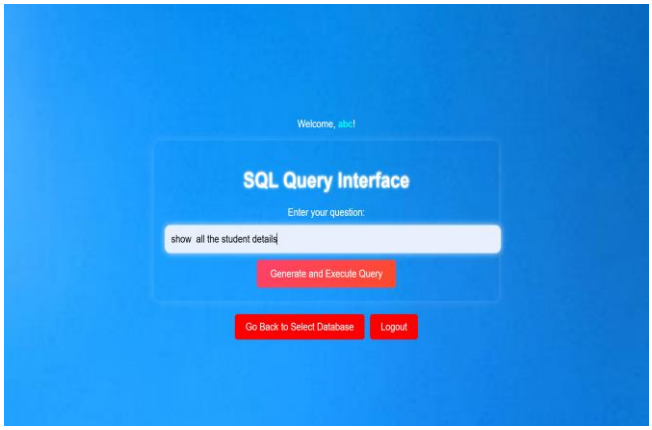
International Journal of Engineering Applied Sciences and Technology, 2025
Vol. 9, Issue 11, ISSN No. 2455-2143, Pages 39-43
Published Online March 2025 in IJEAST (http://www.ijeast.com)



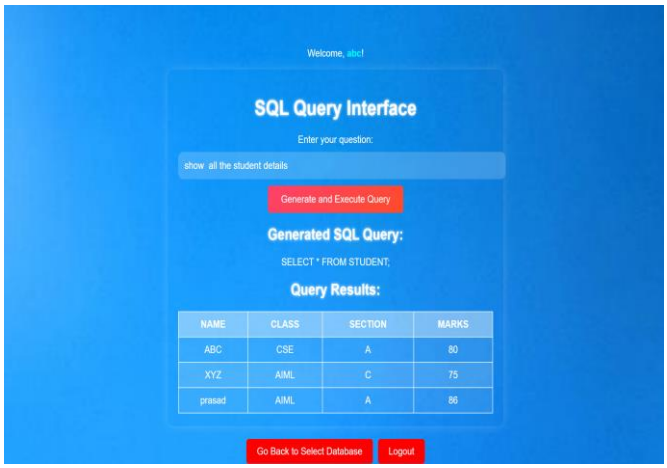
3.Login Page



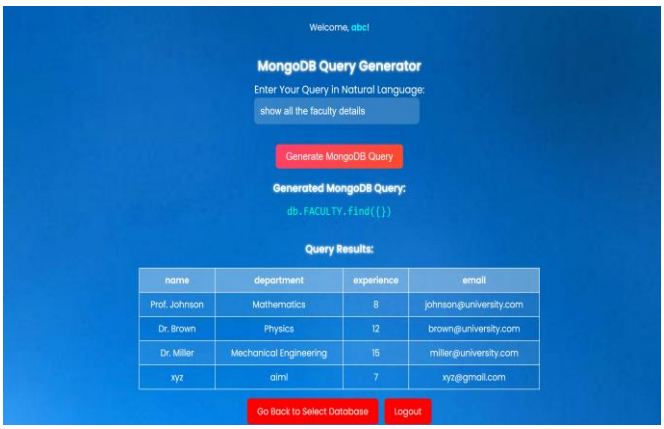
4. Database Selection Page



5.Sql Query Page



6. Generated Query and Result



7.NoSql Query Page

V.CONCLUSION

The Natural Language to Database Query System transforms the way users interact with databases by enabling seamless conversion of natural language inputs into SQL and NoSQL queries. By integrating AI-driven text processing, this system simplifies database querying for users with minimal technical expertise, making data retrieval more intuitive and efficient. Through structured modules such as User Authentication, Query Selection, Query Generation, Query Execution, and Result Display, the platform ensures a smooth workflow from user login to result visualization. The intelligent processing mechanism enhances accuracy by handling schema mismatches, providing error messages, and refining query structures for better performance. This project not only improves accessibility to database systems but also has potential applications across multiple domains, including business intelligence, data analytics, customer support, and enterprise solutions. By bridging the gap between human language and database interactions, it contributes to the advancement of AI-powered data management, making information retrieval faster, more accurate, and user-friendly.

VI.REFERNCES

[1]. Xu X., Liu C., and Song D. (2017). SQLNet: Generating structured queries from natural language without reinforcement learning, arXiv preprint arXiv:1711.04436.

[2]. Zhong V., Xiong C., and Socher R. (2017). Seq2SQL: Generating structured queries from natural language using reinforcement learning, arXiv preprint arXiv:1709.00103.

[3]. Huang P.-S., Wang C., Singh R., Yih W.-t., and He X. (2018). Natural language to structured query generation via meta-learning, arXiv preprint arXiv:1803.02400.

[4]. Mahmud T., Hasan K. A., Ahmed M., and Chak T. H. C. (2015). A rule-based approach for NLP-based query processing, in Proc. 2nd Int. Conf. on Electrical Information and Communication Technologies (EICT), IEEE, (pp. 78–82).

[5]. Han J., Haihong E., Le G., and Du J. (2011). Survey on NoSQL databases, in Proc. 6th Int. Conf. on Pervasive Computing and Applications, IEEE, (pp. 363–366).

[6]. Parker Z., Poe S., and Vrbsky S. V. (2013). Comparing NoSQL MongoDB to an SQL DB, in Proc. 51st ACM Southeast Conference, ACM, (p. 5).

[7]. Dong L. and Lapata M. (2016). Language to logical form with neural attention, arXiv preprint arXiv:1601.01280.

[8]. Anand P. and Farooqui Z. (2017). Rule-based domain-specific semantic analysis for natural language interface for database, Int. J. of Computer Applications, 164(11).

[9]. Nandhini S., Viruthika B., Saba A., and Das S. S. (2019). Extracting SQL Query Using Natural Language Processing, Int. J. of Eng. and Adv. Tech. (IJEAT), 8(4), ISSN: 2249–8958.

[10]. Parvat A., Chavan J., Kadam S., Dev S., and Pathak V. (2017). A survey of deep-learning frameworks, in Proc. Int. Conf. on Inventive Systems and Control (ICISC), IEEE, (pp. 1–7).

[11]. Addison T. and Vallabh S. (2002). Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers, in Proc. SAICSIT, (pp. 128–140).

[12]. Babar M. A. and Paik H. (2009). Using Scrum in Global Software Development: A Systematic Literature Review, in Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE Int. Conf., (pp. 175–184).

[13]. Yu T., Zhang R., Lin Q., and Radev D. (2018). Typesql: Knowledge-based type-aware neural text-to-sql generation, in Proc. of the NAACL-HLT, (pp. 588–594).

[14]. Herzig J. and Berant J. (2020). Span-based semantic parsing for compositional generalization, in Proc. of the 58th Annual Meeting of the Association for Computational Linguistics, (pp. 908–922).

[15]. Li F., and Jagadish H. V. (2014). Constructing an interactive natural language interface for relational databases, VLDB Endowment, 8(1), (pp. 73–84).