ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

AI-Powered SQL Bug Analyzer and Auto Fix Assistant

Dr. V. S. Gaikwad[1], Ganesh Borole[2], Omkar Jadhav[2], Himanshu Tambe[2] and Krishna Kinikar[2]

[1]Head of IT department

[2]Student Department of Information Technology, Trinity College of Engineering and Research, Pune, India

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150500252

Received: 30 May 2026; Accepted: 06 June 2026; Published: 23 June 2026

ABSTRACT

This work describes the development of a system based on AI and chat technology that allows users to communicate with several databases by posing questions in natural language. Standard databases require knowledge of SQL and technical skills from their users; therefore, they are difficult to use for individuals who lack expertise in computer science and programming. In this work, we incorporate MERN stack technology and integrate AI technology within it to translate user commands into valid queries for databases. Our proposed system is capable of processing queries and communicating with different types of databases, including MongoDB, MySQL, SQLite, and BigQuery..

Keywords - Artificial Intelligence,Text-to-SQL, SQLFixAgent, Large Language Models, SQL error correction, Multi- agent framework, Semantic accuracy

INTRODUCTION

Database interaction is one of the key operations performed by almost any application nowadays, however, writing correct queries is quite complicated. Users need to know syntax and structure of the queries as well as their logical sense. In addition, even minor error might lead to inaccurate result or system malfunctioning, which may affect productivity negatively..In order to make queries more accessible to developers and regular users alike, several tools were created that allow users to enter a command in plain English that would automatically be converted into a database query. While these approaches seem promising at first sight, there's one main problem related to such platforms – inability to properly capture user's intentions, which means that resulting queries could still be wrong from logical point of view.

In order to fix aforementioned issue and make databases more accessible and easier to work with, the proposed system introduces an AI-powered multi-database chat application developed within MERN stack architecture. As opposed to other similar services, it implements intelligent pipeline of query processing, which means that a query will be entered via a chat window and processed by an AI to produce a correct database operation.

LITERATURE SURVEY

The issue of comprehending, constructing, and correcting SQL queries has been extensively researched in database systems, machine learning, and natural language processing. A number of solutions have been suggested over time, starting with rules-based methods all the way through to sophisticated AI-powered models.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Traditional SQL Error Detection Techniques

Initial studies concentrated on techniques such as rule-based systems and signature-based systems for identifying SQL errors and vulnerabilities. AMNESIA and SQLCheck, for instance, utilized pre-defined rules and pattern matching to determine any syntax problems or possible injection attacks within queries. These techniques proved useful for detecting known patterns, they struggled with dynamic queries and failed to adapt to new or unseen error types. These systems also produced a high number of false positives and lacked flexibility in modern web applications.

Machine Learning-Based Approaches

Due to the development of machine learning algorithms, scientists have developed supervised and unsupervised machine learning algorithms for improving the performance of SQL error detection. In the former approach, supervised machine learning algorithms like SVM, Random Forest, and Naïve Bayes classification algorithms are applied to the data sets that are labeled into two classes: queries that do not contain any errors and those that include errors. The latter approach includes clustering and anomaly detection algorithms like Isolation Forest.

Despite their superior results compared to the traditional algorithm- based approaches, these algorithms depend heavily on huge and accurate data sets..

Deep Learning and NLP-Based Systems

In recent times, deep learning approaches have been examined to treat SQL queries as structured natural language. For example, LSTM, BiLSTM, CNN and Transformer-based models like BERT have been utilized in order to analyze the syntax and semantics of SQL statements. This analysis of SQL statements involves embeddings along with sequence modeling to learn the relationships between elements of the SQL statement. Deep learning-based models have demonstrated substantial improvements in terms of robustness and precision. They are efficient in recognizing semantic errors as well as handling text-to-SQL mappings.

AI-Based Text-to-SQL and Multi-Agent Systems

Today, there has been a move towards intelligent systems that can take natural language inputs and turn them into SQL queries. Some models used include Seq2SQL and RAT-SQL which utilize deep learning and reinforcement learning techniques to translate queries into SQL code. Nonetheless, such models tend to struggle with accuracy and ambiguity of queries.

Multi-agent AI models have thus been developed where various agents will play distinct roles in generating queries, validating, and refining queries, among other roles.

Hybrid and AI-Enhanced Frameworks

Hybrid methods that integrate static and dynamic analysis along with AI- driven reasoning have been highlighted by recent literature. Such approaches analyze not only the structure but also the run-time performance of queries to identify any anomalies. In addition, techniques such as anomaly detection, adversarial learning, and explainable artificial intelligence (XAI) are being employed to increase system robustness and transparency.

These approaches are more flexible and able to address changing query patterns.

Improvement with Machine Learning Techniques

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Machine learning models significantly improved the detection of SQL anomalies by learning patterns from data. Supervised models provided good accuracy for known query types, while unsupervised models enabled detection of unknown or zero-day anomalies. However, their performance depended heavily on dataset quality and size

Deep Learning Enhanced Semantic Understanding

Deep learning and NLP-based approaches demonstrated better capability in understanding both syntax and semantics of SQL queries. Models such as LSTM, BiLSTM, and Transformers improved performance in handling complex queries and natural language inputs.

Rise of Text-to-SQL Systems

Text-to-SQL systems made database interaction easier by allowing users to write queries in natural language. While these systems improved accessibility, they often failed to capture exact user intent, leading to logically incorrect queries despite correct syntax

Research Gaps Identified

Though there have been many breakthroughs in the domain of processing SQL queries using artificial intelligence techniques, a number of problems remain unsolved till date. The common problem among all text-to-SQL systems and even machine learning techniques is their inability to interpret the intention behind user-generated queries, resulting in logically wrong outputs despite syntactically correct results. Apart from that, though some systems excel in error detection, they are unable to detect semantic errors in queries.

Another critical limitation associated with many text-to-SQL systems is their heavy dependence on large-scale, high-quality training data that are not available in case of rare query patterns. Another problem faced by various systems is their inability to learn in order to generalize unseen queries or dynamically changing databases as well as nesting in queries. Existing solutions also lack user- friendly real-time user interfaces that make them impractical to use. On top of that, the majority of text-to-SQL systems are single database-oriented systems and cannot handle multi- database environments. There is clearly a need for designing a multi-platform, real-time text-to-SQL system that uses artificial intelligence to perform its functions

Problem Definition

Sr No	Reference (Year)	Core Functionality	Key Contribution	Limitation
1	Cen et al 2025	Multi-agent Text-to-SQL system for query parsing and correction	Introduced SQLFixAgent with multi-agent system for SQL error detection and correction	Highly System complexity
2	Sergeyuk et al,2024	Study of AI-based coding assistants in real-world development	Identified common use cases like testing and bug fixing; highlighted developer	Context loss; low trust in outputs

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

			adoption trends	
3	Christopher Troy et al,2023	Framework for automatic SQL generation using grammar-based approach	Used ANTLR4 and EBNF grammars for structured SQL generation with strong syntax control	Limited semantic understanding; struggles with complex natural language queries
4	Zhou et al,2022	Analysis of AI tools like Copilot in coding environments	Highlighted usability, integration challenges, and need for better explanation features	Trust issues and lack of transparency in generated outputs;

Table I. Literature Survey of SQL Agent

Systems

Key Findings from Literature

The review of existing research on SQL query processing, error detection, and AI-based systems highlights several important observations:error detection, and AI-based systems highlights several important observations: Interacting with modern databases is still challenging for non-tech users owing to syntax and semantic issues.

Conventional approaches and general-purpose LLMs do not have the capability to perform schema grounding in real-time, leading to hallucinations and user inconvenience while debugging. The present work aims to address the need for a system that can unify the process and heal itself from dialect mismatches in relational and warehousing contexts.

System Architecture

Frontend-Layer

This represents the initial point of interaction within the system and allows users to engage via a chat-based application. Queries can be entered using normal language (such as “Display top customers”) or file/image uploads if the application supports them. The front-end is developed using React.js technology, which offers functionalities such as chat history and instant responses.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

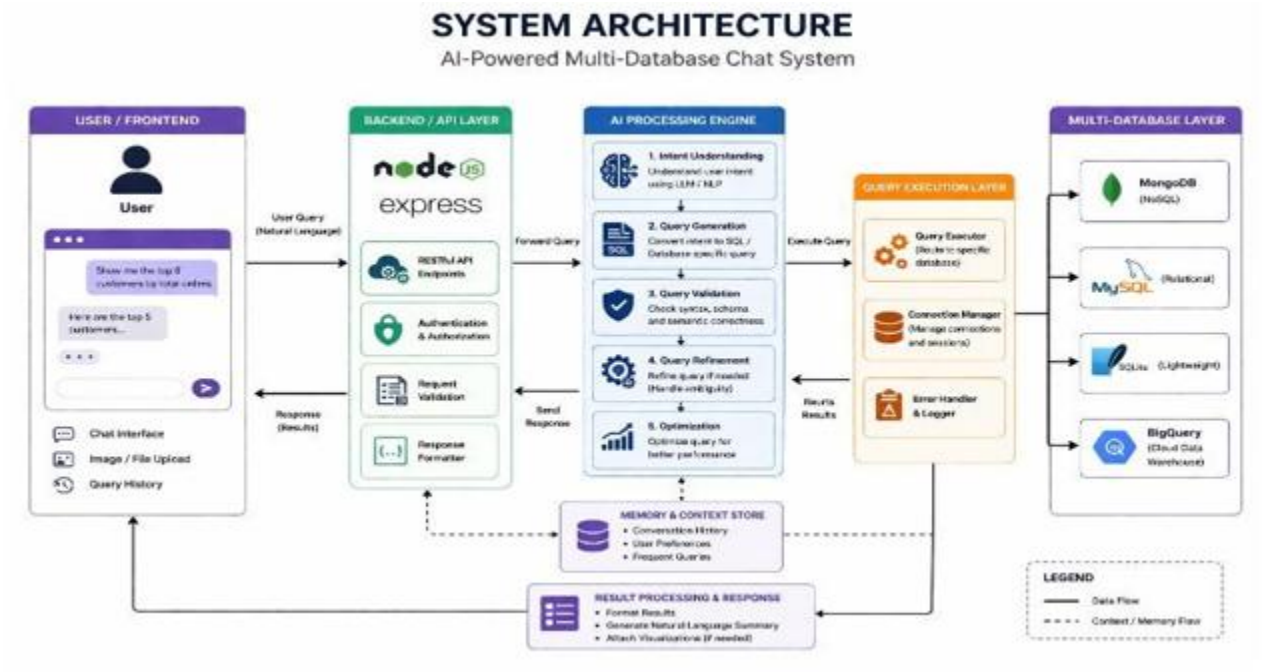


Fig 1. Proposed System Architecture

Backend Layer

The backend, built using Node.js and Express.js, acts as a mediator between the frontend and the AI engine. It is responsible for routing API calls and handling incoming requests efficiently, while also managing authentication and authorization to ensure secure access. Additionally, it validates input data to prevent errors or misuse and formats responses in a structured manner before sending them back to the frontend for proper display.

AI Processing Engine

This module forms the intelligent core of the system, handling user queries through a structured, multi-step process. It begins with intent determination, using NLP or LLM techniques to understand the user's request. Next, it performs query formulation by translating natural language into SQL or database queries. The generated queries are then validated for both syntax and semantic correctness, followed by query optimization to enhance performance and efficiency. This comprehensive approach ensures that the final queries produced are both accurate and efficient.

Query Execution Layer

After query generation, this layer takes care of executing the query:

1. Query Executor directs the query to the right database
2. Connection Manager takes care of the database connection
3. Error Handler & Logger deals with errors system logging

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Multiple Database Layer

This system supports multiple databases, providing flexibility for real-world applications. It includes MongoDB for non-SQL data handling, MySQL and SQLite for SQL-based operations, and BigQuery for scalable, cloud-based SQL querying.

Memory & Context Store

This module stores conversation history, user preferences, and frequently used queries, enabling personalized interactions, improving response relevance, and ensuring continuity across sessions for a more efficient and user-friendly experience

Technology Stack

The system is developed using a robust, full-stack architecture designed to bridge the gap between natural language processing and heterogeneous database management. The frontend layer focuses on real-time interaction and visual data handling, built using React.js for a component-based, high-performance user interface. Styling is handled with Tailwind CSS to ensure a modern and professional look, while React Flow enables users to create UML or class diagrams through a drag-and-drop canvas. Communication between the frontend and backend is managed using Axios for seamless data exchange.

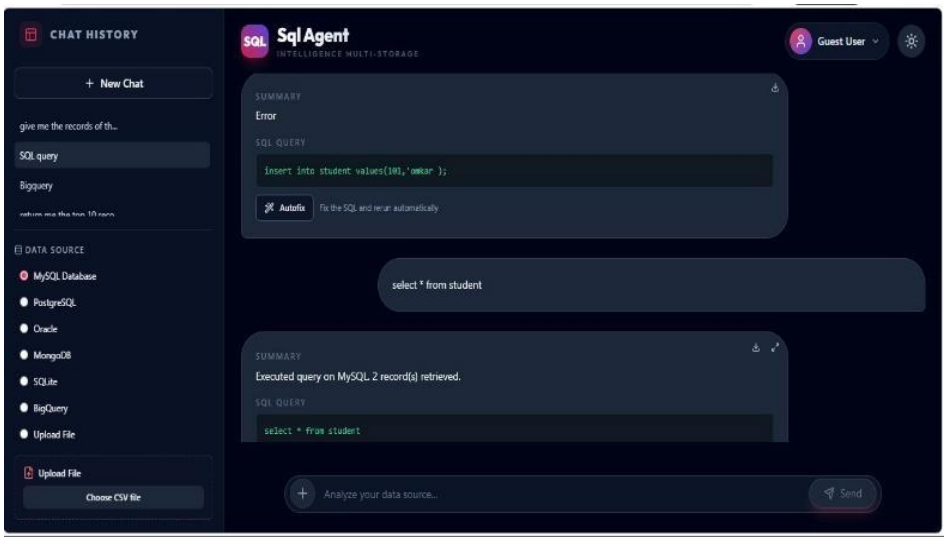
The backend and orchestration layer ensures secure, scalable, and efficient system operations. It uses JWT for secure user authentication and session management. The system supports multiple databases, including MySQL and PostgreSQL for structured data storage, along with Google BigQuery for large- scale data analytics. Additionally, MongoDB is used to store unstructured data and maintain a failure memory repository for tracking query errors and fixes, while SQLite supports lightweight, file-based operations and rapid prototyping.

RESULTS AND EVALUATION

Successful Execution of Queries

The system executes SQL queries that are provided by the user via the chat interface. As evidenced from the output below, the system was able to execute the query select * from student, returning 2 results from the MySQL database.

Fig 1. Example Proper UI visible



ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Detection of Errors and Auto-Fixing Facility

This system has the capability to detect any errors in the query. As in the case of the query insert into student values(101,'omkar'); an error was encountered, and the same was detected by the system. AutoFix provides suggestions for correction and helps users run the query again without much effort.

The system enables users to communicate with it through basic text commands rather than having to formulate intricate SQL commands. The system then interprets the intentions of the users and executes the queries accordingly.

The application has an option to link with various databases like MySQL, PostgreSQL, Oracle, MongoDB, SQLite, and BigQuery.

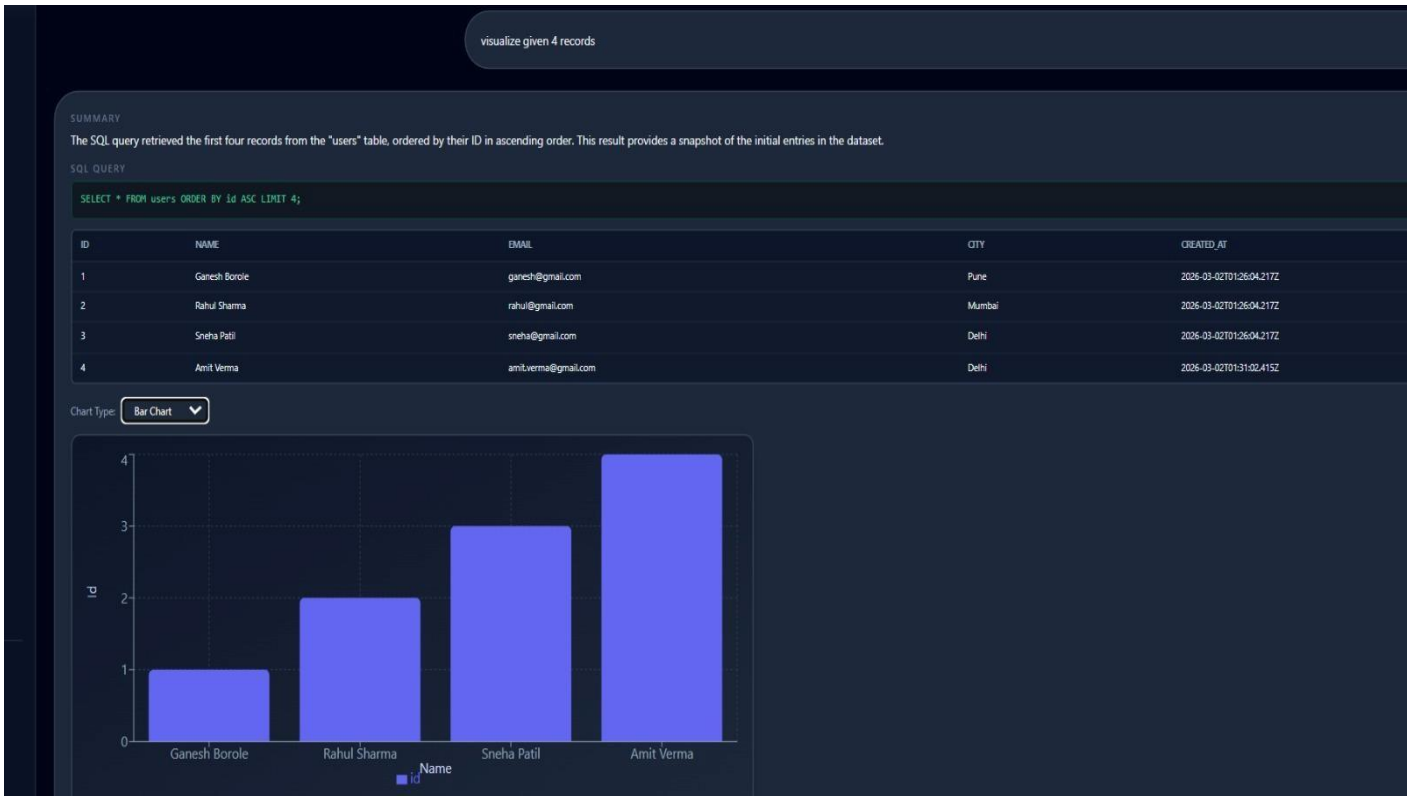
The chat interface allows users to interact with the system using functions like: Chat history, Query summary, Upload files (supports CSV format), Immediate response visualization

The system processes queries in real-time with minimal delay. The integration of AI reduces the time required for writing and debugging queries, thereby increasing overall efficiency

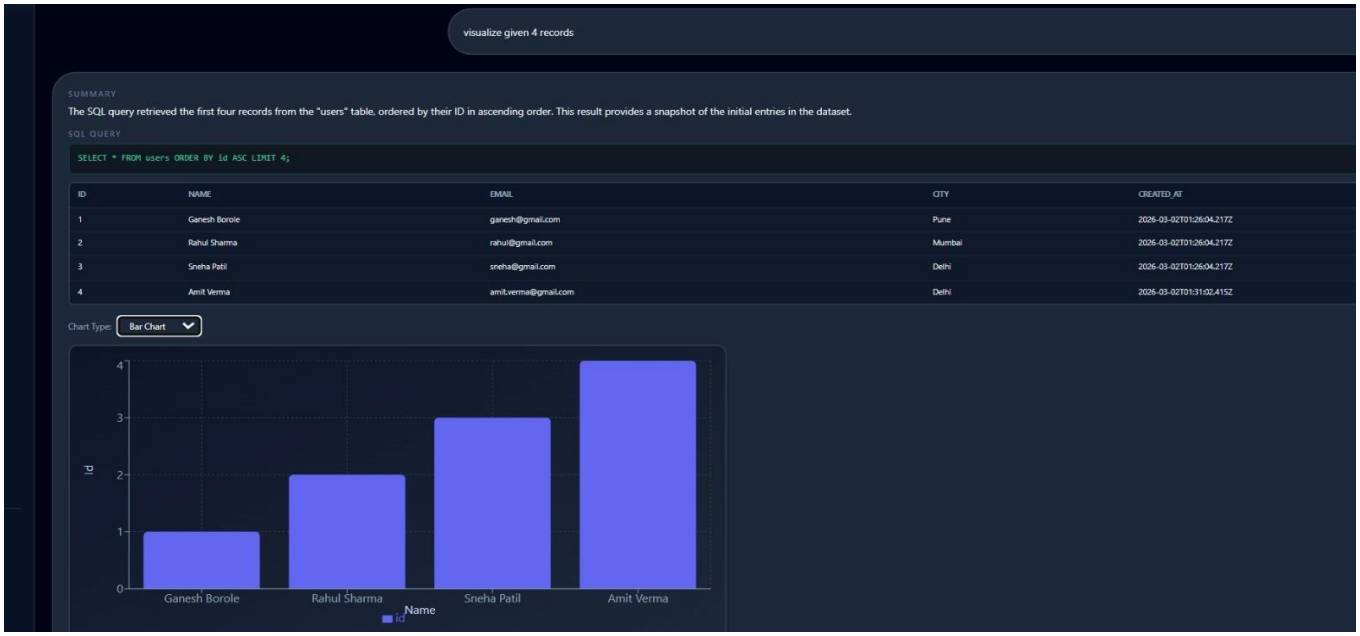
The system ensures high accuracy by validating queries before execution and refining them using AI. It reduces syntax and logical errors, providing reliable outputs.

Outputs of the System

Fig 2. Autofix-Assistant Application UI



ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026



DISCUSSION

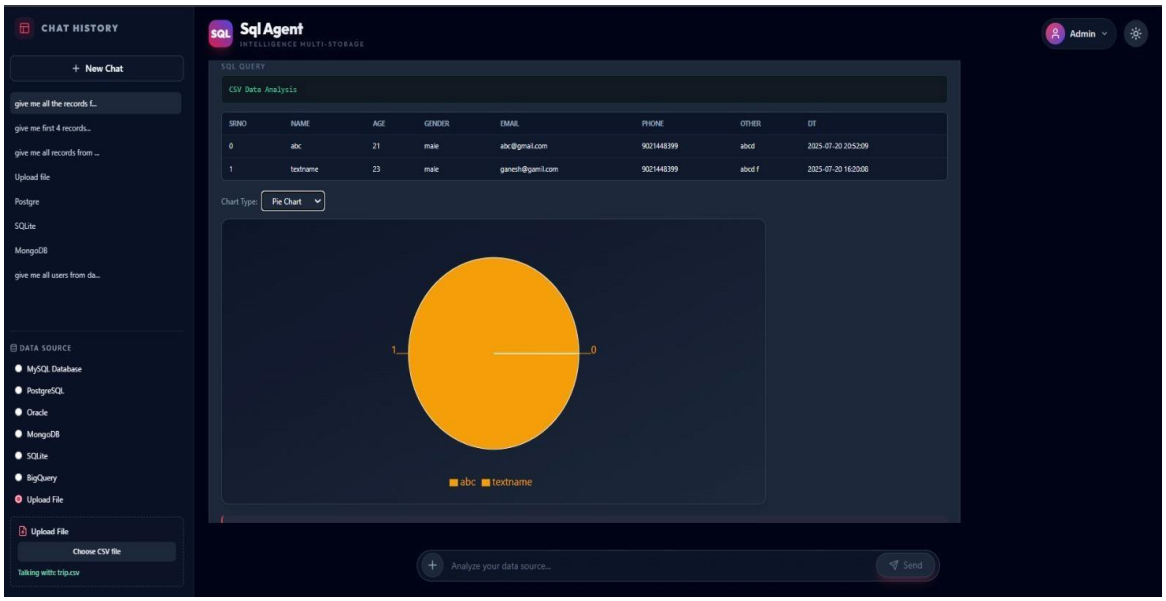


Fig 4. Data Visualization

guarantees proper interactions among various system components, and the AI processor optimizes the query results. On the other hand, the use In summary, the suggested approach of implementing the AI- enabled multi-database chatbot represents a considerable enhancement compared to conventional methods of working with databases. The use of natural language processing in combination with the possibility of executing database queries creates an environment that makes it much easier for users to manage their data without having to be familiar with SQL commands. Query validation and auto-fixing are some additional improvements that help to ensure the accuracy of the executed operations.

The test results reveal that the software is efficient for use in live environments due to its ability to give precise

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

output queries.

Furthermore, the software supports several databases including MySQL, MongoDB, and BigQuery. The multi-layered system of a chat interface makes the interaction more interesting for the users. However, the difficulties with handling complicated queries and dependence on the precision of AI models are still there.

In conclusion, the proposed system has emphasized the significance of AI in database management and highlighted the capability of AI to link up with both tech-savvy and non-tech-savvy users. The combination of AI technology with MERN stack is an example of what makes the system applicable to the real world. In future, areas that could be improved include semantic knowledge, voice commands, and scalability.

CONCLUSION

The suggested chat system based on artificial intelligence that utilizes multiple databases has been able to overcome the drawbacks of conventional database querying systems by allowing queries to be made through natural language. With the use of AI and MERN stack, the process of creating queries has become much simpler, there is reduced reliance on experts, and there are no mistakes when executing SQL commands.

The capability of the system to facilitate the creation of different databases, like MySQL, MongoDB, SQLite, and BigQuery, makes the system versatile and flexible, making it applicable in practical applications. The chat interface is another feature that ensures a user-friendly system, as it provides an interactive interface for data retrieval. Based on the evaluations, the system operates efficiently, producing accurate outputs while minimizing the complexity and processing time of queries.

Future Work

However, the above-discussed system represents a good basis for AI- powered database interfacing operations. Nonetheless, a number of improvements may be done in order to increase the efficiency of such a solution even more. Thus, in the future, one can implement voice-based query input in the system, which will allow a user to use a spoken language for querying. Improving the understanding of more complicated and nested queries can also help boost the accuracy of the process.

Furthermore, introducing more sophisticated AI models and fine- tuning existing datasets is also a potential step for enhancing the performance of the system. It may be helpful to develop certain visualization features for displaying results. Finally, introducing role-based access control mechanisms will help provide safe interaction with the database.

REFERENCES

1. Z. Zhong, C. Xiong, and D. Yu, ”Seq2SQL: Generat ing Structured Queries from Natural Language Using Reinforcement Learning,” in Proc. ACM SIGMOD, pp. 1913–1927, 2017.
2. Yu, Z. Zhang, and D. Yu, ”Spider: A Large Scale Human- Labeled Dataset for Complex and Cross Domain Semantic Parsing and Text-to-SQL Task,” in Proc. EMNLP, pp. 3911–3921, 2018
3. Wang, X. Liu, and M. Richardson, ”RAT-SQL: Relation-Aware Schema Encoding and Linking for Text to- SQL Parsers,” in Proc. ACL, pp. 7567–7578, 2020. OpenAI, ”GPT-4 Technical Report,” arXiv preprint arXiv:2303.08774, 2023.
4. Z. Hong, J. Xu, and Y. Chen, ”MetaGPT: Meta Pro gramming for Multi-Agent Collaboration,” arXiv preprint arXiv:2308.00352, 2023.
5. S. Sergeyuk, L. Mironova, and A. Pashkov, ”Using AI Based Coding Assistants in Practice: A Large-Scale Sur vey,” Journal of Software Engineering Research, vol. 12, no. 3,

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

6. pp. 215–228, 2024.
7. J. Lee, K. Park, and S. Choi, ”FixAgent: A Multi-Agent Framework for Automated Debugging and Code Repair,” in Proc. IEEE Int. Conf. on Artificial Intelligence and Data Science (AIDAS), pp. 88–95, 2024.
8. L. Wang, Y. Qian, and D. Zhou, ”MAC-SQL: Multi-Agent Collaboration for Robust Text-to-SQL Parsing,” in Proc. AAAI Conf. on Artificial Intelligence, pp. 12765–12773, 2024.
9. H. Gao, Y. Sun, and X. Li, ”Enhancing Semantic Con sistency in Text-to-SQL Generation with Context-Aware Reasoning,” in Proc. NAACL, pp. 6432–6445, 2024.
10. H. Gao, Y. Sun, and X. Li, ”Enhancing Semantic Con sistency in Text-to-SQL Generation with Context-Aware Reasoning,” in Proc. NAACL, pp. 6432–6445, 2024.
11. Y. Cen, K. Liu, and H. Li, ”SQLFixAgent: Towards Semantic- Accurate Text-to-SQL Parsing via Consistency Enhanced Multi-Agent Framework,” arXiv preprint arXiv:2503.01234, 2025.
12. OpenAI, “Codex: Evaluating Large Language Models for Code Generation,” OpenAI Research Publication, 2021.
13. J. Austin, A. Odena, M. Nye, et al., “Program Synthesis with Large Language Models,” arXiv preprint arXiv:2108.07732, 2021.
14. Y. Sun, T. Gao, and H. Li, “Improving Semantic Accuracy in Text-to-SQL Systems Using Reflective Query Validation,” in Proc. International Conference on Computational Linguistics (COLING), pp. 4521–4533, 2023.
15. AutoGPT Team, “AutoGPT: An Autonomous GPT-4 Experiment for Complex Task Solving,” GitHub Repository, 2023. [Online]. Available: https://github.com/Significant-Gravitas/AutoGPT
16. JetBrains, “JetBrains AI Assistant: Intelligent Coding Support for Developers,” JetBrains Official Documentation, 2024. [Online]. Available: https://www.jetbrains.com/ai/
17. Tabnine, “Tabnine AI Code Assistant for Software Development,” Tabnine Official Documentation, 2024. [Online]. Available: https://www.tabnine.com/
18. Microsoft, “Azure OpenAI Service for Intelligent Database Query Assistance,” Microsoft Documentation, 2024. [Online].Available: https://learn.microsoft.com/
19. Google Cloud, “AI-Powered Database Query Optimization Using Generative AI,” Google Cloud Research, 2024.