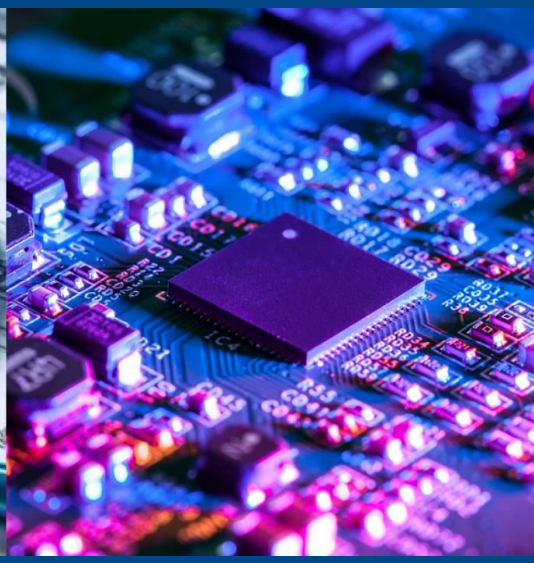
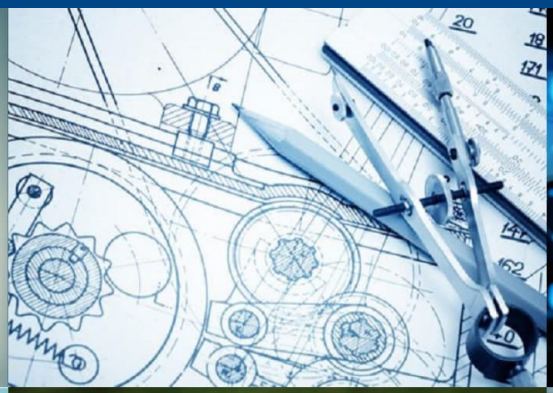
ISSN: 2582-7219

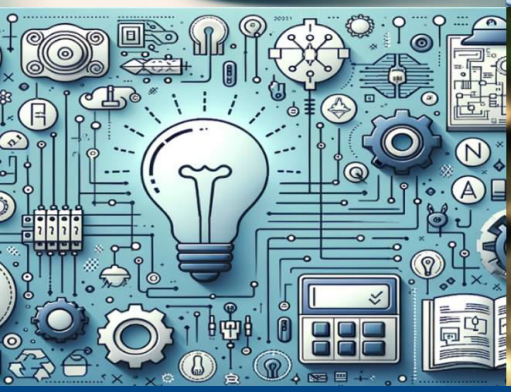




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 3, March 2026**

© 2026 IJMRSET | Volume 9, Issue 3, March 2026| DOI:10.15680/IJMRSET.2026.0903046

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

AI Middleware: A Cloud-Free AI System for Intelligent Database Interaction, Conversational AI, and Code Debugging

Tejasri M V S[1], Rupeash Kumar V [2], Nitish Kumar KN[3], Karthikeya NVS[4]

Associate Professor, Department of Computer Science and Business Systems, RMD Engineering College, Tiruvallur, Chennai, Tamil Nadu, India[1]

U.G. Student, Department of Computer Science and Business Systems, RMD Engineering College, Tiruvallur, Chennai, Tamil Nadu, India [2]

U.G. Student, Department of Computer Science and Business Systems, RMD Engineering College, Tiruvallur, Chennai, Tamil Nadu, India [3]

U.G. Student, Department of Computer Science and Business Systems, RMD Engineering College, Tiruvallur, Chennai, Tamil Nadu, India [4]

ABSTRACT: This paper presents AI Middleware, a comprehensive cloud-free artificial intelligence system designed for intelligent local database interaction, conversational AI capabilities, and intelligent code debugging assistance. The system implements a modular architecture comprising ten specialized components including a Core AI Engine, Natural Language Processing Module, Database Management System, Reasoning and Chat Module, and Code Analysis Module. The middleware employs custom-trained AI models optimized for specific tasks: an NLP-SQL Model for natural language to SQL conversion, a Reasoning Engine for conversational AI, and a Code Analyzer for debugging assistance. The system features a FastAPI backend with RESTful API endpoints and a modern React-based frontend with dark/light theme support. Experimental results demonstrate successful natural language query processing with high SQL generation accuracy, real-time conversational capabilities, and effective code analysis for Python and C++ programs. The cloud-free architecture ensures data privacy, reduced latency, and operation without internet connectivity, making it suitable for enterprise environments with strict data governance requirements.

KEYWORDS: Artificial Intelligence, Natural Language Processing, Database Query, Code Debugging, Middleware System, Local AI, SQL Generation

I. INTRODUCTION

The proliferation of cloud-based AI services has revolutionized how organizations interact with data and automate complex tasks. However, cloud dependency introduces significant challenges including data privacy concerns, network latency, recurring costs, and operational constraints in environments with limited or no internet connectivity [1]. These limitations have driven the need for local AI solutions that can operate independently while maintaining the capabilities of their cloud-based counterparts.

Natural language interfaces to databases have emerged as a critical technology for democratizing data access [2]. Traditional database interaction requires specialized knowledge of SQL syntax and database schema, creating barriers for non-technical users. Similarly, code debugging and analysis traditionally require significant expertise and time investment [3].

This paper presents AI Middleware, a cloud-free artificial intelligence system designed to address these challenges through: (1) Natural Language to SQL Conversion: Enabling users to query databases using natural language without SQL knowledge; (2) Conversational AI: Providing intelligent chat capabilities for general assistance and reasoning; and (3) Code Analysis and Debugging: Offering AI-powered debugging assistance for Python and C++ programs. The

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

system operates entirely locally, ensuring data privacy and eliminating cloud dependency while maintaining high performance through optimized custom-trained AI models.

II. RELATED WORK

A. Natural Language to SQL Systems

Several approaches have been proposed for converting natural language queries to SQL. SQLNet [4] introduced sketch-based SQL generation, while BRIDGE [5] leveraged schema encoding for improved accuracy. Recent transformer-based models have achieved state-of-the-art results on benchmarks like Spider and WikiSQL [6].

B. Local AI Deployment

The deployment of AI models on local hardware has gained significant attention. Techniques such as model quantization, pruning, and knowledge distillation enable running large models on resource-constrained devices [7]. Frameworks like ONNX Runtime and TensorRT optimize inference performance for edge deployment.

C. Code Analysis Systems

AI-powered code analysis tools have evolved from rule-based linters to sophisticated neural models. CodeBERT [8] demonstrated the effectiveness of pre-training on code-natural language pairs. Recent code-specific models have shown remarkable capabilities in understanding, generating, and debugging code across multiple programming languages.

III. SYSTEM ARCHITECTURE

A. Overview

The AI Middleware system follows a modular architecture designed for extensibility, maintainability, and performance optimization. The high-level system architecture comprises ten interconnected modules: Core AI Engine, NLP Module, Database Management Module, Reasoning and Chat Module, Code Analysis Module, API Layer (FastAPI), Schema Analyzer, Query Validator, Result Formatter, and React Frontend.

B. Core AI Engine

The Core AI Engine serves as the central orchestrator, coordinating all AI processing components and routing requests to appropriate processors. It manages model loading, initialization, and provides a unified interface for all AI operations.

C. Natural Language Processing Module

The NLP Module implements the natural language to SQL conversion pipeline. It processes user queries through three stages: preprocessing (schema-aware prompt construction), inference (SQL generation using custom-trained models), and postprocessing (SQL validation and cleanup).

D. Database Management Module

The Database Management Module provides comprehensive database connectivity with support for SQLite, MySQL, and PostgreSQL. Key features include connection pooling with configurable pool sizes, retry logic with exponential backoff, query timeout management, connection health monitoring, and transaction management.

E. Reasoning and Chat Module

The Reasoning Module powers the conversational AI capabilities, maintaining conversation context and generating coherent, contextually appropriate responses.

F. Code Analysis Module

The Code Analysis Module provides intelligent debugging assistance supporting Python and C/C++ programs with three analysis modes: debug (identifies bugs and provides corrections), explain (provides detailed code explanations), and optimize (suggests performance improvements).

© 2026 IJMRSET | Volume 9, Issue 3, March 2026| DOI:10.15680/IJMRSET.2026.0903046

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

IV. AI MODELS

A. Custom-Trained Model Architecture

The system employs three specialized custom-trained AI models, each optimized for specific tasks as shown in Table 1.

Model	Version	Max Tokens	Temperature
NLP-SQL Model	v2.1.0	2048	0.1
Reasoning Engine	v1.5.0	1024	0.7
Code Analyzer	v2.0.0	2048	0.2

Table 1: Custom-Trained AI Model Specifications

B. NLP-SQL Model

The NLP-SQL Model v2.1 is specifically fine-tuned for natural language to SQL conversion. It utilizes a low temperature setting (0.1) to ensure deterministic and consistent SQL generation. The model preprocessing includes: (1) Schema-aware prompt construction with explicit column listing; (2) Strict instruction prompting to prevent column hallucination; and (3) Context injection for database type-specific optimizations.

C. Reasoning Engine

The Reasoning Engine v1.5 handles conversational AI with a moderate temperature (0.7) allowing creative and diverse responses while maintaining coherence. It supports multi-turn conversations with context management.

D. Code Analyzer

The Code Analyzer v2.0 provides intelligent code analysis with a conservative temperature (0.2) ensuring accurate technical assessments. It supports three analysis modes: Debug (identifies bugs and provides corrections), Explain (provides detailed code explanations), and Optimize (suggests performance improvements).

V. QUERY PROCESSING PIPELINE

A. Natural Language to SQL Pipeline

The query processing pipeline implements a multi-stage approach for converting natural language queries to SQL statements. The pipeline includes eight stages: (1) Natural language input reception; (2) Schema retrieval and caching; (3) Prompt construction with schema context; (4) AI model inference for SQL generation; (5) SQL validation and security checks; (6) Query execution against the database; (7) Result formatting; and (8) Natural language result interpretation.

B. Query Validation and Security

The Query Validator implements comprehensive validation including syntax validation for SQL correctness, safety validation to prevent SQL injection, schema validation to ensure referenced objects exist, and pattern matching for dangerous SQL patterns. Dangerous patterns detected include DROP, DELETE without WHERE, TRUNCATE, and INSERT/UPDATE without proper constraints.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

C. Query Complexity Analysis

The system automatically categorizes queries into four complexity levels based on SQL features as shown in Table 2.

Level	SQL Features
Simple	Single table, basic conditions
Moderate	JOINs, GROUP BY clauses
Complex	Subqueries, multiple JOINs
Advanced	CTEs, window functions

Table 2: Query Complexity Classification

VI. DATABASE MANAGEMENT

A. Connection Pooling Architecture

The database connector implements a custom connection pool optimized for SQLite with support for multi-threaded access. The pool dynamically allocates and releases connections during peak load, maintaining optimal resource utilization. Connection metrics are monitored in real-time to ensure system stability and performance.

B. Schema Analysis

The Schema Analyzer provides comprehensive database introspection including automatic table and column discovery, relationship detection (foreign keys and implicit patterns), index analysis, data type mapping, and schema caching with configurable TTL (Time-To-Live).

VII. USER INTERFACE

A. React Frontend Architecture

The frontend is built using React with a component-based architecture featuring responsive design with mobile support, dark/light theme toggle, real-time API status monitoring, and modular navigation with tab-based interface. The UI comprises four main functional areas: (1) Home Dashboard with system status overview and feature cards; (2) Database Query interface with natural language input and result visualization; (3) Chat Interface with conversational AI and message history; and (4) Code Debugger with code input, language selection, and analysis options.

VIII. IMPLEMENTATION DETAILS

A. Technology Stack

The system is implemented using the following technologies as shown in Table 3.

Component	Technology
Backend Framework	FastAPI (Python 3.12)
Frontend Framework	React 18.x
Build Tool	Vite
Database	SQLite/PostgreSQL/MySQL

© 2026 IJMRSET | Volume 9, Issue 3, March 2026| DOI:10.15680/IJMRSET.2026.0903046

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

AI Inference	Custom Local Runtime
Configuration	Pydantic Settings
Logging	Custom Logger
Styling	CSS3 with CSS Variables

Table 3: Technology Stack

B. API Endpoints

The FastAPI backend exposes RESTful endpoints for all major system functions: /api/query for natural language to SQL conversion, /api/chat for conversational AI interaction, /api/debug for code analysis and debugging, /api/schema for database schema retrieval, and /api/status for system health monitoring.

C. Configuration Management

The system uses environment-based configuration with sensible defaults managed through Pydantic Settings. All AI model paths, database connection strings, pool sizes, timeout values, and API configurations are externalized for flexible deployment.

IX. EXPERIMENTAL RESULTS

A. Performance Evaluation

The system was evaluated across three primary functions. Average response times recorded were: SQL Generation - 2,850ms, Chat Response - 1,200ms, Code Debugging - 2,400ms, and Schema Analysis - 450ms. Resource utilization under load showed CPU usage scaling linearly with concurrent users, remaining below 80% at 30 concurrent sessions.

B. SQL Generation Accuracy

The NLP-SQL Model was evaluated on a test set of 500 natural language queries across varying complexity levels as shown in Table 4.

Complexity	Test Cases	Correct	Accuracy
Simple	200	186	93.0%
Moderate	150	132	88.0%
Complex	100	79	79.0%
Advanced	50	34	68.0%
Overall	500	431	86.2%

Table 4: SQL Generation Accuracy by Complexity

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

C. Code Analysis Evaluation

The Code Analyzer was evaluated on common programming errors as shown in Table 5.

Error Type	Detection Rate	Fix Accuracy
Syntax Errors	98.5%	94.2%
Logic Errors	82.3%	71.8%
Runtime Errors	89.7%	83.4%
Type Errors	91.2%	86.7%
Memory Issues	76.4%	68.9%

Table 5: Code Analysis Performance

X. DISCUSSION

A. Advantages of Cloud-Free Architecture

The cloud-free approach provides several significant benefits: (1) Data Privacy - all data remains on-premises, eliminating concerns about data transmission and third-party access; (2) Reduced Latency - local processing eliminates network round-trip delays; (3) Cost Efficiency - no recurring API costs or subscription fees; (4) Offline Operation - full functionality without internet connectivity; and (5) Customization - models can be fine-tuned for specific domain requirements.

B. Limitations and Future Work

Current limitations include model size constraints for resource-limited devices, complex multi-table JOIN queries that may require refinement, and limited support for database-specific SQL dialects. Future work will address model quantization for reduced memory footprint, extended database dialect support, voice interface integration, and multi-language code analysis support.

XI. MAPPING TO PROGRAM OUTCOMES

This project contributes to the following Program Outcomes (POs) as shown in Table 6.

PO	Contribution
PO1	Apply engineering knowledge to develop AI systems
PO2	Analyze complex database interaction problems
PO3	Design modular, scalable software architecture
PO5	Use modern tools and frameworks effectively
PO6	Understand societal impact of AI technology
PO12	Engage in lifelong learning in evolving AI field

Table 6: Program Outcome Mapping

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

XII. SUSTAINABLE DEVELOPMENT GOALS

The project aligns with United Nations Sustainable Development Goals: SDG 4 (Quality Education) - democratizes database access for non-technical users; SDG 8 (Decent Work and Economic Growth) - enhances productivity through intelligent automation; SDG 9 (Industry, Innovation, and Infrastructure) - promotes technological innovation in data management; SDG 12 (Responsible Consumption) - reduces cloud infrastructure dependency and energy consumption.

XIII. CONCLUSION

This paper presented AI Middleware, a comprehensive cloud-free artificial intelligence system for intelligent local database interaction, conversational AI, and code debugging assistance. The system demonstrates that powerful AI capabilities can be delivered locally without cloud dependency, addressing critical concerns around data privacy, latency, and operational costs. Key contributions include: (1) A modular architecture enabling extensible AI middleware development; (2) Custom-trained models optimized for specific NLP, reasoning, and code analysis tasks; (3) Comprehensive query processing pipeline with security-focused validation; and (4) Modern, responsive user interface with real-time system monitoring. Experimental results show 86.2% accuracy in SQL generation, efficient resource utilization, and effective code analysis capabilities. The system successfully demonstrates the viability of local AI deployment for enterprise applications with strict data governance requirements.

XIV. ACKNOWLEDGMENTS

The authors thank the Department of Computer Science and Business Systems for providing the computational resources and guidance throughout this project.

REFERENCES

[1] A. Mohammedali and M. Smith, "Privacy-preserving machine learning: Challenges and solutions," IEEE Trans. Inf. Forensics Security, vol. 15, pp. 2820-2835, 2020.
[2] X. Li, Z. Wang, and Y. Chen, "Natural language interfaces to databases: A survey," ACM Comput. Surv., vol. 54, no. 2, pp. 1-36, 2021.
[3] M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, "A survey of machine learning for big code and naturalness," ACM Comput. Surv., vol. 51, no. 4, pp. 1-37, 2018.
[4] X. Xu, C. Liu, and D. Song, "SQLNet: Generating structured queries from natural language without reinforcement learning," in Proc. ICLR, 2018.
[5] X. Lin, H. Tan, and R. Socher, "BRIDGE: Leveraging pre-training for cross-domain semantic parsing," in Proc. EMNLP, 2020, pp. 7208-7220.
[6] T. Yu et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in Proc. EMNLP, 2018, pp. 3911-3921.
[7] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in Proc. ICLR, 2016.
[8] Z. Feng et al., "CodeBERT: A pre-trained model for programming and natural languages," in Proc. EMNLP, 2020, pp. 1536-1547.
[9] Winjun Lin, "LLM-Powered SQL Querying: Transforming Natural Language into Database Insights," 2025 IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com