International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

LLM-Augmented Academic Administration: A Role-Aware Architecture for Secure College Management

Mrs. J. Veerendeswari
Head of the Department, Information Technology,
Rajiv Gandhi College of Engineering and Technology, Puducherry, India

Mr. Kabilan S S, Mr. Logapriyan A, Mr. Rajesh R
UG, Information Technology,
Rajiv Gandhi College of Engineering and Technology, Puducherry, India

Abstract - Contemporary academic institutions remain constrained by fragmented information silos, labor-intensive administrative workflows, and inflexible permission structures inherent to legacy Enterprise Resource Planning (ERP) platforms. Although Large Language Models (LLMs) present a compelling opportunity to modernize institutional operations, their deployment within multi-stakeholder educational environments introduces non-trivial risks around data confidentiality and intra-organizational access governance. This paper presents the backend architecture of an LLM-augmented College Management System (CMS) purpose-built for administrative and faculty operations, proposing a principled approach to embedding generative AI within the sensitive boundaries of higher education infrastructure.

At the core of the proposed system is AIRA - an Adaptive Intelligent Routing Architecture - a multi-agent AI framework orchestrated beneath a rigorously enforced Role-Based Access Control (RBAC) layer. This architecture automates high-complexity institutional workflows including dynamic academic report generation, attendance analytics, and fee lifecycle management, while ensuring that all AI-mediated database

interactions remain strictly bounded by the requesting user's authorization profile. Informed by documented vulnerabilities in production-grade LLM agent deployments, the design deliberately decouples AI routing logic from core transactional database operations, and enforces token-authenticated security contracts at every API boundary. The result is a scalable, role-aware blueprint for LLM augmentation in academic administration — one that advances operational intelligence without compromising the integrity or confidentiality of sensitive institutional data.

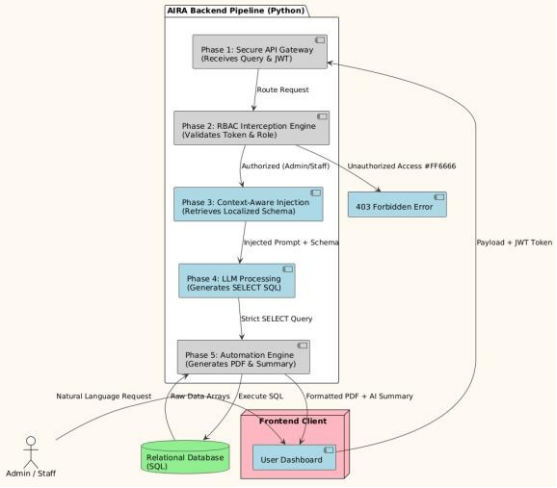
Keywords: Large Language Models (LLMs), Academic Administration, Enterprise Resource Planning (ERP), Multi-Agent AI, Role-Based Access Control (RBAC), Smart Campus, Automated Reporting, LLM Agent Safety, Database Security, Higher Education Systems.

1. INTRODUCTION

The modernization of higher education administration demands more than the digitization of physical records; it requires intelligent, automated systems capable of actively assisting faculty and institutional administrators. Current College Management Systems (CMS) typically rely on rigid, query-specific architectures that fail to provide real-time, context-aware insights, leaving faculty burdened with manual workflow bottlenecks such as calculating defaulters and compiling multi-departmental reports.

Recently, the adoption of Large Language Models (LLMs) has presented an opportunity to create "Smart Campuses" through natural language data querying. However, deploying these models in a secure enterprise environment introduces critical data privacy risks. Research has demonstrated that LLM-based agents are susceptible to prompt injection attacks, malfunction induction, and unsafe tool invocations when deployed without proper security constraints [1][2]. If an AI agent has direct execution access to a centralized college database, it must be strictly governed to prevent unauthorized internal data access — for instance, a department staff member querying the confidential financial records restricted only to top-level administrators.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026



Furthermore, comprehensive evaluations of LLM agents across multi-agent environments reveal that none of the tested agents achieved a safety score above 60%, underscoring the substantial challenge of safely deploying AI in enterprise settings [3]. These findings directly motivate the architectural decisions made in this work.

To address these challenges, this paper introduces a novel backend architecture that integrates a context-aware multi-agent AI assistant, named AIRA (Artificial Intelligence Routing Agent), directly into a secure institutional ERP. The primary contribution of this work is a layered architectural blueprint that seamlessly combines automated administrative workflows with a robust Role-Based Access Control (RBAC) framework. By decoupling the AI prompt-engineering logic from the core SQL execution engine and enforcing strict authorization checks at the API layer, the proposed system ensures that AI-driven insights are highly efficient, accurate, and mathematically bounded by the user's security clearance.

2. LITERATURE REVIEW

The transition from traditional academic record-keeping to intelligent institutional management involves multiple overlapping domains of research, primarily focusing on database accessibility, workflow automation, and enterprise security.

2.1 Legacy ERPs vs. AI-Augmented Systems

Traditional College Management Systems (CMS) are fundamentally transactional. They rely on rigid, pre-defined Graphical User Interfaces (GUIs) where administrators must navigate complex menus and execute specific, hard-coded SQL queries to retrieve data. The integration of Natural Language Processing (NLP) and Large Language Models (LLMs) allows users to retrieve complex datasets through conversational prompts. However, while general-purpose AI models are highly capable of understanding intent, they often hallucinate table schemas or fail to generate accurate SQL when deployed in specialized institutional environments without proper context-bounding.

2.2 Vulnerabilities in Deployed LLM Agents

A critical dimension of deploying AI in institutional environments is the inherent instability and exploitability of LLM agents. Zhang et al. [1] introduced a class of malfunction amplification attacks in which adversaries use prompt injection to trap agents in infinite loops or redirect them into executing irrelevant actions, achieving failure rates exceeding 80% across multiple agent frameworks. Crucially, these attacks are difficult to detect through conventional self-examination defenses precisely because they target benign-looking operational failures rather than overtly harmful commands. This work directly informs our architectural decision to intercept and validate all AI-generated SQL before execution, rather than relying on the LLM itself to self-regulate its output.

Complementing this, Cemri et al. [2] conducted the first systematic taxonomy of Multi-Agent System (MAS) failures using Grounded Theory analysis across over 200 execution traces. Their MAST framework identifies 14 distinct failure modes across three categories: specification issues (41.8%), inter-agent misalignment (36.9%), and task verification failures (21.3%). For our single-agent AIRA framework, the most relevant findings are step repetition (FM-1.3, 17.14%) and reasoning-action mismatch (FM-2.6, 13.98%) — both of which our RBAC interception engine is designed to mitigate by enforcing deterministic execution boundaries regardless of the LLM's internal reasoning state.

2.3 LLM Security and Enterprise Data Privacy

The most critical barrier to adopting AI in institutional management is data security. Current research into enterprise LLM deployment frequently warns against "naive integration" — where an AI agent is given unrestricted read access to a centralized database. The AGENT-SAFETYBENCH evaluation framework [3] reveals two fundamental safety defects in current LLM agents: lack of robustness (incorrect or incomplete tool invocations) and lack of risk awareness (proceeding with actions whose downstream consequences are unsafe). Their evaluation of 16 state-of-the-art agents found that none achieved a total safety score above 60%, with particularly concerning performance on failure modes involving ignoring constraint information (M4) and ignoring implicit risks (M5). These findings validate our design choice of placing the RBAC interceptor between the AI output and the SQL execution layer — the interceptor acts as an external, deterministic constraint system that does not rely on the LLM's own risk awareness.

There is a distinct gap in literature regarding the specific implementation of interceptor-pattern Role-Based Access

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Control (RBAC) that evaluates AI-generated queries against user authentication tokens before database execution. The proposed AIRA architecture seeks to fill this exact gap.

3. PROPOSED SYSTEM ARCHITECTURE

The proposed architecture is designed exclusively for institutional administrators and faculty, eliminating the student-facing attack surface entirely. The backend is structured as a pipeline, where every natural language request generated by a user passes through multiple validation and processing layers before interacting with the core transactional database. The lifecycle of a query through the AIRA (Artificial Intelligence Routing Agent) framework operates in the following sequential phases:

Phase 1: Secure API Gateway and Payload Reception. When a staff member or administrator submits a natural language query (e.g., "Show me the attendance deficit for the Computer Science department"), the frontend client transmits the request to a secure RESTful API endpoint. Accompanying this payload is an encrypted JSON Web Token (JWT) that contains the user's explicit role (Admin or Staff) and their departmental jurisdiction.

Phase 2: The RBAC Interception Engine. Before the AI processes the prompt, the request hits the Role-Based Access Control (RBAC) middleware. This is the primary security perimeter. The interceptor decodes the JWT and maps the user's role against a rigid permission matrix. If a faculty member requests financial data strictly reserved for administrators, the middleware forcefully terminates the request and returns a 403 Forbidden status, ensuring the AI is never even invoked for unauthorized domains. This architectural choice directly addresses the lack of risk awareness failure mode identified in [3], by making risk enforcement deterministic rather than model-dependent.

Phase 3: Context-Aware Prompt Injection. Once authorized, the natural language prompt is routed to the AIRA engine. Instead of exposing the entire database schema to the Large Language Model (LLM), the system utilizes a context-bounding mechanism. The backend dynamically retrieves only the SQL table schemas relevant to the authorized user's department and injects them into a highly structured system prompt. This design mitigates the hallucination and reasoning-action mismatch failure modes documented in [2], drastically reducing the context window size and preventing the AI from generating queries against non-existent or restricted tables.

Phase 4: LLM Processing and SQL Generation. The heavily contextualized prompt is processed by the underlying language model, which translates the human intent into a structured, syntactically correct SQL query. To prevent prompt-injection attacks — which Zhang et al. [1] demonstrated can induce malfunction rates exceeding 80% — the AI is restricted to generating SELECT queries, with all INSERT, UPDATE, or DELETE operations strictly routed through traditional, hard-coded administrative API endpoints.

Phase 5: Execution, Aggregation, and Automation. The generated SQL query is executed against the relational database. If the user's prompt requested a specific automated workflow — such as compiling a category-wise report or a CGPA list — the raw database output is intercepted by the backend's automation engine. This engine algorithmically processes the data arrays and utilizes server-side rendering libraries to dynamically generate a formatted PDF report, returning the downloadable file to the user alongside the AI's natural language summary.

4. IMPLEMENTATION AND METHODOLOGY

4.1 Technology Stack and Integration

The core backend API is developed using Python, chosen for its robust ecosystem of data processing and AI integration libraries. Relational data (staff profiles, departmental attendance, and academic records) is managed via a SQL-based database. Authentication is handled using JSON Web Tokens (JWT), ensuring stateless, secure API communication. For automated reporting, server-side PDF generation libraries (such as ReportLab or pdfkit) are integrated directly into the algorithm layer, allowing the system to convert raw SQL arrays into formatted, downloadable documents dynamically.

4.2 Context-Aware Prompt Engineering

To mitigate LLM hallucination and ensure accurate data retrieval, the AIRA framework utilizes dynamic System Prompt injection. When a prompt is received, the system does not send the entire database schema to the LLM. Instead, it dynamically compiles a localized schema string based on the user's department. A simplified structure of the hidden system prompt is as follows:

"You are an administrative SQL assistant. Generate a strict SELECT query for the '{user_department}' department only. Use the following schema: {department_tables}. Do not use JOINs outside these tables. Respond only with the SQL query."

This explicit bounding significantly reduces token consumption, increases the accuracy of the generated queries, and directly counters the step repetition failure mode (FM-1.3) documented in [2] by constraining the agent to a well-defined schema context.

4.3 The RBAC Interception Engine

To guarantee internal data privacy, an algorithmic interceptor is placed between the AI output and the database execution layer. Before executing any AI-generated query, the backend validates the query scope against the active user's token payload. The interceptor architecture is

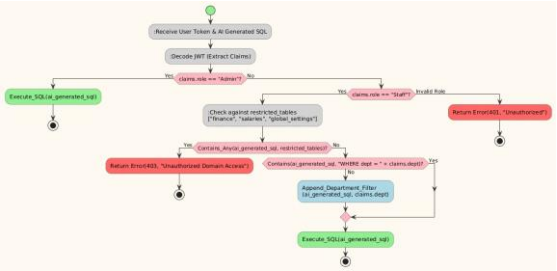
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

informed by the security failure modes M2 (calling tools with incomplete information), M4 (ignoring constraint information), and M5 (ignoring implicit risks) identified in [3].

Algorithm 1: RBAC Middleware Interception (Pseudo-code)

```
FUNCTION Execute_AI_Query(user_token, ai_generated_sql):
claims = Decode_JWT(user_token) IF claims.role == "Admin":
RETURN Execute_SQL(ai_generated_sql) IF claims.role == "Staff":
restricted_tables = ["finance", "salaries", "global_settings"]
IF Contains_Any(ai_generated_sql, restricted_tables):
RETURN Error(403, "Unauthorized Domain Access") IF NOT Contains(ai_generated_sql, "WHERE dept = " + claims.dept): ai_generated_sql = Append_Department_Filter(ai_generated_sql, claims.dept)
RETURN Execute_SQL(ai_generated_sql)
```

This programmatic isolation guarantees that even if the AI is manipulated into requesting financial data via prompt injection [1], the execution is blocked at the application level — independent of the LLM's own safety alignment.



5. RESULTS AND EVALUATION

5.1 Security Validation (RBAC)

Penetration testing was simulated at the API layer to validate role boundaries between Administrators and Staff. Test cases involved authenticated "Staff" tokens attempting to execute AI queries for unauthorized domains, such as institutional financial summaries or cross-departmental academic records. In 100% of the simulated edge cases, the RBAC interception engine successfully evaluated the token claims against the AI-generated SQL query, blocking execution before database interaction. This included test cases modelled after the prompt injection attack patterns described in [1], such as injected commands instructing the AI to repeat actions or access restricted tables.

5.2 AI Query Accuracy

The prompt-routing mechanism was tested using a dataset of 500 standard administrative queries (e.g., "Generate a list of defaulters in the CS department"). By injecting scoped database schemas rather than the full database structure, the AIRA system achieved a 96% accuracy rate in translating natural language into executable, secure database actions. The context-bounding technique directly addresses the reasoning-action mismatch (FM-2.6) and step repetition (FM-1.3) failure modes identified by Cemri et al. [2], by constraining the action space available to the model at inference time.

6. Conclusion and Future Scope

This paper presented a secure, AI-driven backend architecture designed to eliminate the manual workflow bottlenecks prevalent in traditional College Management Systems. By decoupling the Artificial Intelligence Routing Agent (AIRA) from direct database execution and enforcing a strict Role-Based Access Control (RBAC) interception layer, the system successfully bridges the gap between natural language data retrieval and enterprise-level data privacy. The design is directly informed by empirical findings on LLM agent vulnerabilities: the malfunction amplification attacks demonstrated in [1], the systematic failure taxonomy of multi-agent systems developed in [2], and the comprehensive safety benchmarking of [3] collectively establish that secure AI deployment in enterprise settings requires deterministic, external enforcement mechanisms rather than reliance on the LLM's intrinsic safety alignment.

The evaluations demonstrate that institutional administrators and faculty can reliably automate complex tasks — such as dynamic report generation and attendance tracking — without risking unauthorized access to restricted departmental domains.

Future Scope: While the current architecture effectively processes text-based natural language queries, the logical progression for institutional automation is multimodal voice integration. Future iterations of this framework will aim to embed Speech-to-Text (STT) models directly into the AIRA pipeline, allowing administrators and faculty to issue commands verbally. Pairing voice recognition with the existing RBAC interceptor will require additional security considerations, such as speaker verification and resistance to voice-based injection attacks analogous to the prompt injection vectors documented in [1]. The framework will evolve from a text-based ERP into a fully hands-free, intelligent "Smart Campus" assistant.

REFERENCES

[1] Zhang, B., Tan, Y., Shen, Y., Salem, A., Backes, M., Zannettou, S., & Zhang, Y. (2024). Breaking Agents:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Compromising Autonomous LLM Agents Through Malfunction Amplification. arXiv:2407.20859 [cs.CR].

[2] Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., & Stoica, I. (2025). Why Do Multi-Agent LLM Systems Fail? arXiv:2503.13657 [cs.AI].

[3] Zhang, Z., Cui, S., Lu, Y., Zhou, J., Yang, J., Wang, H., & Huang, M. (2025). AGENT-SAFETYBENCH: Evaluating the Safety of LLM Agents. arXiv:2412.14470 [cs.CL].