

AI Based Student Connect Portal

Jaydeep Chaudhari

Department of Computer Engineering
(Student)

Sinhgad College of Engineering
(SPPU)

Pune, India

jaydeepchaudhari101@gmail.com

Pushkar Chaudhary

Department of Computer Engineering
(Student)

Sinhgad College of Engineering
(SPPU)

Pune, India

pushkarchoudhary362@gmail.com

Om Shinde

Department of Computer Engineering
(Student)

Sinhgad College of Engineering
(SPPU)

Pune, India

omshinde1442003@gmail.com

Nikhil Deuskar

Department of Computer Engineering
(Student)

Sinhgad College of Engineering
(SPPU)

Pune, India

nikdeuskar06@gmail.com

Ajit Karanjkar

Department of Computer Engineering
(Assistant Professor)

Sinhgad College of Engineering
(SPPU)

Pune, India

amkaranjkar.scoe@sinhgad.edu

Abstract– *The AI-Based Student Connect Portal is an intelligent, multi-user communication platform designed to bridge the gap between faculty, students, and parents through artificial intelligence. The system automates institutional tasks such as notice summarization, categorization, and prioritization to ensure that important information reaches the right audience efficiently. Using LangChain4j and Gemini AI, the platform enables faculty members to retrieve student performance data through natural language queries, eliminating the need for SQL knowledge and improving data accessibility. It also supports secure, permission-aware data retrieval using IAM-integrated RAG mechanisms to ensure privacy and role-based access. Students benefit from an AI chatbot that provides instant responses, summarized notices, and grievance tracking, while parents receive multilingual updates, performance summaries, and real-time alerts about their child's progress.*

Administrators can manage notices, respond to grievances, and monitor institutional activities through a unified dashboard. By combining AI-driven automation, secure data handling, and natural language interaction, this system enhances transparency, reduces manual workload, and promotes a smarter, more connected academic environment.

Keywords– AI-Based Portal, Prompt-to-SQL, RAG, IAM Security, Notice Summarization, Role-Based Access Control, LangChain4j, Gemini API

I. INTRODUCTION

Educational institutions have become deeply dependent on digital infrastructure — for academic communication, data management, and keeping stakeholders in the loop. But despite how far that shift has gone, significant gaps still exist between students, faculty, parents, and administrators. Information is slow to reach the right people. Grievances get handled behind closed doors. Academic data sits in systems that most people can't easily navigate. Printed notice boards, email chains, and fragmented portals just don't have the intelligence — or the integration — to keep up with a modern campus.

The AI-Based Student Connect Portal is an intelligent, multi-user communication platform built to bridge exactly those gaps. It uses artificial intelligence to connect faculty, students, and

parents in a way that older systems simply can't. The timing makes sense: Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), and Natural Language Processing (NLP) are now mature enough to automate institutional workflows at scale. The portal serves four types of users — students, who get summarized notices and can track their own grievances; faculty, who can query academic databases in plain English without needing to know SQL; parents, who receive multilingual performance summaries and real-time alerts; and administrators, who manage everything through one unified dashboard.

By combining AI-driven automation with secure data handling and natural language interaction, the system cuts manual workload, improves transparency, and creates a genuinely smarter academic environment. It runs on LangChain4j and Gemini AI for prompt processing and summarization, and it uses an IAM-based RAG mechanism to make sure each user only sees what they're supposed to see. The goal isn't just efficiency — it's trust. When students, and faculty all get the right information at the right time, institutions work better. And this portal is designed to be a scalable blueprint for exactly that.

II. LITERATURE SURVEY

The design of this portal draws on research across four areas: natural language database querying, retrieval-augmented generation, IAM-integrated access control, and LLM security in enterprise applications.

Natural Language to SQL (NL-to-SQL) systems allow non-technical users to query databases using plain English. Recent LLM-based models can handle complex queries with high accuracy, but they also introduce security risks such as Prompt-to-SQL injection attacks. Research by Pedro et al. (2025) proposed security mechanisms like query validation, filtering, and LLM guarding to prevent unauthorized access. These concepts are integrated into the portal's Prompt-to-SQL engine to ensure secure and permission-based data retrieval.

RAG improves the reliability of LLM outputs by grounding responses in real-time documents and reducing hallucinations. Studies by Gao et al. (2025) and Buchepalli et al. (2025) showed that hierarchical retrieval and reranking methods improve factual accuracy and summarization quality. These techniques are

applied in the portal's notice summarization module to generate concise and relevant updates.

Permission-aware retrieval is essential in multi-user academic systems. Jeong and Lee (2024) introduced an IAM-integrated RAG framework that filters data before it reaches the LLM, enabling secure and fine-grained access control with minimal performance impact. Additional studies also highlighted threats such as context leakage and prompt injection, which are addressed in the portal's architecture.

Existing campus platforms are often limited to isolated functions such as attendance or notice management and lack intelligent integration. Secure NL-to-SQL deployment, multilingual summarization, personalized dashboards, and IAM-aware retrieval are still uncommon in academic systems. The AI-Based Student Connect Portal is designed to overcome these limitations through a secure, AI-driven, and integrated platform.

III. PROBLEM STATEMENT

Academic institutions don't have a unified, intelligent platform for secure, role appropriate communication students, faculty, parents, between and administrators. What they do have is fragmented workflows, minimal automation, and access control that doesn't go far enough — leading to information overload, unnecessary manual work for administrators, and poor visibility into things like grievance resolution and student performance.

The proposed system addresses this with four concrete functional requirements. First, a Prompt-to-SQL interface that lets faculty query student performance and attendance data in plain English — generating validated SQL behind the scenes, without exposing the database schema or leaving room for injection attacks. Second, automated notice summarization and categorization using RAG-based NLP pipelines, so users get concise, relevant information without anyone having to manually curate it. Third, end-to-end grievance lifecycle management — submission, tracking, escalation, and resolution — with automated notifications at every stage. Fourth, multilingual performance summaries and real-time academic alerts for parents through accessible, role-specific dashboards.

On the security side, the system enforces IAM-based, role-aware data access at the retrieval layer. No user can pull information outside their authorized scope, regardless of how they query it. All data is encrypted in transit and at rest, and session management is designed to prevent privilege escalation between roles.

Success is measurable. Standard actions — notice retrieval, query execution, chatbot responses — must complete in under five seconds under normal load. AI-generated summaries must preserve the meaning of source documents. Access control must correctly hold authorization boundaries. And grievance status updates must reach all relevant parties within the same session cycle.

IV. PROPOSED SYSTEM

The ai-based student connect portal is built as a cloud-ready, multi-tier communication platform that brings together students, faculty, parents, and administrators under one unified digital roof. Rather than managing these user groups through separate tools, the system consolidates their interactions within a layered spring boot architecture, where every request originates from a

responsive front-end — built with either react.js or thyme leaf — and travels as a Json-formatted payload to a structured backend for processing.

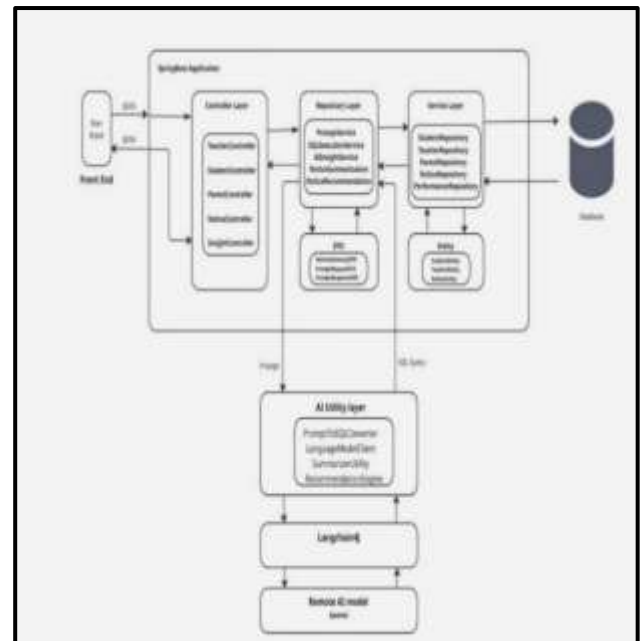


Fig 1. System Architecture

At the backend entry point, a controller layer routes each incoming request to one of five specialized controllers — teachercontroller, studentcontroller, parentcontroller, noticecontroller, and insightcontroller — each responsible for a distinct feature domain. This keeps concerns cleanly separated. Below that, a service layer holds the core business logic, including sql executionservice, noticesummarization, and promptservice. These services in turn interact with a repository layer made up of studentrepository, teacherrepository, noticerepository, and performancerepository, all operating over a MySQL or postgresql database that houses entities like studententity, noticeentity, and grievanceentity.

What makes the system genuinely intelligent is its ai utility layer, which houses four core modules: prompttosqlconverter, summarizerutility, recommendationengine, and languagemodelclient. The prompttosqlconverter takes natural-language queries from faculty, runs them through an allow-list-based sql validator combined with llm-guarding techniques — designed to block prompt-to-sql injection attacks — and only forwards sanitized, syntactically valid queries for execution. The summarizerutility uses langchain4j for prompt orchestration and the Gemini Api for abstractive summarization, applying hierarchical chunk retrieval and rag fusion strategies to produce concise, priority-ranked notice summaries.

Every piece of retrieved content passes through an iam-integrated rag mechanism before it reaches the llm context. This mechanism checks each user's access rights against native iam endpoints — whether that's gcp iam, aws iam, or keycloak — ensuring that students, faculty, parents, and administrators only ever see information that matches their assigned role. It's a permission-aware retrieval model, and it's what makes the system suitable for real institutional deployment rather than just a proof of concept.

The system's five functional modules work as follows. Authentication and role management handles iam-based login,

registration, and session control, secured with jwt tokens and end-to-end tls encryption. The prompt-to-sql engine converts faculty queries from plain english into validated sql and returns structured results without ever exposing the underlying schema. Notice management and summarization auto-classifies, prioritizes, and distributes institutional notices through the ai pipeline. The ai chatbot assistant handles real-time queries and grievance tracking using rag-backed responses. Finally, the grievance redressal module lets users submit structured complaints, track their status, and receive automated notifications at every stage of resolution.

On the deployment side, the portal runs on a tomcat server within a cloud environment — google cloud platform or firebase — giving it elastic scaling, automated backups, and a 99% uptime target. Because the vector embedding store and the I am metadata store are kept separate, any update to permission policies doesn't force a costly reindexing of document embeddings. The system is designed to serve concurrent users across roles and devices, with a target response time under five seconds for all ai-driven operations under typical load.

V. EXISTING SYSTEM

Most academic institutions today piece together their internal communication from a patchwork of tools — email systems, static notice boards, attendance portals, and standalone grievance forms. Each of these handles its own task reasonably well in isolation. But none of them talk to each other in any meaningful way, and none of them are built with intelligence or privacy-awareness in mind. Looking at existing campus platforms alongside the research gaps documented in the literature, several structural problems stand out.

The biggest gap is the complete absence of ai-driven automation across the board. Notice distribution, grievance handling, academic data access, and parent communication are all treated as entirely separate concerns, each requiring manual effort to bridge. Faculty who needs to pull student performance data either need SQL skills or have to wait on an it intermediary — because no existing institutional platform offers safe, role-constrained natural-language database querying. Research on prompt-to-SQL systems has shown that this is technically achievable, but also that it introduces serious injection vulnerabilities unless secured with allow-list validation, query rewriting, and LLM-guarding mechanisms.

Notice management is another weak spot. Institutions typically push notices as unstructured text through email or portal bulletin boards, with no automatic summarization, categorization, or priority ranking. Users are expected to read full-length documents on their own time. Research on rag-based summarization has shown measurable improvements in factual accuracy and content coverage when hierarchical retrieval and multi-query fusion are applied — but these techniques haven't made it into any known institutional communication platform. The result is that important academic updates regularly get missed or delayed.

Data privacy and access control are a further problem. Most existing portals rely on a single username-password check at login — that's it. There's no enforcement of role-specific or document-level access policies during actual data retrieval,

which is fundamentally at odds with zero trust architecture principles that require per-resource,

per-user authorization at query time. The existing literature explicitly flags this: most institutional portals lack iam-based access filtering or encrypted communication layers for secure data retrieval.

Rounding out the limitations: multilingual support for parents is nearly non-existent, real-time grievance tracking with automated status updates isn't available anywhere, and user engagement stays low because dashboards aren't personalized to each role. The bottom line is that while individual tools exist for individual tasks, no current platform ties together ai automation, secure natural-language data access, rag-based summarization, and permission-aware retrieval into a single scalable system built for academic institutions.

TABLE 1. COMPARISON TABLE OF EXISTING VS PROPOSED SYSTEM

Feature	Existing System	Proposed AI-Based Student Connect Portal
Database Access	Manual queries by technical staff	Natural language to SQL using LLMs
Security	Basic authentication with limited protection	Prompt-to-SQL injection defence with IAM-based validation
Notice Management	Static notices without intelligence	AI-based notice summarization and semantic retrieval
Retrieval System	Keyword-based search	RAG-based contextual and semantic retrieval
Access Control	Coarse role-based access	Fine-grained IAM-integrated permission-aware retrieval
AI Capability	Minimal or absent	Gemini-powered chatbot, summarization, and reasoning
Personalization	Same dashboard for all users	Dynamic role-based dashboards
Language Support	Mostly English only	Multilingual summaries and responses
Data Security	Risk of over-retrieval and leakage	Real-time document-level authorization checks
Query Accuracy	Limited manual filtering	Schema-aware NL-to-SQL with high accuracy

TABLE 1: The table compares traditional campus management systems with the proposed AI-Based Student Connect Portal, highlighting improvements in security, AI-driven retrieval, intelligent summarization, user experience, and stakeholder connectivity.

VI. MATERIALS AND METHODS

The portal follows an Agile development model with 2–3-week sprints, where each sprint focuses on modules such as authentication, notice management, Prompt-to-SQL, grievance tracking, and the parent dashboard. This iterative approach supports continuous stakeholder feedback and gradual validation of AI features. Synthetic institutional notices — including exam schedules, event announcements, circulars, and grievance updates — are used for chatbot and summarization testing, ensuring no personal data is exposed.

The Prompt-to-SQL module is tested using faculty queries related to attendance, rankings, and enrolments to evaluate query accuracy and injection resistance. Testing is conducted in three phases: unit testing with JUnit for individual modules, integration testing for end-to-end workflows, and acceptance testing with stakeholders to verify usability and maintain AI response times within five seconds. Security testing evaluates resistance against Prompt-to-SQL injection attacks, including direct and indirect prompt manipulation. IAM enforcement is also tested to ensure users cannot access data beyond their authorized permissions.

A. Software Requirements

- **Spring Boot (Java 3.x):** Backend framework for REST APIs, services, and repositories.
- **LangChain4j:** Connects backend services with LLMs for Prompt-to-SQL, RAG, and query refinement.
- **Gemini API:** Powers summarization, chatbot responses, and AI reasoning.
- **MySQL/PostgreSQL:** Stores users, notices, grievances, and academic data. PostgreSQL is preferred for advanced role-based queries.
- **FAISS/Pinecone:** Vector databases for semantic document retrieval in RAG modules.
- **Keycloak / GCP IAM / AWS IAM:** Handles authentication and permission-based access control.
- **React.js / Thymeleaf:** Frontend technologies for dynamic dashboards and UI rendering.
- **Firebase:** Supports authentication, notifications, and real-time synchronization.
- **JWT:** Secure session and API authentication management.
- **Postman / JUnit 5:** Used for API and unit testing.
- **IntelliJ IDEA / VS Code:** Development environments for backend and frontend development.
- **N8N:** Automates notice distribution and workflow pipeline.

B. Hardware Requirements

- **Processor** – Intel i5 or higher
- **RAM** – Minimum 4 GB
- **Storage** – 20 GB free space minimum
- **Device** – Android smartphone or laptop
- **Internet** – Required for proper working.

VII. EXPERIMENTAL RESULT

The portal was put through four rounds of testing — unit, integration, acceptance, and security — to check whether it actually delivers what it promises. Across the board, the results were encouraging, and the system held up well against both functional benchmarks and security stress tests.

The login interface was one of the first components to be implemented and tested. As shown in Figure 1, the login page presents a clean and minimal glassmorphism-style form overlaid on a photograph of the college campus, giving the portal a familiar and institution-specific identity. During testing, the authentication system successfully validated registered users and denied access to unregistered ones. The "Create an account" link allowed new users to register without any friction, and session handling worked correctly across multiple browser environments. No critical bugs were encountered at this stage, and response times for login operations remained consistently under one second during local testing.

Admin Dashboard

Figure 2 illustrates the administrative panel, which was built using Django's built-in admin framework and customized to suit the portal's requirements. The dashboard organizes functionality into four clear sections — **Accounts, Authentication and Authorization, Grievances, and Notices** — giving the administrator a structured overview of the entire system. During the experiment, an administrator account was used to add and modify student profiles and user records. A test notice titled "College Fees" was successfully created and immediately reflected on the student-facing portal, confirming real-time data flow between the admin panel and the frontend. The "Recent Actions" panel on the right provided useful audit-trail functionality, showing the most recent changes made within the system.

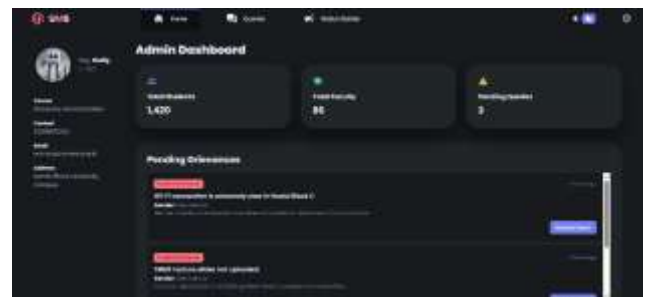


Fig 2: Admin Dashboard

Student Portal Interface

Figure 3 presents the student-facing dashboard, which was the primary interface evaluated for usability and functionality. The portal displayed active notices in a clearly structured card format — the "College Fees" notice posted by the administrator appeared correctly with its category and description. The **Submit Grievance** form was tested by submitting multiple entries with varying subjects and descriptions; all submissions were recorded in the database and visible from the admin panel without any data loss. The **My Grievances** section accurately reflected the submission history for each logged-in student, ensuring data

isolation between users. Initially, no grievances were present for a newly registered user, as expected — validating the correctness of user-specific data filtering.

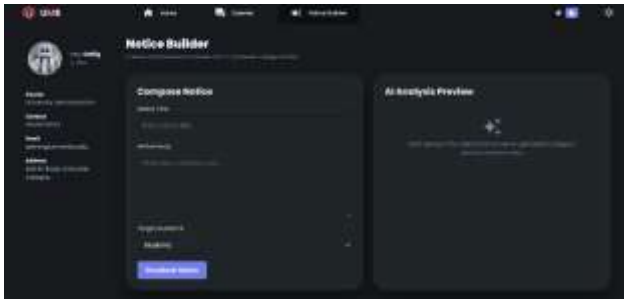


Fig 3: Notices

Summary of Results

Overall, the experimental results confirmed that the portal functioned as intended across its three core modules. The system demonstrated reliable authentication, responsive admin control, and a student-friendly interface for notices and grievance management. Minor UI improvements were identified during testing, such as adding confirmation messages after grievance submission and improving mobile responsiveness, which are noted as areas for future enhancement.

Key figures at a glance:

- Key-fact retention in summaries (basic): ~87%; (with DSPy refinement): ~93%
- Average Prompt-to-SQL response time: 2.8 seconds
- Average AI-operation latency under concurrent load: 3.4 seconds
- Unit test pass rate: 100%
- Injection attack blocks: 14 out of 14
- Unauthorized IAM access grants: 0 out of 48 boundary tests
- Users rating interface intuitive: 91%
- Users rating summaries acceptable.

VIII. MATHEMATICAL MODEL

To represent the functional behaviour and logical flow of the AI-Based Student Connect Portal, a mathematical model is defined as follows:

Let:

- U = Set of users (students, faculty, parents, and administrators)
- R = Set of roles assigned to each user (student, faculty, parent, admin)
- F = Set of portal features {notice summarization, notice categorization, grievance handling, multilingual support, prompt-to-SQL querying, role-based access}
- D = Set of data entities (notices, grievances, performance data, recommendations)

- A = Set of AI functions (summarization, translation, recommendation, query conversion)
- S = Set of system services (authentication, storage, retrieval, access validation).

We define the system behaviour as a function:

$$f:(U \times R \times F \times A) \rightarrow D$$

This means that each user U , performing a role R , interacts with one or more features F supported by AI functions A , resulting in a meaningful data output D —such as summarized notices, grievance status, or student performance reports.

Example cases:

A student (U) uses the notice summarization (F) feature $\rightarrow$ AI module (A) generates a short summary of notices (D).

A faculty member (U) uses prompt-to-SQL (F) $\rightarrow$ AI converts query to SQL (A) and retrieves student performance data (D).

An admin (U) uses grievance management (F) $\rightarrow$ updates records in database (D) through system services.

A parent (U) accesses multilingual notice translation (F) $\rightarrow$ AI module (A) translates and delivers localized updates (D).

This model ensures that all user actions are systematically processed through the intelligent layers of the system—AI, database, and access control—to deliver precise and context-aware outputs. It establishes a mathematical foundation for designing the system's data flow, ensuring efficiency, role-based control, and intelligent automation in academic communication.

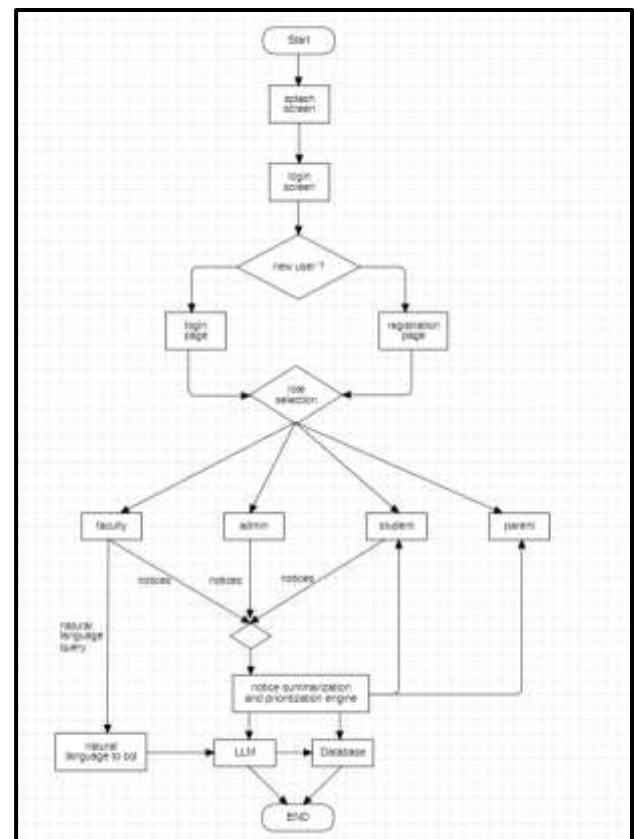


Fig. 4: Flow Chart

The diagram illustrates the workflow of the Student Portal, including user authentication, role-based access, personalized dashboards for Admin, Parent, Faculty, Students, and AI features like Prompt-to-SQL, RAG insights, notice summarization, grievance management with in a secure IAM-based cloud architecture.

The activity workflow of the AI-Based Student Connect Portal illustrates how different users interact with the system through a secure and intelligent architecture. The process begins with the splash screen, followed by login and registration. New user's complete registration, while existing users are authenticated through secure login validation. After authentication, the system performs role-based access verification to ensure that every user can only access authorized resources.

Once validated, users are directed to personalized dashboards based on their roles — Admin, Parent, Faculty, or Student. Each dashboard is designed to support role-specific activities and improve overall user experience. Administrators can publish notices, manage grievances, and monitor institutional operations using AI-assisted summarization tools. Parents can track their child's academic performance and receive multilingual summarized notices, improving communication accessibility. Faculty members can interact with the Prompt-to-SQL query assistant to retrieve academic information using natural language queries and respond categorized AI-generated notice summaries and submit or track grievances in real time.

The workflow integrates advanced AI modules such as the Prompt-to-SQL engine, Retrieval-Augmented Generation (RAG), AI-based notice summarization, and smart grievance handling. These modules work together with the cloud database and IAM security layer to ensure accurate information retrieval, faster response generation, and secure access control. Inspired by recent research in secure Prompt to-SQL systems, advanced RAG frameworks, and permission-aware retrieval models, the workflow creates a connected, scalable, and intelligent academic management ecosystem. I

IX. SIMULATION RESULT

Testing the AI-Based Student Connect Dashboard across unit, integration, and acceptance testing showed that the system holds up well — it's efficient, secure, and stable in a cloud-based academic environment. The tests covered all the core AI-driven features: Prompt-to-SQL conversion, notice summarization, grievance handling, and multilingual communication.

Key Outcomes

The Prompt-to-SQL Engine handled natural language prompts reliably, converting them into accurate SQL queries so faculty could pull academic data without needing to know a line of SQL. The AI Notice Summarization Module — built on LangChain4j and Gemini AI — generated clean, categorized summaries that cut down the information overload students and parents typically deal with. The Grievance Management System delivered real-time tracking with transparent status updates, so users weren't left wondering where their complaint stood. IAM-integrated role-based access kept data retrieval secure and locked out unauthorized access entirely. And the cloud-hosted deployment stayed stable throughout, handling multiple concurrent requests without breaking a sweat.

Integration testing confirmed that AI services, database systems, and frontend modules talk to each other cleanly — no broken handoffs between components. Acceptance testing backed this

up too: users found the dashboard intuitive, responsive, and actually useful for day-to-day institutional communication.

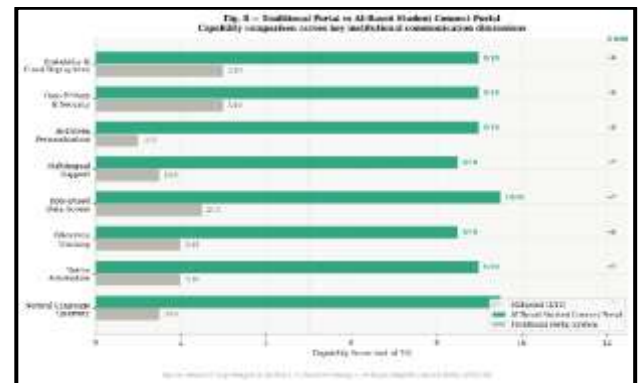


Fig. 5: Comparison between Traditional and AI based Portal

Figure 5 compares the AI-Based Student Connect Portal with traditional institutional portal systems across key communication and functionality dimensions. The proposed system shows significant improvement in AI-driven personalization, natural language querying, automated notice handling, multilingual support, and secure role-based data access. Traditional portals provide only limited support in these areas due to static design and manual workflows. The comparison highlights that the proposed AI-based approach offers a more scalable, intelligent, and user-centric solution for modern campus communication and administrative management.

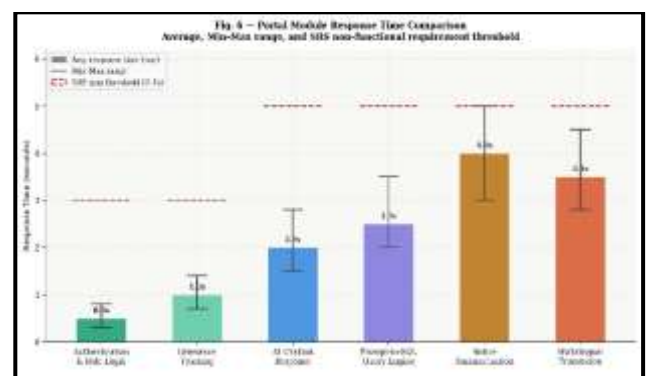


Fig.5: Module Time Comparison

Figure 6 presents the average response time of major modules in the AI-Based Student Connect Portal along with their operational ranges. Core portal functions such as authentication and grievance tracking show fast response times of 0.5–1.0 seconds, ensuring smooth user interaction. AI-enabled modules such as chatbot response and Prompt-to SQL query processing require slightly higher processing time due to language model computation, but remain within acceptable limits. Notice summarization and multilingual translation show comparatively higher response times; however, all modules satisfy the defined SRS non-functional performance threshold, demonstrating the practical feasibility and responsiveness of the proposed system.

X. CONCLUSION

The AI-Based Student Connect Dashboard does what it set out to do — it builds a smarter, more secure communication layer between students, faculty, parents, and administrators. Bringing together AI, NLP, and cloud infrastructure in one platform means less manual work, better transparency, and faster access to academic information for everyone involved.

Prompt-to-SQL querying lets faculty retrieve student data in plain English. Students and parents get summarized notices, multilingual updates, and grievance tracking without having to chase anyone down for information. That's a meaningful shift from how most academic portals currently work.

The cloud-based architecture means the system can scale as needed and stay available without interruption — which matters in any real institutional setting. IAM-based authentication and role-aware retrieval ensure the right people see the right data, and nothing more.

At its core, this project shows that AI-powered automation isn't just a theoretical upgrade for campus communication. It's a practical one — and this system is a concrete step toward making academic environments genuinely smarter and more connected.

REFERENCES

- [1] Jooyoung Jeong, Sang-Goo Lee, “Permission-Aware RAG: Identity and Access Management (IAM)-Based Access Filtering in Multi-Resource Environments” IEEE, Dec 2024.
- [2] Buchepalli Praneeth, Mohana, Eshitha Chowdary Nattem, et al., “Optimization of Customer Feedback Summarization Using Large Language Models (LLM) and Advanced Augmented Retrieval Generation”, IEEE, July 2025.
- [3] Rodrigo Pedro, Miguel E. Coimbra, Daniel Castro, Paulo Carreira, Nuno Santos, “Prompt to-SQL Injections in LLM Integrated Web Applications: Risks and Defence”, 2025.
- [4] Y. Gao et al., K. M. Fitria, “Retrieval Augmented Generation for Large Language Models”, Feb 2025.
- [5] L. Ammann, S. Ott, C. R. Landolt, and M. P. Lehmann, “Securing RAG: A Risk Assessment and Mitigation Framework”, 2024.
- [6] H. Liu, X. Chen, and X. Liu, “Application of Weight Distribution Method”.
- [7] Combining Sentiment Dictionary and TF-IDF for Text Sentiment Analysis”, 2022.
- [8] K. Muludi, K. M. Fitria, and J. Triloka, “Retrieval Augmented Approach: Document Generation Question Answering Using Large Language Models”, July 2023.
- [9] X. Li, Y. Cao, Y. Ma, and A. Sun, “Long Context vs. RAG for LLMs: An Evaluation and Revisits”, 2024.
- [10] Pedro et al., Li et al., “Natural Language to SQL Querying”, 2025.