

Design of a Role-Based Smart Campus Management System with AI-Assisted Student Support and Unified Administrative Workflows

Sushant Thakare, Vaibhav Badgajar, Ashwini Valkhe, Soham Chavan, Mr. S. P. Bhamre

Department of Information Technology

MVPS Rajarshi Shahu Maharaj Polytechnic, Nashik, Maharashtra, India

Abstract: Educational institutions coordinate academic and administrative activities such as student enquiries, attendance, departmental reporting, fee-related processes, notices, complaints, and service requests. When these activities rely on disconnected tools or manual follow-up, information can be difficult to locate and workflows can become inconsistent. This paper proposes a web-based Smart Campus Management System that organizes selected campus services through a unified interface. The proposed application defines role-specific dashboards for Super Admin, Principal, Head of Department (HOD), Faculty, Student, Accountant, and Office Admin. Server-side authorization and department-level access rules are intended to restrict records according to institutional responsibilities. An AI-assisted helpdesk is proposed to answer routine questions using approved institutional information and route unresolved issues to human-managed complaint or service-request workflows. The first implementation is planned as a responsive web application. Since the work is at the proposal stage, this paper presents the architecture, access model, and evaluation plan rather than completed implementation results. Future evaluation will include functional tests, authorization tests, response-time measurements, and assessment of helpdesk answers against a predefined question set.

Keywords: Smart Campus, Role-Based Access Control, Campus Management, AI Helpdesk, Workflow Automation, Web Security

I. INTRODUCTION

Educational institutions manage a range of academic and administrative activities, including attendance, timetable information, notices, student enquiries, fee records, complaints, and departmental reporting. When these activities are handled through separate tools or manual processes, staff and students may need to contact multiple offices to obtain information. Administrators may also spend time forwarding routine requests and checking whether tasks have been completed.

A centralized campus management platform can provide a common interface for selected services while retaining different responsibilities for each user group. However, a shared interface should not mean that every user can access every record. Role-based access control (RBAC) is a recognized approach for assigning permissions through organizational roles rather than granting permissions individually to every user [1], [2]. In a college environment, these permissions can be combined with department-level restrictions so that users access only records required for their duties.

This paper proposes a Smart Campus Management System with seven user roles: Super Admin, Principal, HOD, Faculty, Student, Accountant, and Office Admin. The proposed design combines role-specific dashboards, centralized workflows, department-aware authorization, and an AI-assisted campus helpdesk. The initial target is a responsive web application accessible from desktop and mobile browsers.



A. Objectives

The main objectives are to: (1) define role-specific access to campus information and actions; (2) organize academic and administrative requests through a unified workflow; (3) provide a helpdesk that responds using approved institutional information; and (4) define a test plan for evaluating functionality, authorization, usability, and response time after implementation.

B. Scope of the Paper

The work is currently a proposed design. It does not claim a completed deployment, measured performance improvement, or validated AI accuracy. The following sections describe the proposed architecture and evaluation plan. Implementation and empirical findings will be reported only after the prototype has been developed and tested.

II. PROPOSED SYSTEM AND METHODOLOGY

A. Proposed Architecture

The proposed system uses a browser-based frontend, an application backend, a relational database, and a separate AI-helpdesk service. The frontend presents the dashboard and forms available to the signed-in user. The backend validates each request, checks the user role and department scope, applies workflow rules, and reads or updates permitted records in the database. The AI-helpdesk component retrieves relevant information from a curated set of approved college documents before constructing an answer. If a question cannot be answered confidently from the approved material, the interface should direct the user to the appropriate office or service-request workflow.

A possible implementation stack is React with TypeScript for the user interface, Python with Django REST Framework for backend APIs, and PostgreSQL for structured data. These technologies are recommendations for implementation, not a claim that a working prototype already uses them. Authentication, authorization, input validation, audit logging, and database access controls must be implemented and tested on the server rather than relying only on frontend visibility.

B. User Roles and Responsibilities

The proposed roles and their principal responsibilities are summarized below. Exact permissions should be confirmed with the institution before implementation.

Role	Proposed dashboard access	Key restrictions
Super Admin	User and system configuration; role assignment; system-level monitoring.	Administrative actions restricted to authorized system settings.
Principal	Institution-level overview, summaries, notices, and escalated requests.	Access limited by institutional policy; sensitive records remain permission-controlled.
HOD	Department overview, faculty coordination, and department reports.	Department-scoped records except where wider access is explicitly granted.
Faculty	Assigned classes, attendance-related workflows, notices, and student requests relevant to teaching.	Only assigned classes and permitted student records.
Student	Own profile, notices, requests, and status tracking.	Personal records and own submissions only.
Accountant	Fee-related records, payment status, and finance reports.	Financial permissions separated from academic and general administration.
Office Admin	Student-office workflows, document requests, and general service requests.	Access limited to assigned administrative functions.

TABLE I. PROPOSED USER ROLES AND ACCESS BOUNDARIES



C. Core Workflows

A student should be able to sign in, view relevant notices, submit a request, and track its status. A request is routed to the responsible office or department, assigned to an authorized staff member, updated through defined statuses, and closed with a recorded outcome. Faculty and HOD workflows should be limited to their assigned classes or department scope. Finance-related actions should be available only to authorized accounting users. The system should retain timestamps and relevant status changes to support accountability.

D. AI-Assisted Helpdesk

The helpdesk is intended for routine questions about campus procedures, office contacts, document requirements, and published policies. Responses should be grounded in institution-approved content rather than unrestricted model knowledge. Where the information is missing, outdated, or ambiguous, the helpdesk should acknowledge the limitation and provide a route to a human office. It should not make binding decisions about marks, disciplinary action, fees, scholarships, or other matters that require an authorized institutional decision-maker.

E. Development Method

Development is proposed in incremental stages: requirements confirmation with college stakeholders; role and permission definition; database and API design; implementation of login and dashboards; workflow development; integration of the helpdesk with approved documents; and testing. The first prototype should prioritize essential workflows and security boundaries before adding optional modules.

III. SYSTEM MODELING AND SECURITY DESIGN

A. Authorization Model

For each request, the backend should evaluate at least the authenticated user, assigned role, requested action, target record, and applicable department or ownership scope. A request is permitted only when the policy allows that combination. Hiding a button in the interface is not sufficient because a user may call an API directly. Access checks must therefore be applied at the server-side API and data-access layers. This follows the principle of using explicit access-control policy rather than trusting the client interface [1]–[3].

A simple policy can be represented as: $\text{Permit} = \text{Authenticated} \wedge \text{RolePermission} \wedge \text{ScopePermission}$. “RolePermission” checks whether the role can perform the action, while “ScopePermission” checks whether the record belongs to the user, assigned class, department, or institution-wide scope allowed by policy. Deny should be the default when identity, role, scope, or policy information is missing.

B. Data Entities

The initial database model may include User, Role, Department, StudentProfile, FacultyProfile, Notice, ServiceRequest, RequestStatusHistory, AttendanceRecord, FeeRecord, ApprovedKnowledgeDocument, and AuditEvent. Relationships should be defined so that department and ownership information can be checked consistently. Only the minimum personal data required for each workflow should be collected. Access to financial and student records should be logged and restricted to authorized roles.

C. Security and Privacy Considerations

The proposed application should use secure password handling, protected sessions or tokens, server-side authorization, input validation, rate limiting where appropriate, encrypted transport, backups, and audit records for sensitive actions. Secrets and API keys should be stored outside source code. AI requests should avoid sending unnecessary personal information to external services. The institution should review data retention, user consent, access logs, and the handling of incorrect or sensitive chatbot responses before deployment. These are design requirements to implement and verify, not claims that the current proposal already satisfies them.



IV. EVALUATION PLAN AND DISCUSSION

A. Evaluation Method

After implementation, the prototype should be evaluated using a repeatable test set. Functional tests will check whether the main user journeys complete correctly. Authorization tests will attempt both permitted and prohibited actions for each role and department scope. Performance tests will record response time under a stated test environment and workload. Helpdesk evaluation will use a predefined list of campus questions with expected source documents or answers reviewed by a human evaluator.

B. Planned Evaluation Measures

Measure	Method	Reporting approach
Functional correctness	Execute test cases for login, dashboards, notices, and service requests.	Passed cases / total cases, with failures documented.
Authorization correctness	Test allowed and denied actions across roles and departments.	Unauthorized actions blocked / unauthorized attempts tested.
Response time	Measure API response time under a documented test setup.	Report median and 95th-percentile latency with workload details.
Helpdesk answer quality	Compare answers with approved source material and human review.	Report correct, partially correct, unsupported, and escalated answers.
Usability	Ask representative users to complete defined tasks and provide feedback.	Report task completion and summarized user feedback.

TABLE II. PLANNED EVALUATION MEASURES

C. Discussion

The expected benefit of the design is a clearer division of responsibilities and a common path for routine campus requests. Role-specific dashboards may reduce unnecessary exposure of information, while a traceable request workflow may make request status easier to follow. The helpdesk may reduce repeated questions when approved information is complete and current. These benefits remain hypotheses until the system is implemented and evaluated. Potential limitations include incomplete institutional data, changes in college procedures, incorrect AI responses, user adoption challenges, and the effort required to maintain role permissions.

V. CONCLUSION

This paper presented the proposed design of a role-based Smart Campus Management System for organizing selected academic and administrative workflows through a responsive web application. The design includes seven user roles, department-aware access restrictions, a unified service-request workflow, and an AI-assisted helpdesk grounded in approved institutional information. The paper also defined security considerations and a planned evaluation using functional, authorization, performance, usability, and helpdesk-quality tests. As the work is currently at the proposal stage, no implementation outcomes or performance claims are reported. Future work will involve confirming requirements with college stakeholders, developing a working prototype, conducting the planned tests, and reporting measured results.

ACKNOWLEDGMENT

The author thanks the faculty and project guide at MVPS Rajarshi Shahu Maharaj Polytechnic, Nashik, for their academic guidance. This acknowledgment should be updated to reflect the actual contributions and approvals before submission.



REFERENCES

- [1] W. Jansen, A Revised Model for Role-Based Access Control, NIST Interagency Report 6192, National Institute of Standards and Technology, 1998. doi: 10.6028/NIST.IR.6192.
- [2] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," ACM Transactions on Information and System Security, vol. 4, no. 3, pp. 224–274, 2001. doi: 10.1145/501978.501980.
- [3] V. C. Hu, D. Ferraiolo, and R. Kuhn, Assessment of Access Control Systems, NIST Interagency Report 7316, National Institute of Standards and Technology, 2006. doi: 10.6028/NIST.IR.7316.
- [4] National Institute of Standards and Technology, "Role Based Access Control (RBAC) Project," NIST Computer Security Resource Center. Available: <https://csrc.nist.gov/projects/role-based-access-control> (accessed Oct. 11, 2026).

