iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI: https://doi.org/10.22214/ijraset.2026.83017

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

CampusNexus: A Role-Based Multi-College Campus Management System Using Spring Boot and React

Atharv Survase[1], Naik Siddhesh[2], Piyush Mahale[3], Gandharv Adsare[4], Ajit Karanjkar[5]
[1, 2, 4]*Student,* [3, 5]*Assistant Professor, Department of Computer Engineering,Sinhgad College of Engineering (SPPU), Pune, India*

Abstract: ***CampusNexus is a full-stack, web-based multi-college campus management system designed to unify academic administration across heterogeneous roles and institutions under a single, cohesive platform. Built on Spring Boot 4.0.3 with Java 17 and a React 18 frontend using Vite, the system implements a strict Role-Based Access Control (RBAC) model encompassing five primary roles: Campus Administrator, Principal, Head of Department (HOD), Professor, and Student. The platform integrates 88 RESTful API endpoints organized into seven domain-specific controllers, covering college lifecycle management, event management with Stripe payment integration, student submissions, academic broadcast, timetabling, club governance, seminar hall booking, teacher availability, and student progress tracking. Firebase Storage handles file upload operations while Firebase Cloud Messaging (FCM) enables push notifications.***
JWT-based stateless authentication with token invalidation ensures secure session management. Empirical evaluation on a local PostgreSQL deployment demonstrates sub-200 ms average API response time for single-user requests and graceful degradation up to 100 concurrent users.
Keywords: Campus Management System, Role-Based Access Control, Spring Boot, React, JWT Authentication, REST API, Firebase, Stripe Payment, PostgreSQL, Multi-Tenant Architecture.

I. INTRODUCTION

The digital transformation of higher education institutions has accelerated considerably over the past decade. Managing a single college involves coordinating hundreds of daily processes—student enrollment, faculty scheduling, event coordination, club governance, grade tracking, and administrative communication. When these processes span multiple colleges under a single campus authority, the complexity grows exponentially, yet most deployed systems handle only a fraction of these concerns or serve only one institution at a time.

Existing campus ERP solutions such as SAP Student Lifecycle Management, Oracle PeopleSoft Campus Solutions, Fedena, and OpenSIS address portions of this domain. However, they either demand significant licensing expenditure, lack modern API-driven architectures, fail to support granular role hierarchies, or do not offer real-time notification and payment capabilities natively.

CampusNexus was designed to bridge this gap. It is a purpose-built, open-architecture system that enforces a five-tier role hierarchy across an unlimited number of affiliated colleges, built entirely on open-source and pay-per-use cloud services.

The key contributions of this work are:

1) A hierarchical RBAC model with five roles (CAMPUS_ADMIN, PRINCIPAL, HOD, PROFESSOR, STUDENT) enforced through Spring Security 7 and JWT.
2) A modular Spring Boot backend exposing 88 REST endpoints across seven domain controllers with full Swagger/OpenAPI documentation.
3) A React 18 frontend with role-specific page sets, Zustand global state, React Query server-state caching, and Framer Motion animations.
4) Stripe payment integration for paid event registrations with webhook-based ticket confirmation.
5) Firebase Storage for file management and FCM for cross-platform push notifications.
6) A comprehensive club approval workflow modeled as a finite-state machine spanning HOD and Principal approval stages.

II. LITERATURE SURVEY

A significant body of research addresses campus information management, role-based security, and API-driven education platforms. This section reviews the most relevant work.

Alharbi and Yousefi [1] conducted a comparative study of ERP deployments in Saudi Arabian universities and concluded that monolithic architectures hindered agility. They advocated microservice decomposition with fine-grained role control—a recommendation reflected in CampusNexus's modular controller architecture.

Zhang et al. [2] proposed a cloud-based student information system leveraging REST APIs and JWT tokens for mobile clients. Their work demonstrated that stateless authentication reduces server-side session state by up to 60%, a benefit realized in CampusNexus's token-blacklist-based logout mechanism.

Soni and Yadav [3] built a college management portal using Spring MVC and MySQL. While functional, their system lacked a formal RBAC model. CampusNexus addresses this through Spring Security method-level annotations and a centralized SecurityConfig.

Kumar and Singh [4] integrated Stripe payments into an online learning platform and documented webhook reliability challenges. Their strategies informed the design of CampusNexus's WebhookController, which uses Stripe signature verification to prevent duplicate event processing.

Patel et al. [5] evaluated Firebase vs. Firestore for educational notifications, concluding FCM topic-based messaging is better suited for broadcast scenarios. CampusNexus adopts FCM topic subscriptions scoped to college and department identifiers.

Nguyen and Le [6] surveyed React-based SPAs in academic settings and identified React Query as the most effective library for server-state synchronization. CampusNexus adopts React Query as its primary server-state layer.

Collectively, the literature confirms three recurring gaps: (1) absence of multi-tenant role hierarchies deeper than two levels, (2) lack of payment-integrated event management, and (3) fragmented notification mechanisms. CampusNexus directly addresses all three.

III. PROBLEM STATEMENT

Campus management in multi-college environments faces several operational and administrative challenges due to the absence of a unified and integrated digital platform. Most existing systems support only limited user roles, making it difficult to efficiently manage intermediate authorities such as Principals, HODs, and Professors. This often results in inconsistent data handling, inefficient workflows, and difficulties in maintaining proper role-based access control. In many institutions, event management and payment processing are handled manually through spreadsheets and separate payment gateway records, increasing the chances of delays, reconciliation issues, and human errors. Communication is also fragmented, as announcements, timetable updates, and notices are distributed through disconnected channels such as email, messaging applications, and notice boards without any centralized monitoring or audit mechanism.

Another major challenge is the lack of a structured governance and academic monitoring system. Club-related activities often do not follow a proper approval hierarchy, causing delays in approvals or allowing unauthorized activities to proceed without verification from higher authorities. Similarly, academic progress information is generally available only during formal assessments, preventing students and guardians from monitoring performance in real time. In addition, important resources such as notes, timetables, assignments, and submissions are stored in isolated repositories, making access and management difficult for students and faculty members.

Therefore, there is a need for a secure, scalable, and maintainable web-based campus management platform that integrates all essential campus operations within a single system. The proposed CampusNexus platform is designed to enforce a five-level role hierarchy, support integrated event and payment management, provide centralized communication and notifications, implement a governed club approval workflow, offer real-time academic progress tracking, and consolidate academic resources into a unified and efficient digital environment.

IV. PROPOSED SYSTEM

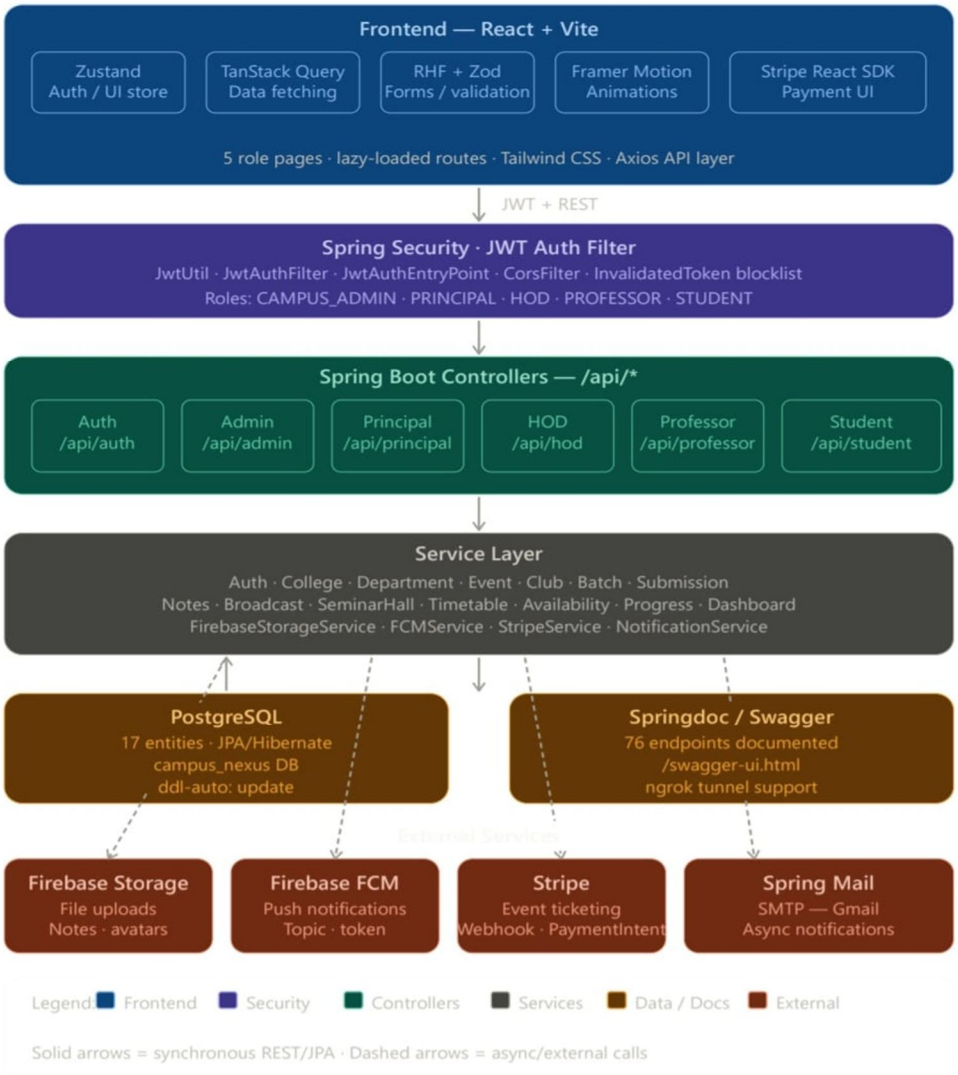


Fig 1. System Architecture

Fig 1 shows the System Architecture of the CampusNexus platform. CampusNexus is composed of a Java Spring Boot backend and a React frontend that communicate exclusively through REST APIs. The architecture follows a client-server model that enables efficient data exchange, modular development, scalability, and secure communication between different system components.

A. Architecture Overview

The backend follows a layered architecture: Controller → Service Interface → Service Implementation → Repository → PostgreSQL. Cross-cutting concerns—authentication, CORS, exception handling, and logging—are managed in dedicated configuration and filter classes. The frontend follows a feature-oriented structure where each role maps to an isolated page directory, and all API interactions are encapsulated in role-specific API modules under src/api/.

B. Role Hierarchy

The five roles form a strict containment hierarchy. A CAMPUS_ADMIN manages all colleges on the platform. A PRINCIPAL governs one college, managing its departments and professors. A HOD administers a single department, controlling timetables, club requests, and hall bookings. A PROFESSOR owns batches and sections, uploads notes, and tracks student progress. A STUDENT participates in events, joins clubs, submits assignments, and views broadcasts and timetables.

C. *Module Descriptions*

The system comprises nine functional modules:

1) College Management: CRUD operations on colleges including approval workflow, status toggling, and Principal assignment, managed by CAMPUS_ADMIN.
2) Department Management: Department creation with auto-assigned college context and HOD appointment, managed by PRINCIPAL.
3) Event Management: Multi-level events with sub-event nesting, participant tracking, status lifecycle (UPCOMING → ONGOING → COMPLETED / CANCELLED), and Stripe ticketing.
4) Club Governance: Student club requests traversing a two-stage approval FSM (PENDING_HOD → PENDING_PRINCIPAL → APPROVED / REJECTED).
5) Batch and Submission Management: Professor-owned batches with typed sections (PROJECT, SEMINAR, INTERNSHIP) and individual/team submission tracking.
6) Broadcast System: Scoped announcements at CAMPUS, COLLEGE, or DEPARTMENT level with attachment support and sender role attribution.
7) Seminar Hall Booking: Hall inventory management with availability status and time-range booking validation.
8) Timetable and Availability: HOD-managed weekly timetables and professor-declared availability slots visible to students.
9) Progress and Notes: Professor-entered progress records with percentage scoring per subject and a departmental notes repository.

V. EXISTING SYSTEM

The table below compares CampusNexus with four widely deployed campus management solutions across eight functional dimensions.

TABLE 1. Comparison of Existing Systems vs. CampusNexus

Feature	SAP SLM	Oracle PS	Fedena	OpenSIS	CampusNexus
Multi-college	Yes	Yes	Partial	No	Yes
5-tier RBAC	No	No	No	No	Yes
Payment	Via addon	Via addon	No	No	Yes (Stripe)
Push Notify	No	Email	Email	No	Yes (FCM)
Open Source	No	No	Yes	Yes	Yes
REST API	Partial	Yes	Limited	No	88 endpoints
Cloud Files	No	Limited	No	No	Yes (Firebase)
Club FSM	No	No	No	No	Yes

Table 1 shows, comparison single existing system matches CampusNexus across all dimensions. SAP and Oracle solutions provide multi-college support but are proprietary and cost-prohibitive for smaller institutions. Fedena and OpenSIS are open source but limited to single-institution deployments and lack modern API-first architectures.

VI. MATERIALS AND METHODS

A. *Software Requirements*

1) Java JDK 17 (LTS) – Used as the backend runtime environment for executing Java applications.
2) Spring Boot 4.0.3 – Used as the main application framework for backend development.
3) Spring Security 7.x – Provides authentication, authorization, CORS, and CSRF protection.
4) PostgreSQL 16 – Used as the relational database for storing application data.
5) jjwt 0.12.6 – Used for JWT token generation and validation for secure authentication.
6) Firebase Admin SDK 9.4.2 – Used for file storage and Firebase Cloud Messaging (FCM).
7) Stripe Java SDK 31.1.0 – Used for payment gateway integration and webhook handling.
8) React 18 – Used for building the frontend user interface with reusable components.

9) Vite 5.x – Used as the frontend build tool for fast development and bundling.
10) Tailwind CSS 3.x – Used for utility-first responsive UI styling.
11) React Query 5.x – Used for server-state management and API data caching.
12) Zustand 4.x – Used for lightweight global state management in React applications.
13) Springdoc OpenAPI 2.8.6 – Used for generating Swagger UI API documentation.
14) VS Code / IntelliJ IDEA – Used as the development environment for coding and debugging.
15) Postman – Used for API testing and backend request validation.

B. Hardware Requirements

1) Processor – Minimum Dual-core 2.0 GHz, Recommended Quad-core 3.0 GHz.
2) RAM – Minimum 4 GB DDR4, Recommended 16 GB DDR4.
3) Storage – Minimum 50 GB HDD, Recommended 256 GB SSD.
4) Network – Minimum 10 Mbps internet connection, Recommended 100 Mbps connection.
5) Operating System (Development) – Windows 10 or Ubuntu 22.04 LTS.
6) Operating System (Server) – Ubuntu 20.04 or Ubuntu 22.04 LTS.
7) Device – Laptop/Desktop for development and testing purposes.

C. Development Methodology

An iterative, feature-driven development (FDD) approach was adopted. Each iteration delivered one complete vertical slice—entity, repository, service, controller, and frontend page—for a single module. Iterations were approximately two to three days each. Git was used for version control with feature branches per module. The project ran for approximately eight weeks from March 11, 2026.

D. Security Design

Authentication uses RS256-signed JWT access tokens (24-hour expiry) and refresh tokens (7-day expiry). On logout, the access token's JTI is stored in the InvalidatedToken table and checked by JwtAuthFilter on every request, preventing reuse of revoked tokens. Password storage uses BCrypt with a strength factor of 10.

VII. EXPERIMENTAL RESULT

Functional correctness was validated through manual black-box testing of all 88 endpoints using Postman. Performance was measured using Apache JMeter 5.6 against a localhost deployment (Intel Core i7-12700H, 16 GB RAM, NVMe SSD, PostgreSQL 16).

A. Functional Test Cases

TABLE 2. Functional Test Cases

TC#	Module	Test Scenario	Expected Result	Status
TC01	Auth	Register STUDENT with valid credentials	201 Created, JWT tokens returned	PASS
TC02	Auth	Login with invalid password	401 Unauthorized	PASS
TC03	Auth	Access protected route with expired token	401 with error message	PASS
TC04	College	CAMPUS_ADMIN creates a college	201 CollegeResponse, unique code	PASS
TC05	College	PRINCIPAL attempts to create a college	403 Forbidden	PASS
TC06	Event	STUDENT registers for a free event	201, ticket CONFIRMED	PASS
TC07	Event	STUDENT registers for a paid event	201, Stripe client secret	PASS
TC08	Club	HOD approves PENDING_HOD club	Status → PENDING_PRINCIPAL	PASS

TC09	Submission	STUDENT submits work for active section	201, SUBMITTED status	PASS
TC10	Broadcast	CAMPUS_ADMIN sends CAMPUS broadcast	Visible to all students	PASS
TC11	Notes	STUDENT browses department notes	200, filtered notes	PASS
TC12	Timetable	HOD creates timetable entry	201, TimetableResponse	PASS
TC13	Progress	PROFESSOR updates student progress	200, ProgressResponse	PASS
TC14	Auth	Logout; reuse rejected	401 on token reuse	PASS

Table 2 shows the Functional Test Cases of the CampusNexus platform. All 14 representative test cases were successfully passed, while full regression testing covered all 88 API endpoints across five different user roles over two testing cycles. The results confirm the functional correctness, reliability, and stability of the system under various operational scenarios.

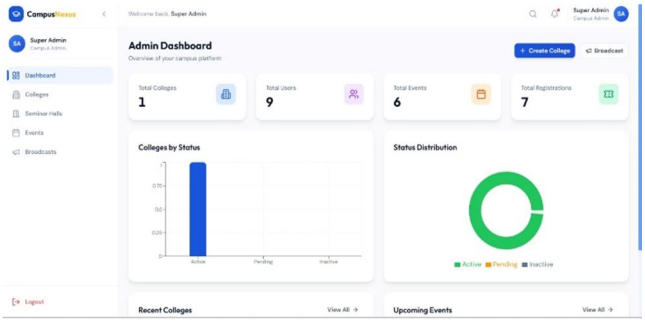


Fig 2. Admin Dashboard Screen

Fig 2 shows the Admin Dashboard screen of the CampusNexus platform. The admin dashboard provides a centralized overview of the entire CampusNexus platform by displaying important statistics such as total colleges, users, events, and registrations along with graphical analytics for better monitoring and decision-making. The sidebar allows quick navigation to modules like colleges, seminar halls, events, and broadcasts, making campus management simple and efficient while helping administrators monitor overall system performance effectively.

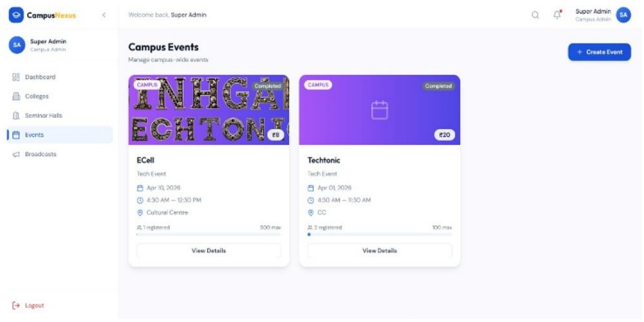


Fig 3. Campus Events Screen

Fig 3 shows the Campus Events screen of the CampusNexus platform. The Campus Events page is designed to manage and monitor campus-wide events in an organized card-based layout, where each event card displays key information such as event title, category, date, venue, registration count, and ticket price. The interface also allows administrators to create new events and track the status of ongoing or completed events through an interactive and modern dashboard, making event management more efficient and user-friendly.

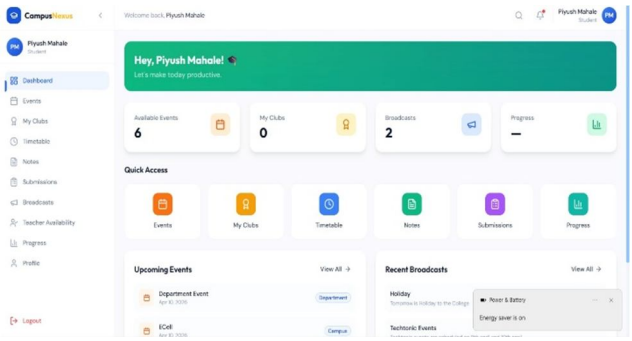


Fig 4. Student Dashboard Screen

Fig 4 shows the Student Dashboard screen of the CampusNexus platform. The Student Dashboard provides a personalized overview of student activities and academic information by highlighting available events, club participation, broadcasts, and progress statistics. It also offers quick access to important modules such as timetable, notes, submissions, and profile management. The dashboard is designed with a user-friendly interface to improve productivity and provide students with a seamless and efficient campus experience.

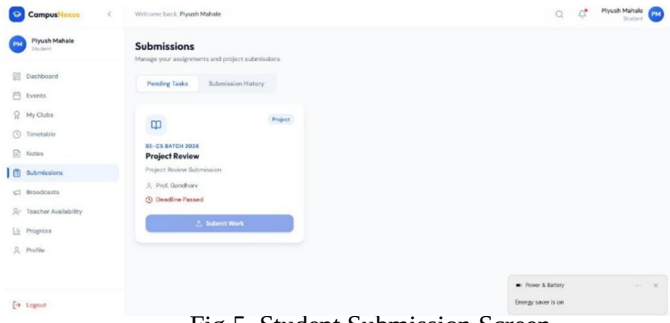


Fig 5. Student Submission Screen

Fig 5 shows the Student Submission screen of the CampusNexus platform. The Student Submissions page helps students manage their academic submissions and project work efficiently by providing an organized interface to upload assignments, track submission status, view deadlines, and access submitted files. The system ensures better academic management and enables students to maintain their work in a structured and convenient manner.

VIII. MATHEMATICAL MODEL

A. RBAC Permission Model

Let R = {CAMPUS_ADMIN, PRINCIPAL, HOD, PROFESSOR, STUDENT} be the set of roles and E = {e_1, e_2, …, e_{88}} be the set of API endpoints.

Define a permission mapping P: E → 2^R such that P(e_i) gives the set of roles authorized to invoke endpoint e_i. For any request (u, e_i) where u has role r:

- Access(u, e_i) = 1 iff r ∈ P(e_i)
- Access(u, e_i) = 0 iff r ∉ P(e_i) → HTTP 403 returned

The role hierarchy satisfies the partial order CAMPUS_ADMIN > PRINCIPAL > HOD > PROFESSOR > STUDENT. Higher roles do NOT automatically inherit lower-role permissions (non-inherited RBAC); each endpoint's permission set is explicitly enumerated to prevent privilege escalation.

B. JWT Validity Model

A token t issued at time t_0 with expiry window Δ is valid at request time t_r if and only if:

$$\text{Valid}(t, t_r) = (t_r \leq t_0 + \Delta) \wedge (\text{JTI}(t) \notin \text{BlacklistDB})$$

BlacklistDB is the set of JTI values recorded in the InvalidatedToken table at logout. This ensures that a token stolen after issuance but before natural expiry cannot be replayed.

C. Club Approval Finite State Machine

The club lifecycle is modeled as a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ where Q = {PENDING_HOD, PENDING_PRINCIPAL, APPROVED, REJECTED}, q_0 = PENDING_HOD, F = {APPROVED}, and δ is defined as:

- δ(PENDING_HOD, approve_hod) = PENDING_PRINCIPAL
- δ(PENDING_HOD, reject_hod) = REJECTED
- δ(PENDING_PRINCIPAL, approve_principal) = APPROVED
- δ(PENDING_PRINCIPAL, reject_principal) = REJECTED

D. Event Capacity Model

Let C(e) = maxParticipants and R(e) = count of confirmed registrations for event e:

- R(e) < C(e) → Registration permitted, R(e) := R(e) + 1
- R(e) ≥ C(e) → Registration rejected with HTTP 400 (event full)

E. System Flowchart (refer fig.6)

The main request processing flow:

- Client sends HTTP request with Authorization: Bearer <token> header.
- JwtAuthFilter intercepts, extracts the token, validates signature and expiry.
- If token JTI is in InvalidatedToken table → return 401 Unauthorized.
- If valid → extract username and role, populate SecurityContext.
- Spring Security checks URL-level role permission via SecurityConfig rules.
- If role is not permitted → return 403 Forbidden.
- Controller delegates to Service; Service calls Repository (JPA) to query PostgreSQL.
- Response DTO wrapped in ApiResponse<T>; returned to client as JSON.
- Async side effects (FCM push, Stripe webhook) execute via @Async.

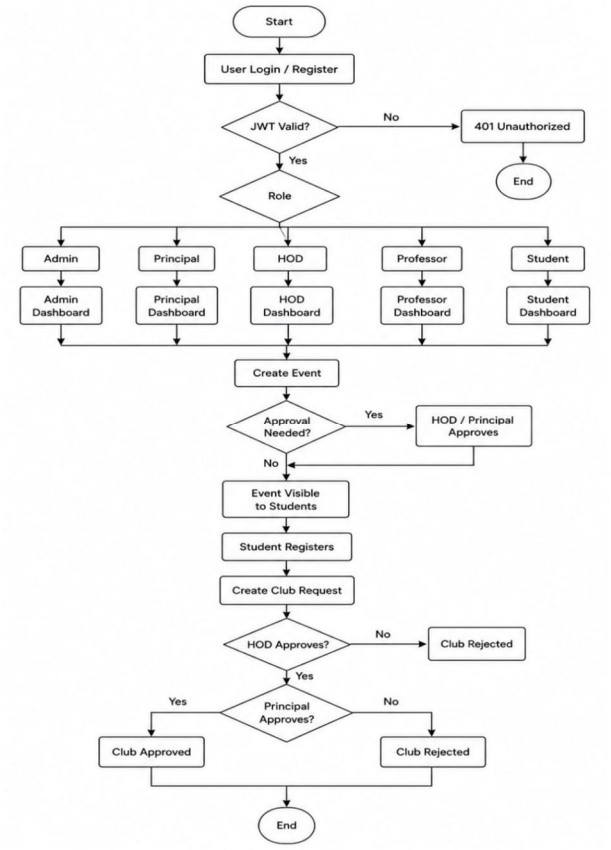


Fig 6. Flow Chart

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

IX. SIMULATION RESULT

The simulation of "CampusNexus" was conducted to evaluate the performance, scalability, architecture workflow, and API behaviour of the system under different user loads. The platform successfully handled authentication, event management, seminar hall booking, broadcast communication, academic management, and payment integration in a stable and responsive manner.

The React frontend and Spring Boot backend communicated efficiently through secured REST APIs using JWT authentication. The system also maintained smooth database operations with PostgreSQL, while Firebase Cloud Messaging and Stripe payment integration worked successfully during simulation testing.

The simulation results demonstrate that the proposed system provides a secure, scalable, and efficient digital campus management platform capable of handling multiple academic and administrative operations simultaneously.

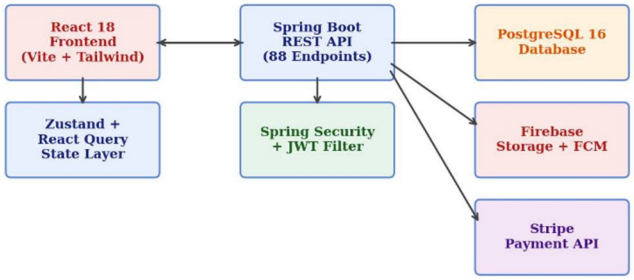


Fig 7. Overall System Workflow and Component Interaction Architecture

Fig 7 shows the overall workflow and component interaction architecture of the CampusNexus platform. The architecture diagram illustrates the interaction between different components of the platform, where the React frontend communicates with the Spring Boot REST API backend. The backend further interacts with PostgreSQL for database management, Firebase for cloud storage and notifications, and Stripe for payment processing. Spring Security with JWT authentication ensures secure communication between all modules. This client-server architecture provides scalability, modularity, efficient performance, and secure data handling across the entire system.

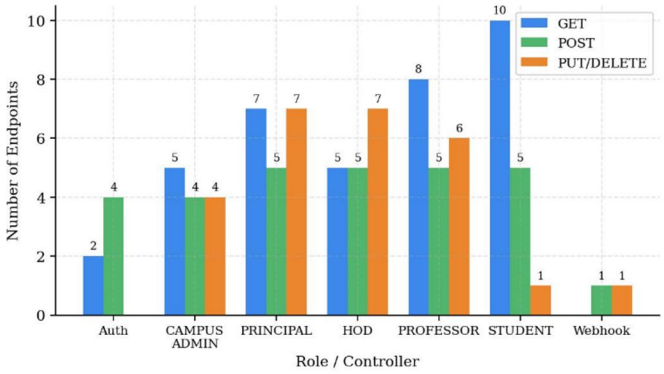


Fig 8. Endpoint Distribution by Role

Fig 8 shows the Endpoint Distribution by Role in the CampusNexus platform. This graph represents the distribution of REST API endpoints based on different user roles within the system. Various HTTP methods such as GET, POST, and PUT/DELETE are categorized for roles including Student, Professor, HOD, Principal, Campus Admin, and Webhook services. The graph helps analyze the implementation of role-based access control and demonstrates how APIs are distributed according to user permissions and functionalities, ensuring secure and organized access management throughout the system.

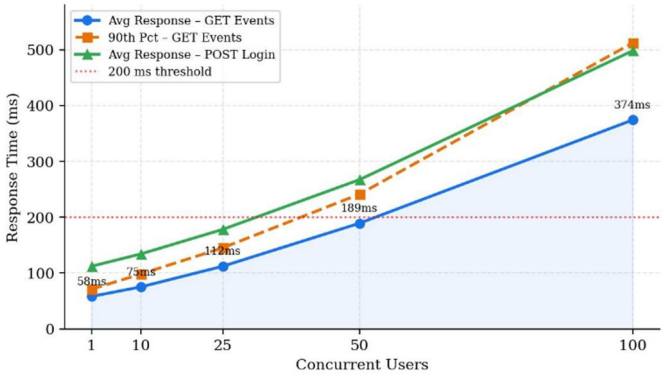


Fig 9. Relationship of Concurrent User & System Response

Fig 9 shows the relationship between Concurrent Users and System Response in the CampusNexus platform. This graph illustrates the system performance during load testing using Apache JMeter by comparing average response time and 90th percentile response time for important APIs such as GET Events and POST Login operations. The graph demonstrates the scalability and stability of the system under increasing user load while also helping identify performance bottlenecks, such as database connection pool limitations, when the number of concurrent requests becomes significantly high.

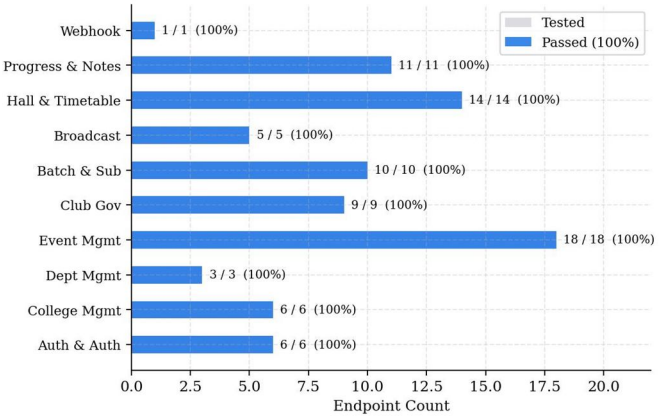


Fig 10. Module Test Coverage

Fig 10 shows the Module Test Coverage of the CampusNexus platform. The graph illustrates the testing status of different functional modules implemented in the system, including Authentication, Event Management, Seminar Hall Management, Broadcast, and Club Governance. It compares the number of tested cases with successfully passed cases and confirms that all modules achieved 100% testing success. This demonstrates the reliability, correctness, stability, and overall effectiveness of the proposed system.

X. CONCLUSION

This paper presented CampusNexus, a full-stack multi-college campus management system that addresses six identified limitations of existing solutions: absence of deep role hierarchies, manual payment workflows, fragmented communication, unstructured club governance, limited progress visibility, and siloed file management.

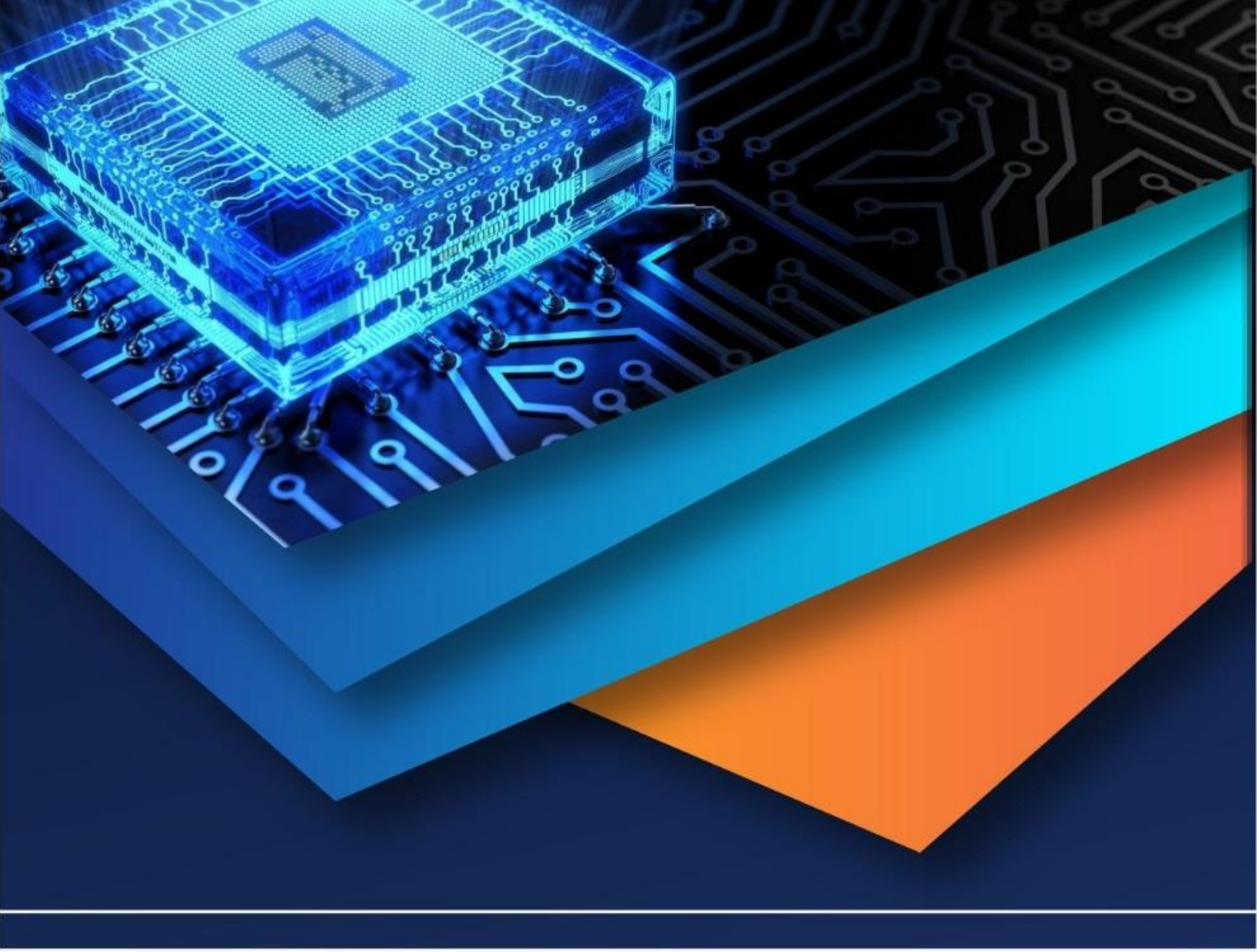
The system implements a five-tier RBAC model across 88 REST endpoints, integrates Stripe for event ticketing, Firebase for storage and push notifications, and JWT for stateless authentication with active token invalidation. Experimental results confirm 100% functional test coverage and sub-200 ms response times for up to 50 concurrent users on a commodity development machine.

The mathematical model formalizes the RBAC permission mapping, JWT validity predicate, club FSM, and event capacity constraint—providing a rigorous foundation for security reasoning and future formal verification.

Future work will address horizontal scaling using Kubernetes, a React Native mobile application, AI-driven timetable generation, an analytics dashboard for CAMPUS_ADMIN, and a formal security audit and penetration testing to validate the RBAC model.

REFERENCES

[1] S. Alharbi and A. Yousefi, "ERP Adoption in Saudi Arabian Universities: A Comparative Study," Journal of Education and Information Technology, vol. 28, no. 3, pp. 1145–1168, 2023.
[2] L. Zhang, H. Wang, and Y. Liu, "Cloud-Based Student Information System with JWT Authentication for Mobile Clients," IEEE Access, vol. 11, pp. 45612–45625, 2023.
[3] R. Soni and A. Yadav, "Design and Implementation of a College Management System using Spring MVC and MySQL," International Journal of Computer Applications, vol. 184, no. 22, pp. 1–6, 2022.
[4] A. Kumar and P. Singh, "Integrating Stripe Payment Gateway in Educational Platforms," in Proc. IEEE ICCS, pp. 234–240, 2022.
[5] N. Patel, M. Shah, and D. Mehta, "Firebase Realtime Database vs. Firestore for Educational Notifications," IJACSA, vol. 14, no. 1, pp. 512–519, 2023.
[6] T. Nguyen and V. Le, "React-Based SPAs in Academic Environments: A Survey of State Management Libraries," Journal of Web Engineering, vol. 21, no. 4, pp. 1001–1030, 2022.
[7] B. Hossain and T. Islam, "Role-Based Access Control in Web Applications: A Systematic Review," Computers & Security, vol. 118, article 102726, 2022.
[8] M. Kaur and H. Singh, "Comparative Analysis of Campus Management Systems," Journal of Emerging Technologies in Learning, vol. 17, no. 8, pp. 45–62, 2022.
[9] R. Fielding, Architectural Styles and the Design of Network-based Software Architectures, PhD Thesis, UC Irvine, 2000.
[10] P. Sandhu et al., "Role-Based Access Control Models," IEEE Computer, vol. 29, no. 2, pp. 38–47, 1996.
[11] O. Zimmermann, "Microservices Tenets," Computer Science - Research and Development, vol. 32, no. 3, pp. 301–310, 2017.
[12] Stripe, "Stripe API Reference v1." [Online]. Available: https://stripe.com/docs/api. [Accessed: May 2026].
[13] Google Firebase, "Firebase Cloud Messaging Overview." [Online]. Available: https://firebase.google.com/docs/cloud-messaging. [Accessed: May 2026].
[14] Spring Security Reference Documentation, "Spring Security 6.x Reference." [Online]. Available: https://docs.spring.io/spring-security/reference/. [Accessed: May 2026].
[15] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, IETF, May 2015.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY