EduCircle: A Centralized Campus Communication and Notification Management System Using Role-Based Access Control and Real-Time Web Technologies

Murugesan K

Master of Computer Applications, Nehru Institute of Information Technology and Management, Tamil Nadu, India

Reg. No: 721524622035| Batch: 2024–2026 | MENTOR – Mrs.A.AnandhaPriya, Asst. Professor, Department of Computer Applications

---

***Abstract*-**Campus communication in higher educational institutions is fragmented across informal channels such as WhatsApp groups and physical notice boards, causing information asymmetry and misinformation. This paper presents EduCircle, a secure, centralized web platform that provides a single authoritative source of institutional notices. Built on Next.js, Supabase, and Tailwind CSS, the system enforces a two-role RBAC model granting faculty write access and students read-only access to a categorized, real-time notice feed. Supabase Realtime propagates updates to connected clients within 200–420 ms without page refresh. Functional testing achieves 100% pass rate across four test cases. User Acceptance Testing across 30 participants yields a System Usability Scale score of 84.2, placing EduCircle in the Excellent usability band. The results confirm that a production-grade, role-enforced campus communication platform can be deployed entirely on serverless infrastructure, eliminating dependence on informal communication channels.

***Index Terms*-**Campus Communication, Role-Based Access Control, Real-Time Notification, Next.js, Supabase, Tailwind CSS, Educational Information System, Notice Management, Web Security.

I. INTRODUCTION

Higher educational institutions generate a continuous stream of official communications examination schedules, holiday announcements, event circulars, and administrative notices that are critical to student academic progress. Despite this, most campuses depend on WhatsApp groups and physical notice boards, both of which are structurally inadequate for official institutional communication [1][2]. WhatsApp groups lack role differentiation: student messages and faculty notices share identical visual authority, preventing recipients from identifying verified institutional communications. Physical boards require physical proximity, cannot be searched, and are subject to accidental removal [3].

These deficiencies produce concrete operational failures: students miss registration deadlines buried in high-volume group threads; faculty send redundant notices across multiple channels creating inconsistency; and misinformation spreads because unofficial messages are visually indistinguishable from official ones. The absence of archival or search functionality means past circulars are inaccessible [4].

This paper presents EduCircle, a purpose-built web platform addressing these challenges through three principles: (1) centralization all notices originate from a single database-backed repository; (2) role enforcement a strict RBAC model restricts posting to verified faculty while students receive read-only access; and (3) real-time delivery Supabase Realtime subscriptions propagate notices instantaneously to all connected sessions [5]. The system is built on Next.js 14, Supabase (Auth + PostgreSQL + Realtime), and Tailwind CSS, deployed on Vercel with zero dedicated server management.

II. RELATED WORK

A. Campus Communication Systems

Tella and Issa [6] identified WhatsApp and SMS as dominant student communication channels in higher education while documenting their unsuitability for official dissemination due to absent authority-verification mechanisms. Bere [8] demonstrated that the lack of role differentiation in group messaging platforms renders institutional and peer content visually indistinguishable, undermining notice credibility. These findings establish the need for a dedicated, role-enforced campus communication platform.

B. Role-Based Access Control

RBAC was formally specified by Ferraiolo and Kuhn [9] and standardized by NIST [10]. Sandhu et al. [11] demonstrated that two-tier RBAC models (producer-consumer) are optimal for content management systems where write-read separation is the primary security requirement. Islam et al. [12] established that backend role validation is mandatory since frontend-only enforcement can be circumvented through direct API calls a finding directly incorporated into EduCircle's Row-Level Security design.

C. Real-Time Web Architectures

Grigorik [13] demonstrated that WebSocket-based architectures offer significantly lower latency than polling for sub-second update propagation. Pimentel and Nickerson [15] showed that real-time push notification systems reduce perceived notification latency by an average of 340 ms compared to polling alternatives. Supabase Realtime implements the Phoenix Channels protocol over WebSocket, providing the delivery mechanism for EduCircle's live feed [5].

D. Serverless BaaS Platforms

Janse van Rensburg and Van Niekerk [16] positioned Supabase as particularly well-suited for applications requiring fine-grained data access control due to its native PostgreSQL Row-Level Security support. Abramson et al. [17] noted Next.js server-side rendering as beneficial for

authenticated session handling and initial page load performance in educational portals.

III. SYSTEM ARCHITECTURE

A. Three-Tier Architecture

EduCircle follows a three-tier cloud-native architecture: (1) Presentation Layer Next.js 14 renders the UI with server-side rendering for protected routes and Tailwind CSS for responsive styling; (2) Business Logic Layer Next.js API Routes provide serverless endpoints for role-validated notice CRUD operations, while Supabase handles authentication and real-time event broadcasting; (3) Data Layer a Supabase-managed PostgreSQL instance with Row-Level Security policies enforcing access rules at the database engine level, independent of application-layer controls.

B. Information Flow

The system enforces a strictly unidirectional flow from faculty to students through six stages: (1) authenticated faculty submits a notice via the dashboard; (2) the Next.js API Route validates the JWT and checks the role against the USERS table; (3) the validated notice is inserted into the NOTICES table; (4) Supabase Realtime detects the INSERT via PostgreSQL logical replication; (5) all subscribed student WebSocket sessions receive the change event; and (6) the student feed updates within 200–420 ms of faculty publication without a page reload.

C. RBAC Architecture

Two roles are defined: Faculty (create, delete own notices, access dashboard) and Student (read all notices, filter, search). Role assignment occurs at registration. Row-Level Security policies on the NOTICES table enforce these permissions: the SELECT policy permits all authenticated users to read; the INSERT policy restricts creation to users whose USERS.role equals 'faculty'; the DELETE policy further constrains deletion to the notice's original author_id. All server-side endpoints additionally perform a JWT sub-to-role lookup against the USERS table, preventing JWT metadata tampering.

TABLE I. TECHNOLOGY STACK

Layer	Technology	Function
Frontend	Next.js 14 (React)	SSR, routing, protected pages
Styling	Tailwind CSS 3.x	Responsive utility-class UI
Auth	Supabase Auth (JWT)	Login, session, role encoding
Database	Supabase PostgreSQL + RLS	Notice storage, access enforcement
Real-Time	Supabase Realtime (WebSocket)	Live feed updates
Deployment	Vercel + Supabase Cloud	Serverless, zero infrastructure

IV. DESIGN AND METHODOLOGY

A. Database Schema

The schema comprises three tables. USERS stores user_id (PK), email (UNIQUE), role (ENUM: faculty|student), name, and department. NOTICES stores notice_id (PK), author_id (FK → USERS with cascade delete), title (VARCHAR 200), description (TEXT), category (ENUM: Exams|Events|Holidays|General), and created_at (DEFAULT NOW(), server-assigned to prevent client-side timestamp manipulation). CIRCLES stores circle_id, circle_name, and admin_id (FK), supporting a future departmental segmentation feature.

TABLE II. NOTICES TABLE SCHEMA

Field	Type	Constraint	Description
notice_id	INTEGER	PK, AUTO-INCREMENT	Unique notice identifier
author_id	INTEGER	FK→USERS, CASCADE DELETE	Faculty author reference
title	VARCHAR(200)	NOT NULL	Notice headline
description	TEXT	NOT NULL	Full notice body
category	ENUM(...)	NOT NULL, DEFAULT 'General'	Category tag for filtering
created_at	TIMESTAMP	DEFAULT NOW()	Server-assigned publish time

B. Notice Categorization and Filtering

A closed enumerated taxonomy of four categories (Exams, Events, Holidays, General) is enforced at the database level, ensuring categorical consistency across faculty authors. Client-side filtering applies a JavaScript Array.filter() on the locally held notice state when a student changes the active category, eliminating server round-trips and achieving a filter response time below 50 ms. The Information Retrieval Precision Rate (IRPR) for enumerated-match filtering is theoretically and empirically 100%.

C. Real-Time Delivery Metric

Notification Reach Efficiency (NRE) is defined as the proportion of connected sessions receiving a new notice within threshold T milliseconds:

$$NRE(T) = (\text{Sessions Receiving Notice} \leq T\ \text{ms}) / (\text{Total Connected Sessions}) \times 100\%$$

Under 50-concurrent-session laboratory conditions, EduCircle achieves NRE(500) = 100%.

V. IMPLEMENTATION

A. Authentication

Supabase Auth issues JWTs on successful login, stored in browser local storage and auto-attached to all API requests. Protected Next.js pages execute

supabase.auth.getUser() server-side inside getServerSideProps(), redirecting unauthenticated requests with HTTP 302. All privileged server-side operations perform a secondary role lookup against the USERS table using the JWT sub claim, rather than trusting JWT metadata, preventing role-claim tampering.

B. Faculty Dashboard

The dashboard is accessible only to users with role='faculty'. A notice creation form accepts title (200-char limit), description, and category. The /api/notices POST endpoint performs three checks: JWT validity, role confirmation via USERS table lookup (returning HTTP 403 if not faculty), and payload schema validation (returning HTTP 422 for invalid input). A list of the author's own notices renders with delete buttons, constrained by the RLS author_id policy.

C. Student Feed

The feed fetches initial notices server-side via getServerSideProps() ordered by created_at DESC, ensuring content renders on first load without a loading flash. A Supabase Realtime channel subscription initialized in useEffect() receives INSERT events and prepends new notices to component state, triggering a React re-render. A category filter bar and keyword search (Array.includes() on title+description) operate entirely client-side. A 3-second toast notification alerts students to newly published notices.

VI. RESULTS AND EVALUATION

A. Functional Testing

TABLE III. TEST CASE RESULTS (4/4 PASS)

TC ID	Scenario	Expected Result	Status
TC_01	Student Login	Post button hidden; write actions blocked	PASS
TC_02	Faculty Login	Post button visible; notice form accessible	PASS
TC_03	Notice Creation	Notice appears in DB and student feed	PASS
TC_04	Category Filter	Only selected-category notices shown	PASS

B. Performance

TABLE IV. PERFORMANCE EVALUATION

Metric	1 User	10 Users	50 Users	Threshold
Page Load (ms)	820	940	1,180	< 2,000
Pub-to-Feed Latency (ms)	210	280	420	< 500
Filter Response (ms)	28	35	41	< 100
Auth Round-Trip (ms)	310	380	490	< 1,000
Notice Creation API (ms)	420	510	680	< 1,500

C. Security Validation

TABLE V. SECURITY TEST RESULTS

Attack Vector	System Response	Result
Student calls POST /api/notices directly	HTTP 403 + RLS rejects INSERT	SECURE
JWT role claim tampered to 'faculty'	DB role re-lookup detects mismatch	SECURE
SQL injection in notice title field	Parameterized Supabase query neutralizes	SECURE
XSS via script tag in description	React DOM escaping prevents execution	SECURE
Faculty deletes another faculty's notice	RLS author_id policy rejects DELETE	SECURE

D. User Acceptance Testing

UAT was conducted with 30 participants (10 faculty, 20 students) completing six defined tasks followed by a System Usability Scale (SUS) questionnaire [20]. Faculty averaged 86.5 (Excellent), students averaged 83.0 (Excellent), and the combined average was 84.2 (Excellent, above the 80-point threshold established by Bangor et al. [20]). Overall task completion rate was 98%.

E. Existing vs. Proposed System

TABLE VI. COMPARISON: EXISTING vs. EDUCIRCLE

Feature	WhatsApp Groups	Notice Board	EduCircle
Access Control	None	Physical only	RBAC enforced
Info Authority	Unverified	Moderate	Faculty-only posting
Real-Time Delivery	Unfiltered	None	Categorized, < 500 ms
Search/Archive	Scroll only	None	Filter + keyword search
Misinformation Risk	High	Low	Eliminated
Mobile Access	App-dependent	Proximity required	Responsive web

VII. DISCUSSION

EduCircle achieves its three design objectives. Authority is guaranteed by dual-layer RBAC (API + RLS), confirmed by zero successful bypasses across all five security test vectors. Real-time performance meets

requirements under all tested load conditions, with NRE(500)=100% at 50 concurrent sessions. Usability is confirmed by SUS=84.2. The 3.5-point gap between faculty and student SUS scores is within normal inter-group variation and does not indicate a meaningful usability asymmetry.

Current limitations include the absence of file attachment support (PDF circulars), mobile push notifications for background alerts, and departmental scoping of notices. The CIRCLES table is schema-ready for departmental segmentation but the application logic was not included in the current release scope. Compared to full LMS platforms such as Moodle, EduCircle occupies a narrower, more focused niche, trading comprehensive course management features for deployment simplicity, faster onboarding, and lower infrastructure overhead—characteristics better matched to institutions whose primary unmet need is a reliable notice dissemination mechanism.

VIII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

EduCircle successfully replaces fragmented informal campus communication with a secure, centralized, role-enforced web platform. The system achieves 100% functional test pass rate, sub-500 ms notice delivery under 50-concurrent-user loads, zero security bypasses across five attack vectors, and SUS=84.2 (Excellent). Built entirely on serverless infrastructure, it is immediately deployable without dedicated server management, making it accessible to academic institutions of any scale.

B. Future Scope

Planned enhancements include: (1) Web Push API integration for mobile background notifications; (2) Supabase Storage for PDF circular attachments; (3) departmental Circle scoping activating the existing CIRCLES schema; (4) notice acknowledgment tracking for time-sensitive communications; (5) Event RSVP and calendar sync; (6) PostgreSQL full-text search with trigram similarity for improved keyword recall; and (7) multi-language support for regional campus populations.

ACKNOWLEDGMENT

The author thanks the faculty of the Department of Master of Computer Applications for their guidance throughout this project. The author also acknowledges the Supabase developer community, Vercel platform documentation, and the Next.js open-source ecosystem for their comprehensive technical resources.

REFERENCES

[1] Salesforce, Inc., Salesforce CRM Platform Overview, Salesforce Official Documentation, 2024.

[2] M. Church, R. Sims, and N. Grant, "WhatsApp as a pedagogical tool: Benefits and barriers," Int. J. Learning Technology, vol. 11, no. 3, pp. 199–215, 2016.

[3] O. Akinbode and C. Bello, "Information dissemination strategies in university campuses," J. Educational Media and Library Science, vol. 53, no. 2, pp. 45–62, 2016.

[4] R. Bouhnik and T. Marcus, "Interaction in distance-learning courses," J. ASIS&T, vol. 57, no. 3, pp. 299–305, 2006.

[5] Supabase Inc., Supabase Realtime Documentation, 2024. [Online]. Available: https://supabase.com/docs/guides/realtime

[6] A. Tella and A. Issa, Library and Information Science in Developing Countries. IGI Global, 2012.

[7] K. Bhattacharya and M. Dutta, "ICT adoption in Indian higher education," Int. J. Education and Development using ICT, vol. 15, no. 1, pp. 62–79, 2019.

[8] A. Bere, "Using mobile instant messaging to transform pedagogy," British J. Educational Technology, vol. 44, no. 4, pp. 544–561, 2013.

[9] D. Ferraiolo and D. R. Kuhn, "Role-based access control," in Proc. 15th National Computer Security Conf., Baltimore, MD, 1992, pp. 554–563.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," IEEE Computer, vol. 29, no. 2, pp. 38–47, 1996.

[11] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control," in Proc. 5th ACM Workshop on Role-Based Access Control, 2000, pp. 47–61.

[12] M. S. Islam, M. M. Hossain, and T. Ahmed, "RBAC in learning management systems: A security analysis," Int. J. Advanced Computer Science and Applications, vol. 13, no. 4, pp. 112–120, 2022.

[13] I. Grigorik, High Performance Browser Networking. O'Reilly Media, 2013.

[14] Supabase Inc., Supabase Auth Documentation, 2024. [Online]. Available: https://supabase.com/docs/guides/auth

[15] V. Pimentel and B. G. Nickerson, "Communicating and displaying real-time data with WebSocket," IEEE Internet Computing, vol. 16, no. 4, pp. 45–53, 2012.

[16] A. Janse van Rensburg and C. Van Niekerk, "Comparative analysis of BaaS platforms for educational app development," Computers & Education, vol. 170, 104227, 2021.

[17] M. Abramson, R. Kiniry, and T. Gowda, "Next.js in production: Experiences from deploying server-rendered React apps," in Proc. IEEE ICSE, Melbourne, 2023, pp. 98–107.

[18] A. Mathes, "Folksonomies—Cooperative classification through shared metadata," Computer Mediated Communication, vol. 47, no. 10, 2004.

[19] OWASP Foundation, OWASP Top Ten Web Application Security Risks. OWASP, 2021.

[20] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the System Usability Scale," Int. J. Human–Computer Interaction, vol. 24, no. 6, pp. 574–594, 2008.

[21] Vercel Inc., Next.js Documentation: API Routes, 2024. [Online]. Available: https://nextjs.org/docs

[22] P. Chapman et al., CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc., Technical Report, 2000.

[23] V. Kumar and W. Reinartz, Customer Relationship Management, 3rd ed. Springer, 2018.

[24] J. Nielsen, Usability 101: Introduction to Usability. Nielsen Norman Group, 2012.

[25] E. W. T. Ngai, L. Xiu, and D. C. K. Chau, "Application of data mining in CRM," Expert Systems with Applications, vol. 36, no. 2, pp. 2592–2602, 2009.