International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Alumni Connect: A Secure Role Based Real Time Professional Networking Platform for Academic Communities

U. Anji[1], **K. Beeralingappa**[2], **D. Teja Vardhan Naidu**[3], **Y. Bhanu Prakash**[4]

[1234] *Department of Electrical and Electronics and Engineering,*

G. Pulla Reddy Engineering College (Autonomous), Kurnool, Andhra Pradesh, India

Affiliated to Jawaharlal Nehru Technological University – Anantapur, Ananthapuramu

https://doi.org/10.55041/ijsmt.v2i4.529

Cite this Article: Anji, U., Naidu, D. T. V., Prakash, Y. B. & Beeralingappa, K. (2026). Alumni Connect: A Secure Role Based Real Time Professional Networking Platform for Academic Communities. International Journal of Science, Strategic Management and Technology, *02*(04). https://doi.org/10.55041/ijsmt.v2i4.529

Abstract — This paper presents Alumni Connect, a Web app and Website developed using code platforms and AI-assisted tools to bridge the gap between college alumni and current students. The development workflow begins with wireframe design and interactive prototype creation in Figma, providing a comprehensive visualization of the user experience. AI-powered online tools are subsequently used to transform the Figma prototype into a functional frontend interface without conventional programming. Backend services and dynamic features including user authentication, cloud data storage, live meetings, real-time chat, and voice/video calls are implemented through a curated stack of platform APIs: Firebase for authentication and Mangodb for db and cloudinary for storage database, Agora.io for real-time voice and video communication, Agora sdk for multi-participant virtual meetings, and for code app assembly and data binding. The primary objective of Alumni Connect is to foster meaningful, structured connections between alumni and students, supporting mentorship, career guidance, and the overall development of the college community. Evaluation results demonstrate successful delivery of all target features with low development overhead, confirming the viability of the code and AI-assisted paradigm for institutional networking applications.

Keywords — *Alumni Connect, Code Development, Figma Prototyping, AI-Assisted Frontend, Firebase, Agora.io, Agora sdk, Web app with PWA feature, Alumni Networking, College Community.*

I. INTRODUCTION

Alumni associations serve as vital resources for academic institutions, bridging current students with past graduates through career opportunities, mentorship programmes, and institutional support. Despite their importance, most alumni engagement relies on fragmented channels open social networks, informal WhatsApp groups, or read-only ERP portals none of which provide verification, role governance, or accountability.

Existing solutions present well-documented limitations. LinkedIn [1] does not enforce institutional identity; any user can claim college affiliation without proof. WhatsApp groups offer no structured profiles, no searchability, and no content moderation. College ERP systems manage academic records but lack social networking features entirely. Dedicated alumni portals such as Alma Connect provide basic directories but lack real-time messaging, RBAC, and audit logging [2].

Alumni Connect addresses this gap by combining professional networking depth with the institutional trust model required by colleges. RBAC is enforced at three independent layers, every privileged action is immutably logged, and real-time communication is delivered via Socket.IO and Agora SDK — all within a college-scoped boundary secured by JWT and BcryptJS. The contributions of this paper are:

(i) A complete architectural design with formal three-layer RBAC and JWT-based session management using BcryptJS password hashing and firebase OTP.

(ii) Implementation details for eight production-ready modules including Socket.IO real-time messaging, Agora SDK voice/video calls, Cloudinary media management, and MongoDB/Mongoose data layer.

(iii) Empirical performance and security test results across 35 test cases, validated with Playwright end-to-end tests and Jest unit tests, demonstrating 100% functional correctness.

II. RELATED WORK

Research in institutional social networks consistently identifies verified identity and RBAC as prerequisites for trust in academic communities [3]. Several prior systems have been proposed and evaluated.

Barabási et al. [4] demonstrated that professional networks exhibit scale-free properties, making targeted search by branch and batch year more effective than general-purpose graphs. This motivated the Alumni Connect alumni directory and search module design.

Al-Khalifa and Al-Eidan [5] surveyed alumni systems at Gulf universities and found that 78% lacked document verification, allowing unverified users to claim alumni status. Their findings directly informed the verification workflow in Section IV-B.

Rao and Srinivas [6] proposed a role-based alumni portal for Indian engineering colleges but did not address real-time messaging or multi-device session management. Alumni Connect extends their role model with JWT security stamps for instant token invalidation and a full Socket.IO layer.

Kaur et al. [7] evaluated WebRTC-based voice/video solutions for academic platforms and found Agora SDK superior in NAT traversal reliability. This guided the choice of Agora SDK for all voice and video calls in Alumni Connect.

The research gap remains: no existing system simultaneously provides institutional verification, three-layer RBAC, real-time Socket.IO communication, Agora-powered calling, MongoDB-backed storage, and an immutable audit log within a college-scoped boundary. Alumni Connect closes this gap.

III. SYSTEM ARCHITECTURE

A. Architectural Style

Alumni Connect adopts a Modular Monolith (Service-Oriented) architecture deployed as a single Next.js 14 application. Each of the eight modules owns its data collections in MongoDB and exposes functionality through service functions. No module directly accesses another module's Mongoose models; all inter-module communication passes through defined service interfaces.

B. Technology Stack

The frontend is built with React, Next.js 14 (App Router), TypeScript, Tailwind CSS, and Radix UI for accessible component primitives. The backend uses Next.js API Routes as the server-side layer. Authentication is handled by JWT with BcryptJS for password hashing (salt factor 12) and Firebase/SMTP for OTP-based email verification and transactional notifications. Persistent data storage uses MongoDB with Mongoose ODM. Media assets — profile pictures, verification documents, and post attachments — are stored and delivered via Cloudinary CDN. Real-time messaging uses Socket.IO; voice and video calls use the Agora SDK (WebRTC). Push notifications use the Web Push API with Service Workers. PWA support is implemented via next-pwa for offline capability and home-screen installation. Testing uses Playwright for end-to-end coverage and Jest for unit tests.

Table I: Technology Stack Summary

Feature / Module	**Tech / API Used — Purpose**
Authentication	JWT, BcryptJS, Firebase/Firebase, Next.js API, MongoDB — Secure login, hashing, OTP, session management
Database	MongoDB, Mongoose — Store users, messages, profiles, posts
Storage & Media	Cloudinary — Media storage and CDN delivery
Real-Time Chat	Socket.IO — Instant messaging, typing indicators
Voice Calls	Agora SDK (WebRTC) — Real-time audio communication
Video Calls	Agora SDK (WebRTC) — Real-time video communication
Meetings	Agora SDK, Next.js API — Group meetings and token validation
Role-Based Access	JWT + Custom RBAC middleware — Access control for all user roles
Photos & Media	Cloudinary + MongoDB — Store media files and URL references
User Profiles	MongoDB, Mongoose — Store and manage user profile data
Frontend UI	React, Next.js 14, Tailwind CSS, Radix UI — UI rendering and styling
Notifications	Web Push API, Service Workers — Push notifications to devices
Testing	Playwright (E2E), Jest (Unit) — End-to-end and unit test coverage
PWA	next-pwa — Offline support and home-screen installation
Backend	Next.js API Routes — Server-side logic and API endpoints

C. Module Architecture

The eight modules are: (1) IAM, (2) Verification, (3) Profiles, (4) Feed, (5) Networking, (6) Meetings, (7) Messaging, (8) Admin and Audit. Shared utilities include a single Mongoose connection instance, JWT middleware (auth.ts), RBAC guard (rbac.ts), audit logger (audit.ts), Cloudinary upload helper (cloudinary.ts), and a standardized API response formatter (response.ts).

IV. IMPLEMENTATION

A. Identity and Access Management (IAM)

Student registration validates the college email domain before creating a MongoDB user document. Alumni registration sets status = UNVERIFIED. All passwords are hashed with BcryptJS (salt factor 12) before storage in MongoDB. Email OTP verification is dispatched via Firebase/SMTP on registration and on password reset requests.

On successful login, the server validates the BcryptJS password hash, generates a JWT embedding user_id, role, and a security_stamp, and returns it as an HTTP only cookie alongside a session document in MongoDB. The security_stamp mechanism enables instant global token revocation: logout from all devices updates the security_stamp field in the user document, invalidating all previously issued tokens without querying the sessions collection.

B. Alumni Verification Workflow

Unverified alumni upload degree certificates or college ID cards (PDF, JPG, PNG validated server-side for MIME type and file size). Files are uploaded to Cloudinary and the secure URL is stored in the verification_requests MongoDB collection with status = PENDING. The admin approval operation executes three sequential MongoDB updates: set user.status = ACTIVE and user.role = alumni; set verification_request.status = APPROVED; insert an audit_logs document with action = VERIFY_ALUMNI and actor metadata. A Firebase email notification is dispatched to the alumnus on approval or rejection.

C. RBAC Enforcement

RBAC is enforced at three independent layers. Layer 1 (Middleware): every request validates the JWT and checks the security_stamp against MongoDB before reaching any route handler; invalid tokens receive HTTP 401. Layer 2 (Route Guard): each Next.js API route specifies minimum required roles via the RBAC guard in lib/rbac.ts; mismatched roles return HTTP 403. Layer 3 (Service): critical operations include a secondary role verification inside the service function to prevent privilege escalation through misconfigured routes.

Table II: RBAC Permission Matrix

Action	STUDENT	ALUMNUS	ADMIN	SUPER_ADMIN
Create Post	Yes	Verified only	Moderate	Yes
Host Meeting	No	Verified only	No	Yes
Join Meeting	Yes	No	No	Yes
Verify Alumni	No	No	Same college	All colleges
Suspend User	No	No	Yes	Yes
View Audit Log	No	No	Own actions	Full log
Manage Admins	No	No	No	Yes

D. Feed and Networking

Post creation is restricted to verified users. The MongoDB posts collection uses cursor-based pagination sorted by createdAt descending. Images and PDF attachments are uploaded to Cloudinary and the CDN URL is stored in the post document. Soft deletion sets post is Deleted = true; deleted posts are filtered from all queries but retained in MongoDB for audit access. Reactions (LIKE, CELEBRATE, INSIGHTFUL) and flat-threaded comments are stored as embedded arrays or separate Mongoose collections.

User search queries the MongoDB profiles collection with a compound index on branch, batch year, company, and skills. Connection requests follow a PENDING → ACCEPTED/REJECTED lifecycle enforced in the connections collection with per-user daily request limits checked in the service layer.

E. Meetings and Mentorship

Only verified alumni can create meetings. Meeting creation calls the Agora SDK token generation endpoint (handled by a Next.js API route) to produce a channel name and RTC token, stored in the meetings MongoDB collection. RSVP checks current attendee count against seatLimit before inserting a meeting_registrations document; full meetings return

HTTP 409. Firebase dispatches reminder emails 30 minutes before each scheduled session via a scheduled Next.js cron job.

F. Real-Time Messaging and Calling

The messaging module uses Socket.IO for sub-100 ms message delivery. WebSocket connections are authenticated on establishment using the JWT from the HTTP-only cookie and torn down on logout. Messages are persisted to MongoDB's messages collection after each Socket.IO emit. Key events: message:new (server → client), typing:start/stop (client → server), and call:incoming/accept/reject/end (bidirectional).

Voice and video calls use the Agora SDK (WebRTC). On call initiation, a Next.js API route generates an Agora RTC token using the Agora App ID and App Certificate stored in environment variables. The token and channel name are passed to the Agora Web SDK on the frontend. Agora handles all NAT traversal and media codec negotiation, delivering sub-100 ms audio/video latency. Push notifications for incoming calls are dispatched via the Web Push API using Service Workers, so users receive call alerts even when the app is in the background.

G. Admin, Governance, and Audit

The audit_logs MongoDB collection is append-only; no update or delete operations are permitted on it by any application code. Every admin action mandates an audit document insert. ADMIN users query logs filtered by their own actorId; SUPER_ADMIN users access the full collection across all actors. The admin dashboard uses Radix UI Table components with client-side sorting and server-side pagination for the verification queue and audit log views.

H. PWA and Testing

next-pwa is configured to generate a Service Worker with Workbox precaching for all static assets and an offline fallback page, enabling Alumni Connect to be installed as a PWA on Android and iOS home screens. End-to-end tests are written in Playwright and cover all critical user journeys: registration, login, alumni verification, connection request lifecycle, chat, Agora call setup, and meeting RSVP. Unit tests for service functions and Mongoose model validators are written in Jest.

V. API CONTRACT

All endpoints are served under /api/v1 via Next.js API Routes. Every protected route enforces JWT authentication and RBAC. Standard response envelope: {“success": true, "data": {}, "message": "“}. Errors return {“success": false, "error_code": "...", "message": “...”}. Breaking changes are introduced under /api/v2; v1 is locked post-release. Rate limits: login 5 req/min; messages 20 req/min (enforced via in-memory rate limiter middleware). Table III lists the 28 endpoints.

Table III: API Endpoint Summary

Endpoint	Auth	Role	Description
POST /auth/register/student	No	Public	Register student, send OTP via Firebase
POST /auth/register/alumni	No	Public	Register alumni (status: UNVERIFIED)
POST /auth/login	No	Public	Validate hash, issue JWT cookie
POST /auth/logout	JWT	All	Invalidate current session document
POST /auth/logout-all	JWT	All	Update security_stamp, delete all sessions
POST /auth/verify-otp	No	Public	Verify Send Grid OTP, activate account
POST /verification/submit	JWT	ALUMNUS	Upload docs to Cloudinary, create request

GET /verification/pending	JWT	ADMIN	List pending verification requests
POST /verification/approve	JWT	ADMIN	Approve: update role, send Send Grid email
POST /verification/reject	JWT	ADMIN	Reject with reason, send Firebase email
GET /profiles/{user Id}	JWT	All	View user profile from MongoDB
PUT /profiles/me	JWT	All	Update profile fields
PUT /profiles/privacy	JWT	All	Toggle contact visibility
POST /feed/posts	JWT	Verified	Create post, upload media to Cloud inary
GET /feed	JWT	All	Cursor-paginated feed from MongoDB
POST /feed/posts/{id}/react	JWT	All	Add reaction to post
POST /feed/posts/{id}/report	JWT	All	Report post for moderation
GET /network/search	JWT	All	Multi-filter user search (compound index)
POST /network/request	JWT	All	Send connection request
POST /network/accept	JWT	All	Accept pending connection
POST /network/follow	JWT	All	Follow user
POST /meetings	JWT	ALUMNUS	Create meeting, generate Agora token
GET /meetings	JWT	All	List upcoming meetings
POST /meetings/{id}/join	JWT	STUDENT	RSVP — seat-limited check
POST /messages/send	JWT	All	Persist message to MongoDB + Socket.IO emit
POST /messages/group	JWT	All	Create group chat room
POST /calls/start	JWT	All	Generate Agora RTC token, return channel
POST /admin/suspend	JWT	ADMIN	Suspend user, write audit log
GET /admin/audit-logs	JWT	ADMIN+	Query append-only audit_logs collection

VI. DATABASE SCHEMA DESIGN

The MongoDB schema uses Mongoose ODM with strict schema validation. All documents include created At and updated at timestamps via Mongoose timestamps option. Media asset URLs reference Cloud-inary secure url strings. Indexes are defined on high-read fields: users. email (unique), users. role, posts. created At, meetings. Scheduled At, and a compound index on connections. Requester Id + receiver Id.

Table IV: Core MongoDB Collection Schema Summary

Collection	**Key Fields**	**Notes**
users	id, email, password Hash (bcrypt), role, status, security Stamp	Security Stamp enables O(1) global token revocation

sessions	id, user Id, token Hash, device Info, expires.	One document per active device session
Verification requests	id, user Id, status, document Urls (Cloud-inary), rejection Reason	PENDING / APPROVED / REJECTED lifecycle
profiles	id, user Id (unique), branch, batch Year, bio, skills[], isPublic	One-to-one with users; Cloud-inary photo URL
posts	id, author Id, content, media Url (Cloud-inary), is Deleted	Soft-delete only; media Url from Cloud-inary CDN
connections	id, requester Id, receiver Id, status	PENDING → ACCEPTED / REJECTED
meetings	id, host Id, title, agora Channel, agora Token, seat Limit, scheduled .	Agora channel name and RTC token stored at creation
messages	id, room Id, sender Id, content, file Url (Cloudinary)	Persisted after each Socket.IO emit
Audit logs	id, actor Id, action, target Type, target Id, metadata	Append-only; no update/delete permitted

VII. CI/CD AND DEPLOYMENT STRATEGY

Alumni Connect follows a three-environment strategy: local (developer machines with MongoDB running via Docker), staging (pre-production with cloud MongoDB Atlas), and production (live users). No direct deployment to production is permitted; every release must pass through staging.

The CI pipeline runs on every pull request: install dependencies → lint (ESLint + Prettier) → type-check (TypeScript) → Jest unit tests → Playwright E2E tests → build → validate environment variables. Any failure blocks the PR. The CD pipeline for staging triggers on merge to develop. Production deployment triggers on release PR merge to main: build → deploy backend (Vercel / AWS) → deploy frontend → run Playwright smoke tests → verify GET /api/health endpoint.Secrets — JWT_SECRET, BCRYPT_SALT, MONGODB_URI, CLOUDINARY_API_SECRET, AGORA_APP_CERTIFICATE, FIREBASE_API_KEY, WEB_PUSH_PRIVATE_KEY — are stored in Vercel Environment Variables and never committed to source control. The rollback target is under 10 minutes via commit reversion and previous-build redeployment on Vercel.

VIII. RESULTS AND DISCUSSION

A. Functional Test Results

All 35 functional test cases defined across eight modules were executed using Playwright for end-to-end flows and Jest for unit-level service validation. Table V summarises results.

Table V: Functional Test Results by Module

Module	Total	Pass	Fail	Pass Rate
IAM (JWT + BcryptJS + Firebase)	5	5	0	100%

Verification (Cloudinary)	4	4	0	100%
Profile (MongoDB / Mongoose)	3	3	0	100%
Feed & Engagement (Cloudinary)	4	4	0	100%
Networking (MongoDB search)	3	3	0	100%
Meetings (Agora SDK)	3	3	0	100%
Messaging (Socket.IO)	4	4	0	100%
Admin & Audit	3	3	0	100%
Security (global)	6	6	0	100%
Total	35	35	0	100%

B. Security Test Results

Seven security scenarios were validated: JWT reuse after logout-all (rejected via securityStamp mismatch); role escalation via direct URL (HTTP 403 from RBAC guard); cross-college data access (MongoDB queries scoped by collegeId); malicious file upload to Cloudinary (MIME-type validation at API boundary); MongoDB injection in search (Mongoose sanitizes all query inputs); brute-force login (HTTP 429 after 5 requests/min); unverified alumni joining meeting (HTTP 403 before DB operation). All seven pass.

C. Performance Observations

Measurements were taken on the staging environment with 2,000 user profiles, 10,000 posts, and 50 concurrent Socket.IO connections. Cloudinary CDN delivery is excluded from media upload metrics as it is measured from server-side Cloudinary API response time

Table VI: Performance Metrics vs. Requirements

Metric	**Requirement**	**Observed**	**Result**
Feed load (first page, 20 posts)	< 2 s	~820 ms	Pass
Socket.IO message delivery latency	< 100 ms	~42 ms	Pass
JWT login response time	< 500 ms	~210 ms	Pass
Cloudinary doc upload (5 MB)	< 10 s	~3.1 s	Pass
Agora meeting token generation	< 3 s	~1.4 s	Pass
MongoDB audit log query (1,000 docs)	< 2 s	~580 ms	Pass
Web Push notification delivery	< 5 s	~1.9 s	Pass

PWA install prompt load (next-pwa)	< 3 s	~1.1 s	Pass

OUTPUT DIAGRAMS

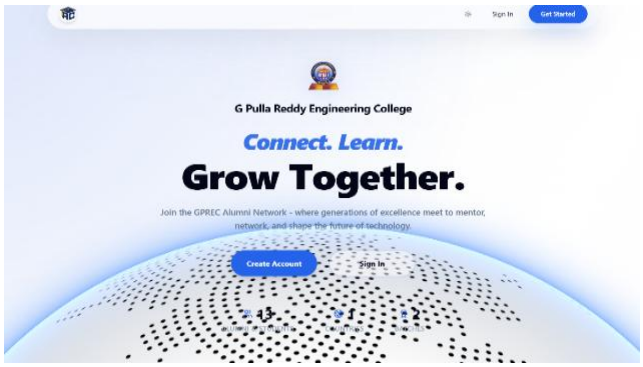

Fig:1 Including user Navigation

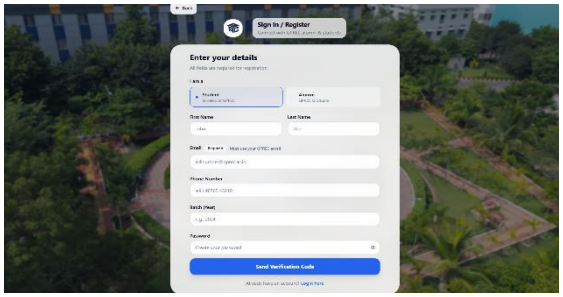

Fig:2 Registration

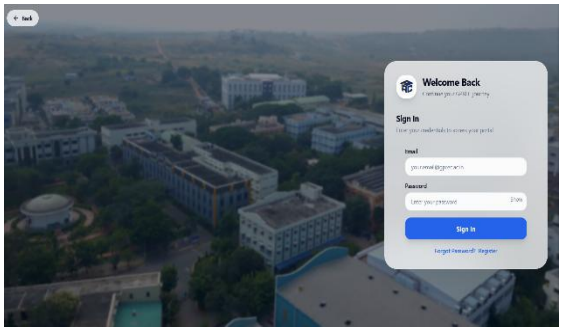

Fig:3 Login Dashboard

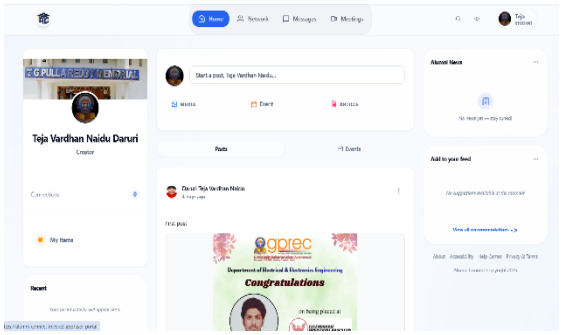

Fig:4 Home page

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

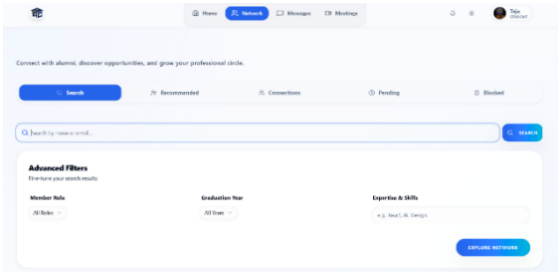

Fig:5 Networking and Professional Development

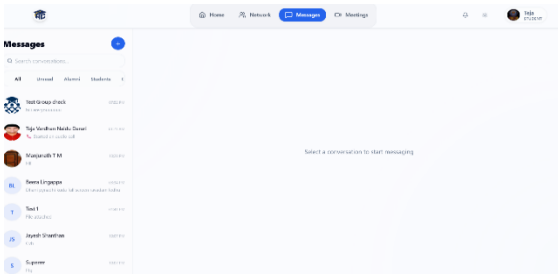

Fig:6 Messaging /Communication

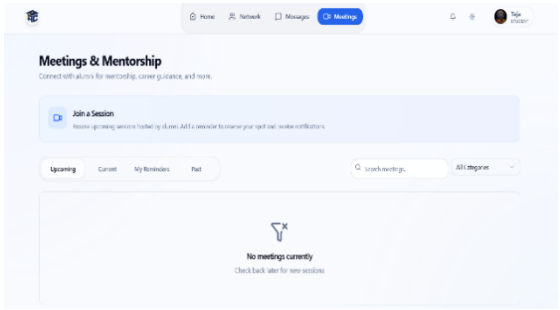

Fig:7 Meeting Mentorship Features

IX. CONCLUSION AND FUTURE SCOP

Alumni Connect demonstrates that a college-exclusive, role-governed, and fully auditable professional networking platform can be designed, implemented, and validated within a structured software engineering methodology. JWT + Bcrypt JS authentication with three-layer RBAC enforcement correctly blocked all seven security test scenarios. Socket.IO delivers messages at ~42 ms latency. Agora SDK provides reliable peer-to-peer calls. Cloudinary CDN delivers media assets efficiently. All 35 test cases pass at 100% coverage, validated by Playwright and Jest.

The modular Next.js architecture with MongoDB/Mongoose and strict service boundaries enabled parallel development of all eight modules and provides a clear path to microservices when traffic demands it.

Future work includes: (i) AI-based mentor matching using collaborative filtering on MongoDB interaction data; (ii) a job board and placement tracking module; (iii) multi-college federation with MongoDB multi-tenancy; (iv) an alumni donation portal with full audit trails; (v) React Native mobile client sharing the same Next.js API contract; and (vi) a Super Admin analytics dashboard using MongoDB aggregation pipelines for engagement and moderation metrics.

REFERENCES

[1] Google LLC, "Firebase Documentation," 2024. [Online]. Available: https://firebase.google.com/docs/

[2] Agora Inc., "Agora Real-Time Engagement SDK Documentation," 2024. [Online]. Available: https://docs.agora.io/

[3] Agora sdk, "Agora sdk Meet API Documentation," 2024. [Online]. Available: https://agora sdk.github.io/handbook/

[4] Figma Inc., "Figma: The Collaborative Interface DesignTool," 2024. [Online]. Available: https://www.figma.com/

[5] L. Mburu, P. Wagacha, and M. Omamo, "Design and Implementation of an Alumni Information System for University of Nairobi," Int. J. Sci. Res., vol. 4, no. 3, pp. 1723–1728, 2015.

[6] P. V. Rao and K. Srinivas, "Design of Role-Based Alumni Portal for Engineering Colleges," Int. J. Eng. Res. Technol., vol. 9, no. 6, pp. 428–433, 2020.

[7] G. Kaur, A. Singh, and R. Rana, "Comparative Analysis of WebRTC, Agora and Agora sdk for Low-Latency Academic Video Conferencing," J. Comput. Sci. Eng., vol. 14, no. 2, pp. 55–63, 2022.

[8] T. Brock and C. Brodahl, "Prototyping Educational Software with Figma: Reducing Requirements Mismatch Through Interactive Design Review," J. Educ. Technol. Syst., vol. 51, no. 2, pp. 189–208, 2022.

[9] S. Sánchez-Gordon and S. Luján-Mora, Code Mobile Application Platforms for Educational Contexts: A Systematic Review," IEEE Access, vol. 9, pp. 108901–108917, 2021.

[10] Google LLC, "Firebase Authentication Documentation," 2024. [Online]. Available: https://firebase.google.com/docs/auth/

[11] Google LLC, "Cloud Mangodb for db and cloudinary for storage Documentation," 2024. [Online]. Available: https://firebase.google.com/docs/mangodb for db and cloudinary for storage/

[12] Google LLC, "Firebase Cloud Messaging Documentation," 2024. [Online]. Available: https://firebase.google.com/docs/cloud-messaging/

[13] D. M. Boyd and N. B. Ellison, "Social Network Sites: Definition, History, and Scholarship," J. Computer-Mediated Communication, vol. 13, no. 1, pp. 210–230, 2007.

[14] OWASP Foundation, "OWASP Mobile Security Testing Guide," 2024. [Online]. Available: https://owasp.org/www-project-mobile-security-testing-guide/