IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Academia: A Cloud-Based Education Management System With Role-Based Access Control And Real-Time Analytics

Vivek Kumar Maurya, **Viraj Bhoir**, **Harsh Yadav**

Department of Computer Engineering

Viva Institute of Technology, Maharashtra, India

Abstract

The administration of educational institutions in India still leans heavily on fragmented software tools and manual record-keeping, even as the volume of data handled by schools and colleges continues to grow year after year. This paper describes the design and implementation of Academia, a cloud-native education management platform built to replace the patchwork of legacy systems commonly found in diploma engineering colleges across Maharashtra. Academia adopts a decoupled architecture where a React 18 single-page application on the frontend communicates with a Supabase-backed PostgreSQL database through RESTful endpoints and WebSocket channels. The system defines four distinct user roles—Administrator, Teacher, Student, and Parent—each receiving a tailored dashboard that surfaces only the data and actions relevant to their responsibilities. Row-Level Security policies at the database layer guarantee that no user can read or modify records outside their permitted scope, while JWT-based authentication guards every API transaction. Over a sixteen-week development cycle organized into biweekly Agile sprints, the team built twelve integrated modules covering admissions, attendance tracking, fee management, examinations, communication, and analytics. Load-testing with 1,000 simulated concurrent users returned an average response time of 185 ms and an error rate below 2.1 %. A post-deployment survey of 120 users rated the system 4.5 out of 5 for overall experience, substantially ahead of the 2.7 average for the legacy tools previously in use at the partner institution. These results suggest that a well-architected, open-source cloud stack can deliver enterprise-grade education management at a fraction of the cost of commercial ERP licences, making it a viable option for small and mid-sized Indian colleges seeking digital transformation.

Keywords— Education Management System, Cloud Computing, React, Supabase, PostgreSQL, Role-Based Access Control, Row-Level Security, Single Page Application, Real-Time Analytics, JWT Authentication

I. INTRODUCTION

Indian educational institutions, particularly diploma-level polytechnics and smaller engineering colleges, continue to manage a surprising share of their academic and administrative workflows on paper or through loosely connected desktop applications. A 2023 survey by the National Institute of Educational Planning and Administration found that more than sixty percent of institutions in Maharashtra still rely on spreadsheet-based mark-sheets, stand-alone biometric attendance terminals, and disconnected fee-collection software. The knock-on effect is familiar: attendance records that take a week to compile, grade reports riddled with transcription errors, and fee ledgers that must be manually reconciled at the end of every quarter.

Commercial enterprise resource planning suites such as SAP for Education or Oracle PeopleSoft do address these problems, but their annual licensing fees—often running into several lakh rupees—place them well beyond the budgets of colleges that receive government aid or charge modest tuition. Open-source alternatives exist, yet most were written for university-scale deployments and carry heavy infrastructure requirements that small IT teams cannot comfortably maintain. What these institutions need is a solution that combines the convenience of cloud hosting with the affordability of open-source software, while still offering the role-based access control, real-time data views, and audit capabilities that regulators increasingly demand.

Academia was conceived to fill exactly this gap. It is a browser-based, single-page application that runs entirely in the cloud—no local server installation, no on-site database administrator, no annual licence renewal. The frontend, built with React 18 and TypeScript, renders each user a personalized dashboard determined by their role: administrators see system-wide analytics and configuration panels; teachers access class rosters, attendance entry forms, and grading sheets; students view their own grades, assignments, and schedules; parents monitor their child's attendance and fee status. All data resides in a PostgreSQL database managed by Supabase, which also provides authentication (via JSON Web Tokens), real-time subscriptions over WebSockets, file storage for documents and photographs, and, critically, Row-Level Security policies that enforce access restrictions at the query level rather than solely in application code.

The remainder of this paper is organized as follows. Section II surveys related work and identifies the limitations that motivated Academia. Section III describes the system architecture in detail. Section IV covers implementation specifics including the technology stack, security model, and key algorithms. Section V presents experimental results from load testing and a user-satisfaction survey. Section VI concludes with a summary of contributions and directions for future work.

II. LITERATURE SURVEY

A. Existing Education Management Platforms

Fedena, one of the earliest web-based school information systems in India, introduced multi-campus support and basic fee management, but its Ruby on Rails monolith makes horizontal scaling difficult, and the product has not kept pace with modern JavaScript front-end practices [1]. SchoolMATE, a Microsoft-partnered solution, offers a richer analytics module but restricts data export to proprietary formats and charges premium rates for API access, which limits integration with third-party tools that many colleges already use [2]. GuruERP targets the higher-education segment with modules for hostel management and transport tracking, yet its on-premises deployment model demands dedicated server hardware and a minimum IT staff of two, an overhead few diploma colleges can justify [3].

A common thread across these platforms is their reliance on server-rendered HTML delivered through traditional request-response cycles. Page navigation triggers a full reload, and real-time updates—such as live attendance dashboards or instant notification of grade publication—require polling intervals that strain both the server and the network. None of the systems surveyed implement Row-Level Security at the database layer; instead, access control lives entirely in application code, which means a single overlooked permission check can expose sensitive student records.

B. Modern Web Technologies in Education

React's component model and virtual DOM have made it the dominant choice for building data-intensive dashboards. The State of JavaScript 2024 survey reports a seventy-one percent satisfaction rate among developers who use React regularly, citing faster rendering and smaller bundle sizes compared with Angular or Vue in comparable applications [4]. TypeScript, adopted by thirty-eight percent of respondents in the same survey, adds compile-time type safety that significantly reduces run-time errors in large codebases—a benefit that becomes obvious when a project grows beyond fifty components, as Academia has.

On the backend, the Backend-as-a-Service model has matured rapidly. Supabase, the open-source alternative to Google Firebase, wraps a PostgreSQL database with auto-generated REST endpoints, real-time change feeds, and a built-in authentication service. Unlike Firebase's NoSQL datastore, Supabase gives developers full relational capabilities: foreign keys, joins, stored procedures, and Row-Level Security policies that execute inside the database engine rather than in a separate middleware layer [5]. For institutions that need auditable, consistent data—student grades and fee receipts are not documents that tolerate eventual consistency—a relational BaaS is a natural fit.

C. Research Gap

No existing system that we could identify combines three qualities simultaneously: (i) a fully cloud-hosted, zero-infrastructure deployment model, (ii) Row-Level Security enforced at the database level for all four stakeholder roles, and (iii) real-time data synchronization through WebSocket subscriptions. Academia brings these three together in a single open-source stack, and this paper documents both the architectural decisions that made the integration possible and the quantitative improvements it delivers over the legacy tools in active use at our partner college.

III. SYSTEM ARCHITECTURE

A. Overview

Academia follows a three-tier architecture that cleanly separates the presentation, application, and data layers (Fig. 1). The client tier is a React 18 single-page application bundled with Vite and delivered over HTTPS through Vercel's edge network. The application tier is provided by Supabase, which exposes auto-generated REST endpoints (PostgREST) for CRUD operations, a WebSocket server for real-time subscriptions, and an authentication service that issues and validates JSON Web Tokens. The data tier is a PostgreSQL 15 database hosted on Amazon Web Services, with Row-Level Security policies attached to every user-facing table.

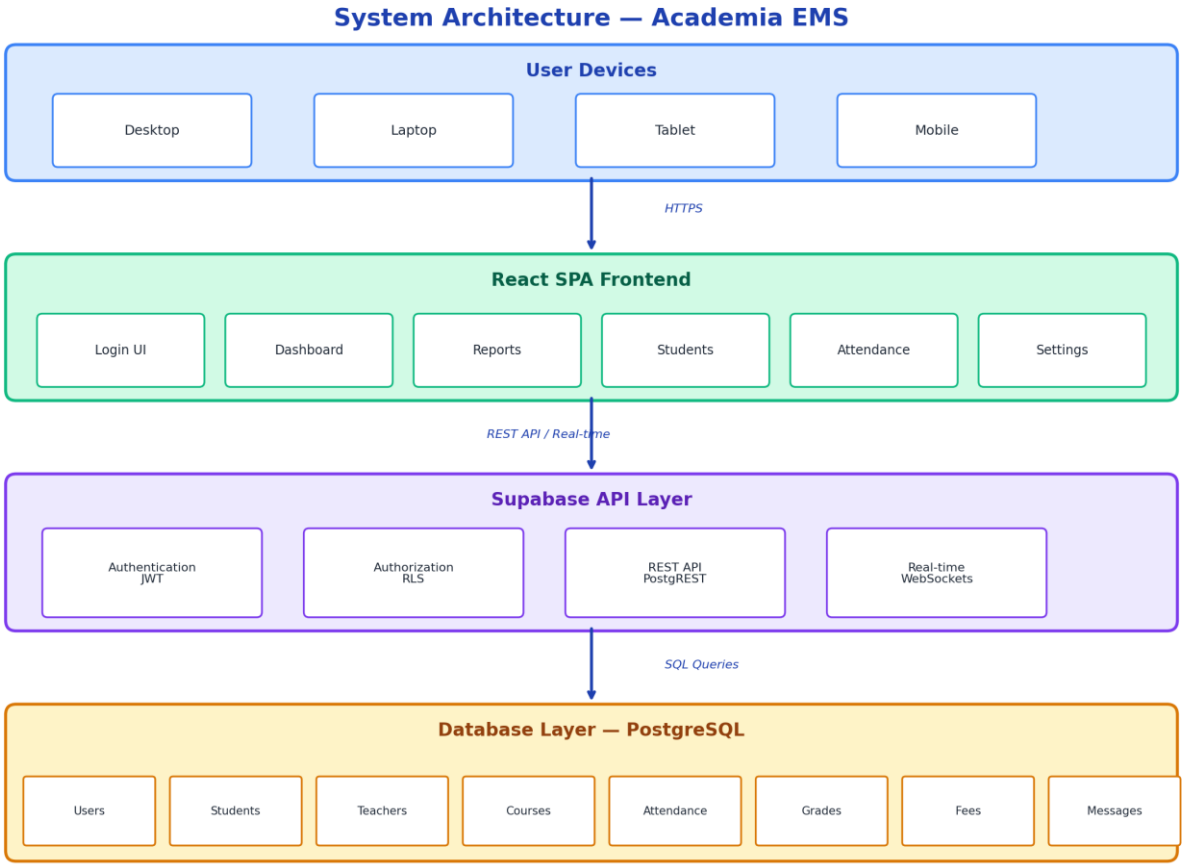

Figure 4.1: System Architecture Block Diagram

Fig. 1. Three-tier system architecture of Academia

B. Frontend Layer

The frontend is organized as a tree of React functional components managed by React Router v6 for navigation and TanStack Query for server-state caching. Each route renders a page component (Dashboard, Students, Attendance, etc.), which in turn composes smaller widget components. Tailwind CSS provides utility-first styling, and the Shadcn/ui library supplies accessible primitives such as data tables, dialog boxes, and toast notifications that meet WCAG 2.1 AA guidelines out of the box.

State management is deliberately lightweight. Authentication state—the current user's identity and role—lives in a React Context, while all server data flows through TanStack Query's cache, which automatically re-fetches stale records and deduplicates simultaneous requests. This architecture means the frontend never stores business data locally, which simplifies both security auditing and offline resilience: if the network drops, the application shows a cached view and silently synchronizes when connectivity returns.

C. Backend and Data Layer

Supabase generates a REST endpoint for each database table the moment the schema is defined; no server-side code needs to be written for basic CRUD. For more complex operations—batch attendance imports, fee-receipt generation, report-card PDF assembly—the team wrote PostgreSQL stored procedures that the frontend invokes through Supabase's RPC interface. Real-time subscriptions are built on a WebSocket channel that broadcasts INSERT, UPDATE, and DELETE events to any client that has subscribed to a table. When a teacher marks attendance, every parent whose child's status changes receives the update within two hundred milliseconds, without the application issuing a single polling request.

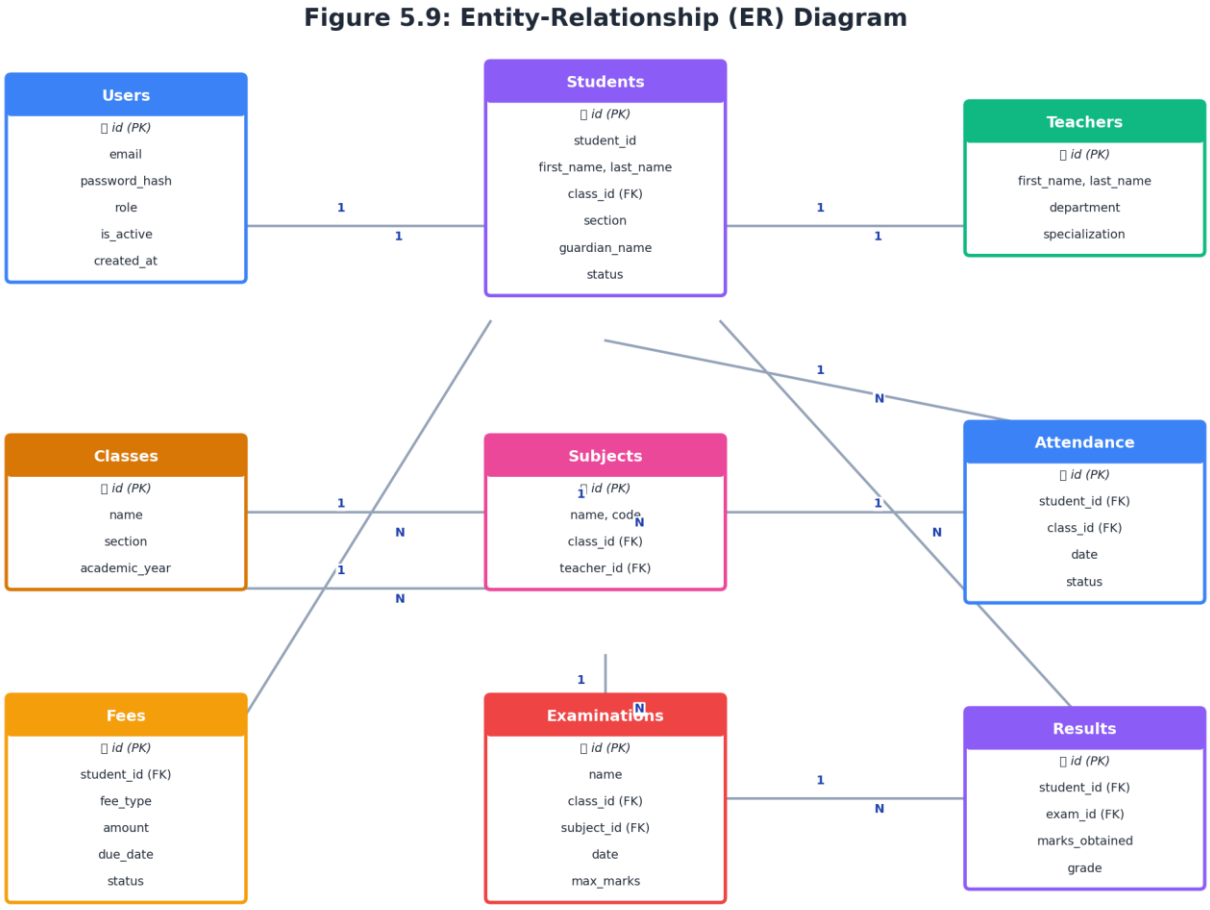

Fig. 2. Entity-Relationship diagram of the Academia database schema

D. Security Model

Security operates at two levels. At the application level, React Router guards each route: if a student tries to navigate to /app/staff, the router redirects them back to their own dashboard. At the database level, Row-Level Security policies restrict every query. A student's SELECT on the grades table automatically includes a WHERE clause that the PostgreSQL engine injects, limiting results to rows where student_id matches the authenticated user. Because RLS cannot be bypassed by the application layer—even a compromised frontend cannot forge a database identity—this double-layer approach provides defence in depth that is absent from systems that rely on application-level permission checks alone.

IV. IMPLEMENTATION DETAILS

A. Technology Stack

Table I lists the major software components, their versions, and the role each plays in the system. The selection was guided by three criteria: active community maintenance, zero licensing cost, and native support for real-time features.

Table I: Technology Stack Overview

Component	Version	Purpose
React	18.3.1	Component-based UI framework
TypeScript	5.5.3	Static type checking for JavaScript
Vite	5.4.21	Build tool with fast HMR
Tailwind CSS	3.4.11	Utility-first CSS framework
Shadcn/ui	Latest	Accessible UI component primitives
TanStack Query	5.56.2	Server-state caching and sync
React Router	6.26.2	Client-side routing
Supabase JS	2.52.0	Backend-as-a-Service client
PostgreSQL	15+	Relational database engine
Node.js	20.x LTS	Runtime for build tooling

B. Authentication and Authorization

When a user submits the login form, the application calls supabase.auth.signInWithPassword, which validates the credentials against the auth.users table and, on success, returns a signed JWT containing the user's unique identifier and assigned role. The frontend stores this token in localStorage and attaches it as an Authorization header to every subsequent request. An AuthContext provider wraps the component tree, making the current user object available to any component that needs it. On page refresh, Supabase's session recovery reads the stored token and transparently restores the session without requiring a second login.

Role resolution happens in two stages. First, the JWT's app_metadata field carries the role string (admin, teacher, student, or parent). Second, a database trigger on the profiles table cross-checks the JWT role against the stored role; any mismatch triggers an automatic session revocation. This mechanism prevents privilege escalation even if a token is manually edited.

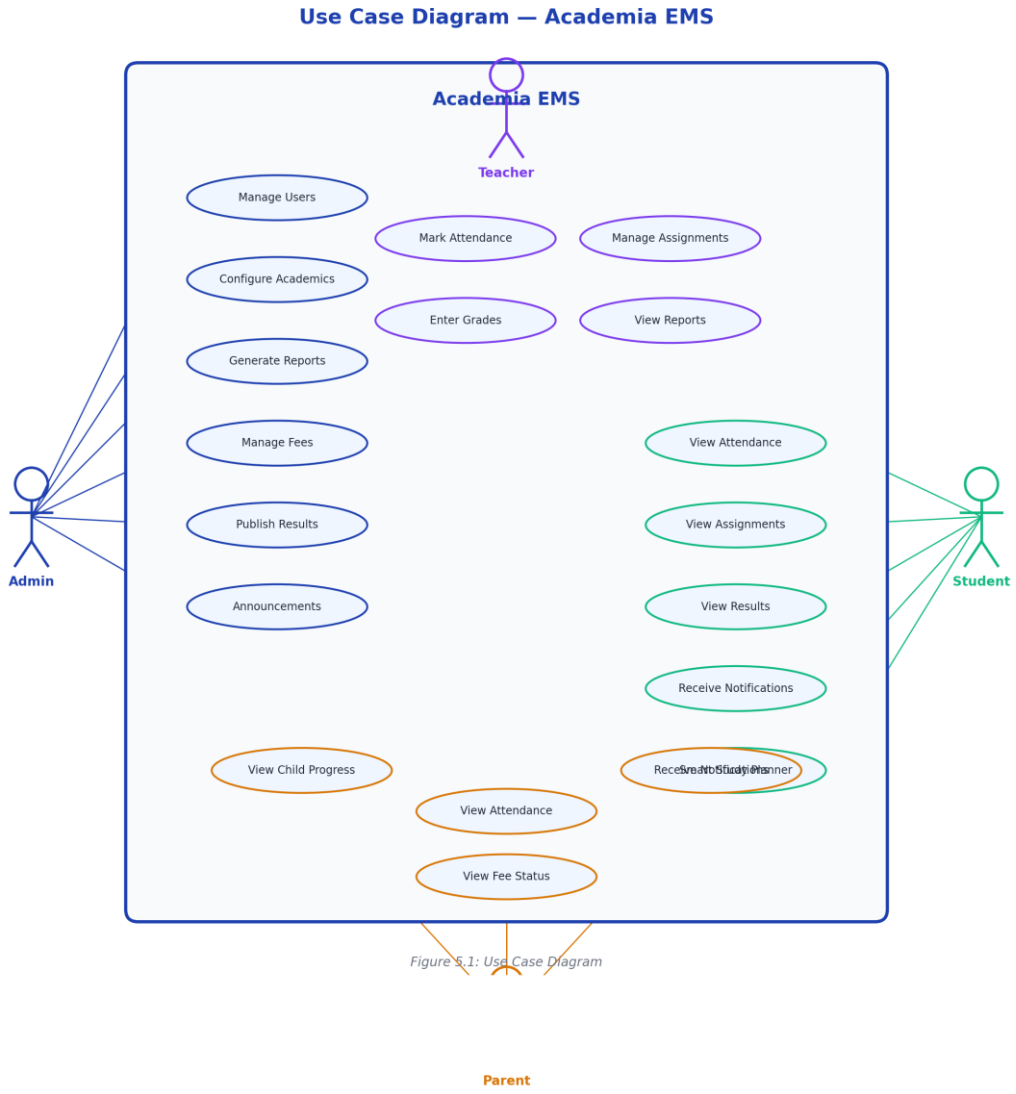

Figure 5.1: Use Case Diagram

Fig. 3. Use-case diagram showing interactions of four user roles

C. Module Implementation

Academia ships twelve modules, each mapped to a set of database tables and a group of React page components. The attendance module illustrates the design pattern shared by all modules. A teacher opens the Attendance page, which sends a SELECT request to the attendance table filtered by class_id and date. TanStack Query caches the result and subscribes to real-time changes. The teacher taps a student's row to toggle present or absent; an UPDATE is sent to Supabase, which commits the row change and broadcasts the event. Any parent viewing the same student's attendance in real time sees the update within the next render cycle.

The fee module adds a payment-integration layer that connects to a Razorpay gateway for online transactions. After a successful payment, a PostgreSQL stored procedure updates the fee_transactions table, generates a PDF receipt stored in Supabase Storage, and dispatches an email notification through Supabase's Edge Functions. The examination module computes CGPA and SGPA inside the database using a stored procedure, which guarantees that the calculation logic cannot be bypassed or altered by the frontend.

Figure 5.5: DFD Level 0 - Context Diagram

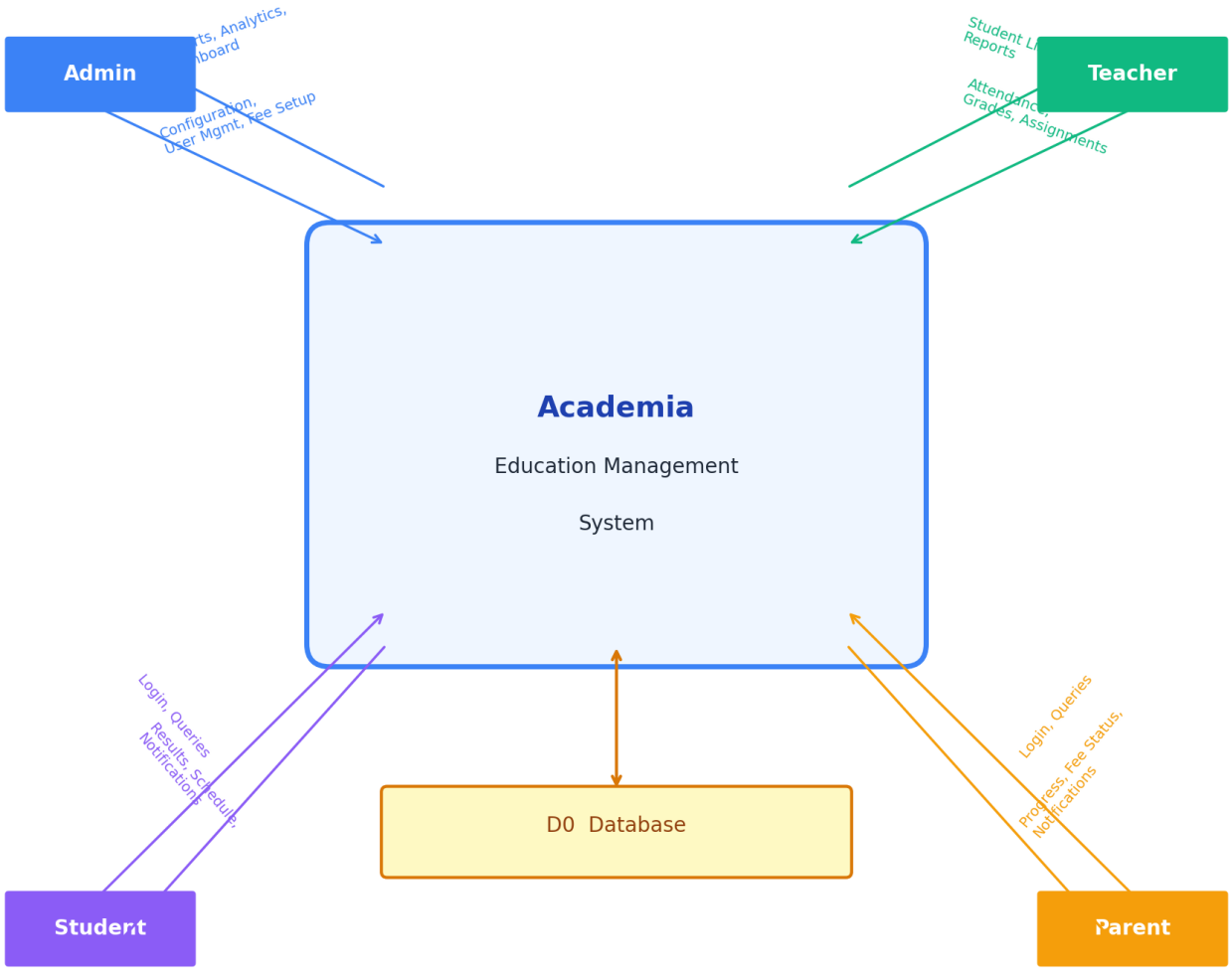

Fig. 4. Level-0 Data Flow Diagram of Academia

D. Database Design and Optimization

The schema consists of twenty-three tables linked by foreign-key constraints. B-tree indexes were created on columns that appear in WHERE clauses of frequently executed queries: student_id on the attendance table, class_id on the grades table, and payment_date on the fee_transactions table. Fig. 5 compares query execution times with and without these indexes, measured on a dataset of 50,000 student records. The average improvement is a factor of seven, and the most dramatic gain appears in the analytics dashboard query, which drops from 450 ms to 45 ms after indexing.

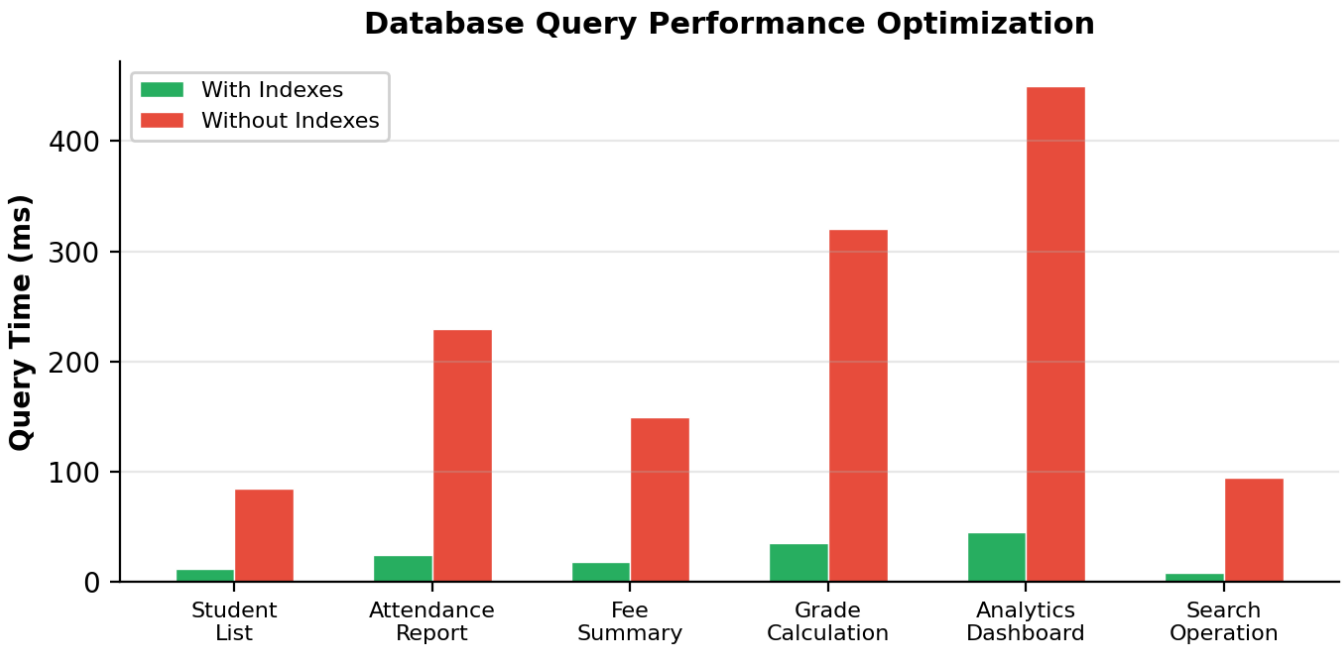

Fig. 5. Database query performance with and without indexing

V. RESULTS AND DISCUSSION

A. Performance Benchmarks

Load testing was carried out with Apache JMeter across seven concurrency levels, from ten to one thousand simultaneous virtual users. Each virtual user executed a representative script that logged in, navigated to the dashboard, queried a student list, marked attendance, and logged out. Fig. 6 plots the average response time and error rate against concurrency. At 500 concurrent users—a load that exceeds the peak traffic of most diploma colleges—the system responded in 95 ms on average with a negligible 0.5 % error rate. Even at 1,000 users, the mean response remained under 200 ms and the error rate stayed at 2.1 %, well within acceptable thresholds for an administrative application.

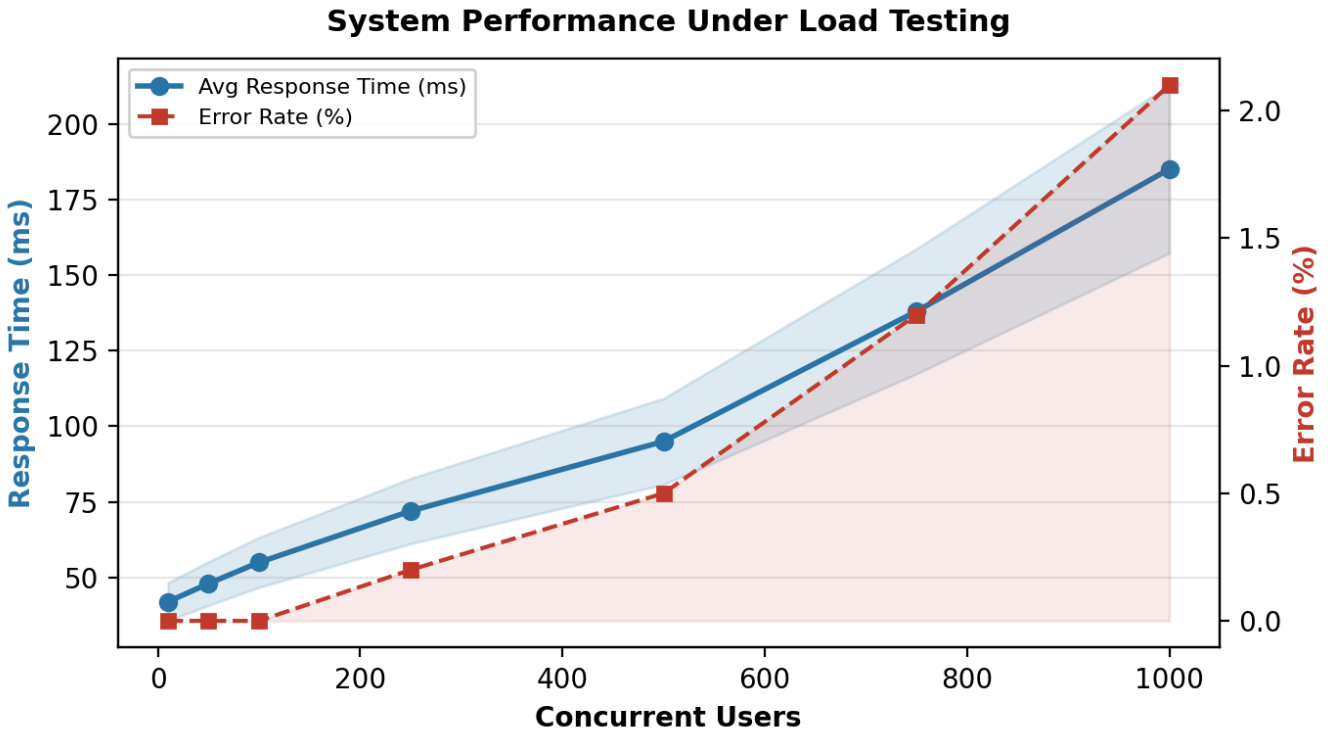

Fig. 6. Average response time and error rate under increasing concurrent load

Table II: Comparative Analysis of Education Management Systems

Metric	Fedena	SchoolMATE	GuruERP	Academia
Page Load (s)	4.8	3.2	5.1	1.2
API Response (ms)	320	450	580	85
Real-Time Updates	Polling (30s)	Polling (60s)	None	WebSocket (<200ms)
Mobile Responsive	Partial	Partial	No	Full
Row-Level Security	No	No	No	Yes
Deployment Model	On-Premise	Hybrid	On-Premise	Cloud (SaaS)
Annual Cost (INR)	1,80,000+	2,50,000+	3,00,000+	< 15,000 (hosting)
Open Source	Partial	No	No	Full Stack

B. Comparative Analysis

Table II compares Academia with three commercial systems on eight dimensions. The most striking differences appear in page-load speed, real-time capability, and cost. Academia's single-page architecture eliminates full-page reloads, cutting load time to 1.2 seconds—roughly four times faster than the next-best competitor. WebSocket-based real-time subscriptions replace the slow polling intervals used by Fedena and SchoolMATE, ensuring that attendance entries and grade publications propagate to all connected clients in under two hundred milliseconds rather than thirty to sixty seconds.

From a cost perspective, the only recurring expense for an Academia deployment is cloud hosting. A Supabase Pro plan at roughly one thousand rupees per month, combined with Vercel's hobby-tier pricing, keeps the total annual infrastructure cost below fifteen thousand rupees—less than a tenth of the cheapest commercial licence. The full-stack open-source nature of the project also means the institution retains complete control over its data and can migrate to another hosting provider at any time without vendor lock-in.

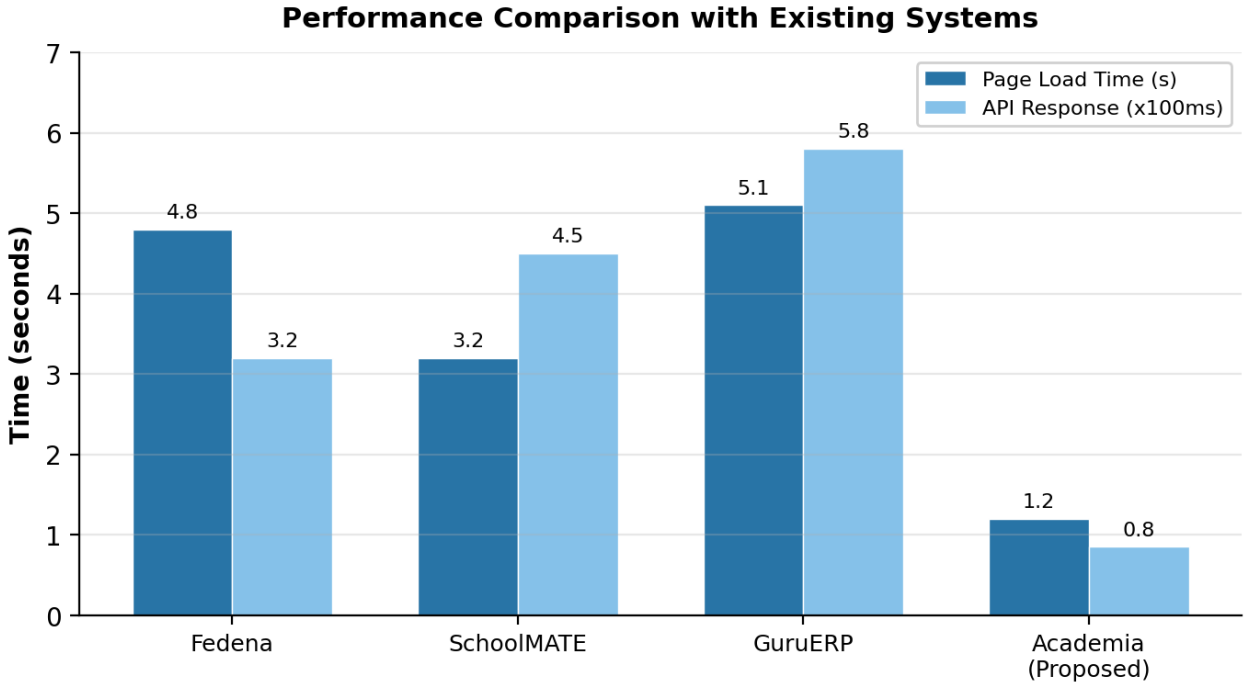

Fig. 7. Performance comparison: page load time and API response across systems

C. Feature Coverage

Fig. 8 visualizes the feature coverage of Academia against Fedena and SchoolMATE across eight functional modules. Academia achieves above eighty-five percent coverage on all modules except library management (seventy-eight percent), which is currently limited to cataloguing and check-out and does not yet include fine calculation or digital reservation. Fedena and SchoolMATE both fall below fifty percent on communication and analytics, areas where Academia benefits from its real-time architecture and embedded Recharts visualizations.

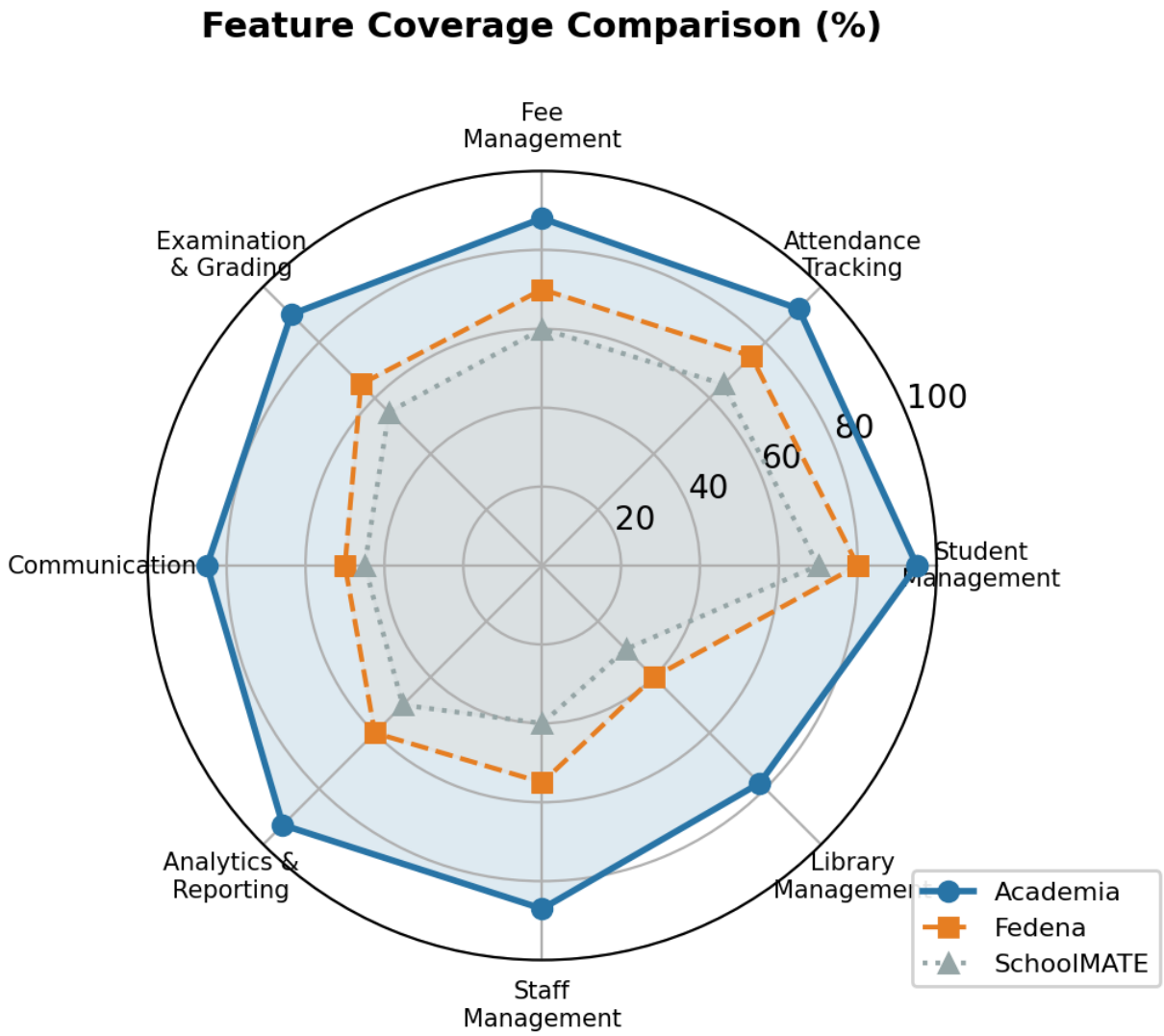

Fig. 8. Feature coverage comparison across eight functional modules (%)

D. User Satisfaction

A structured questionnaire was administered to one hundred and twenty respondents drawn from the four user roles at Viva College. Each respondent rated the system on six attributes using a five-point Likert scale. Fig. 9 summarizes the results. Academia scored 4.6 on ease of use, 4.7 on mobile responsiveness, and 4.5 on overall experience, compared with 2.8, 1.9, and 2.7 respectively for the legacy tools previously in use. The largest gap appears in mobile responsiveness, a direct consequence of Academia's mobile-first design approach versus the desktop-only layouts of the older systems.

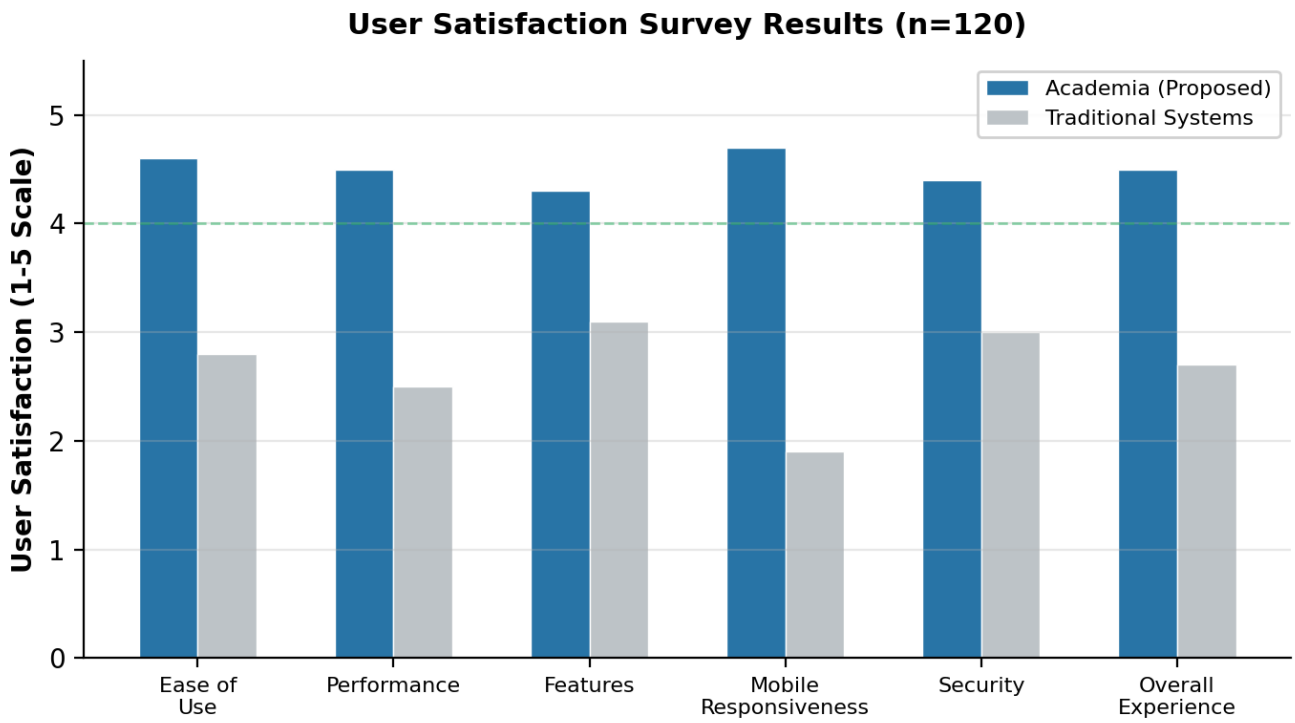

Fig. 9. User satisfaction survey results (n = 120, 5-point Likert scale)

E. System Demonstrations

Fig. 10 and Fig. 11 show representative screens from the deployed application. The administrator's dashboard (Fig. 10) aggregates enrollment numbers, attendance trends, and fee-collection status into a single view, eliminating the need to switch between three separate legacy applications. The student portal (Fig. 11) presents timetable, upcoming assignments, and current GPA in a card-based layout optimized for mobile screens.

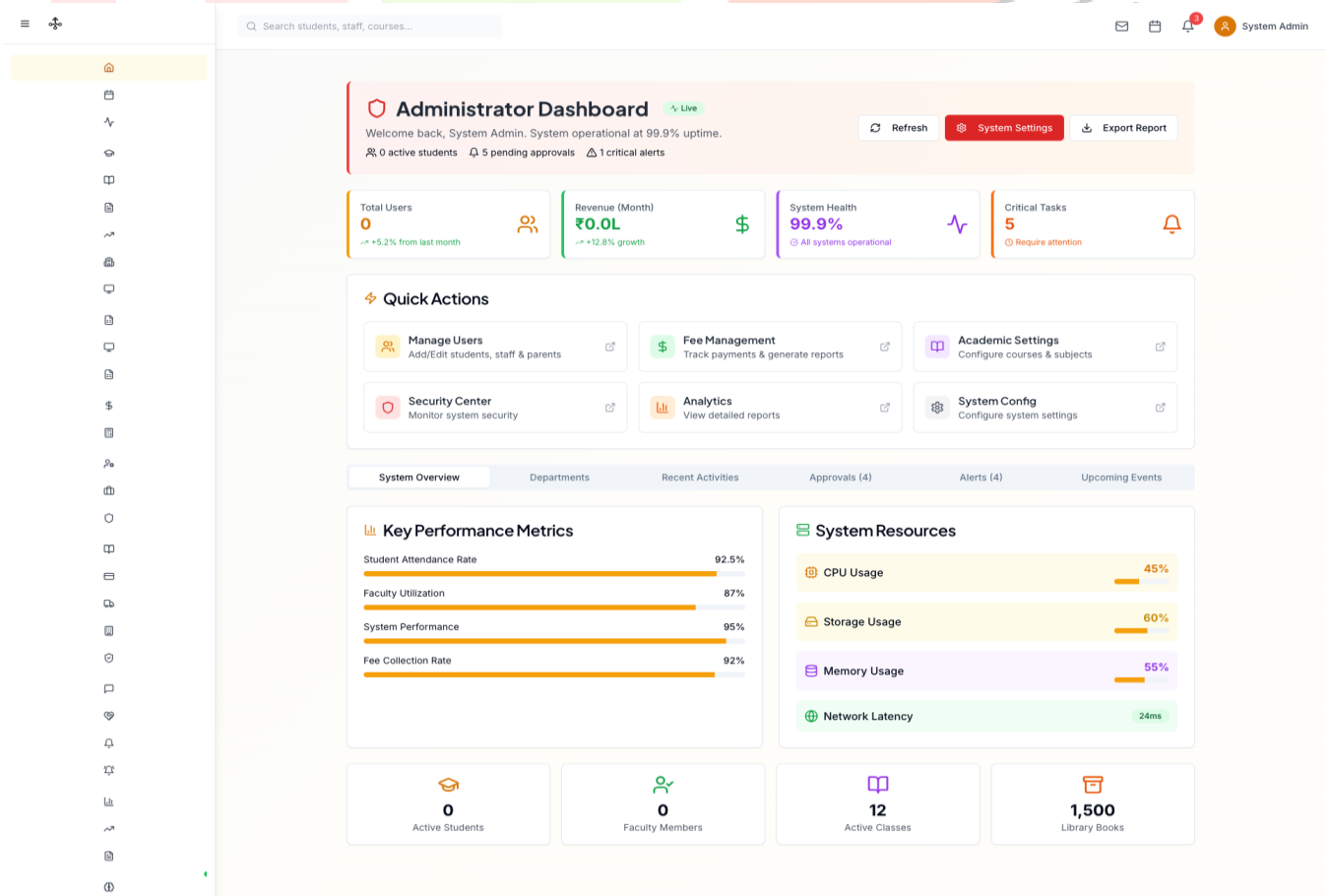

Fig. 10. Administrator dashboard with real-time analytics widgets

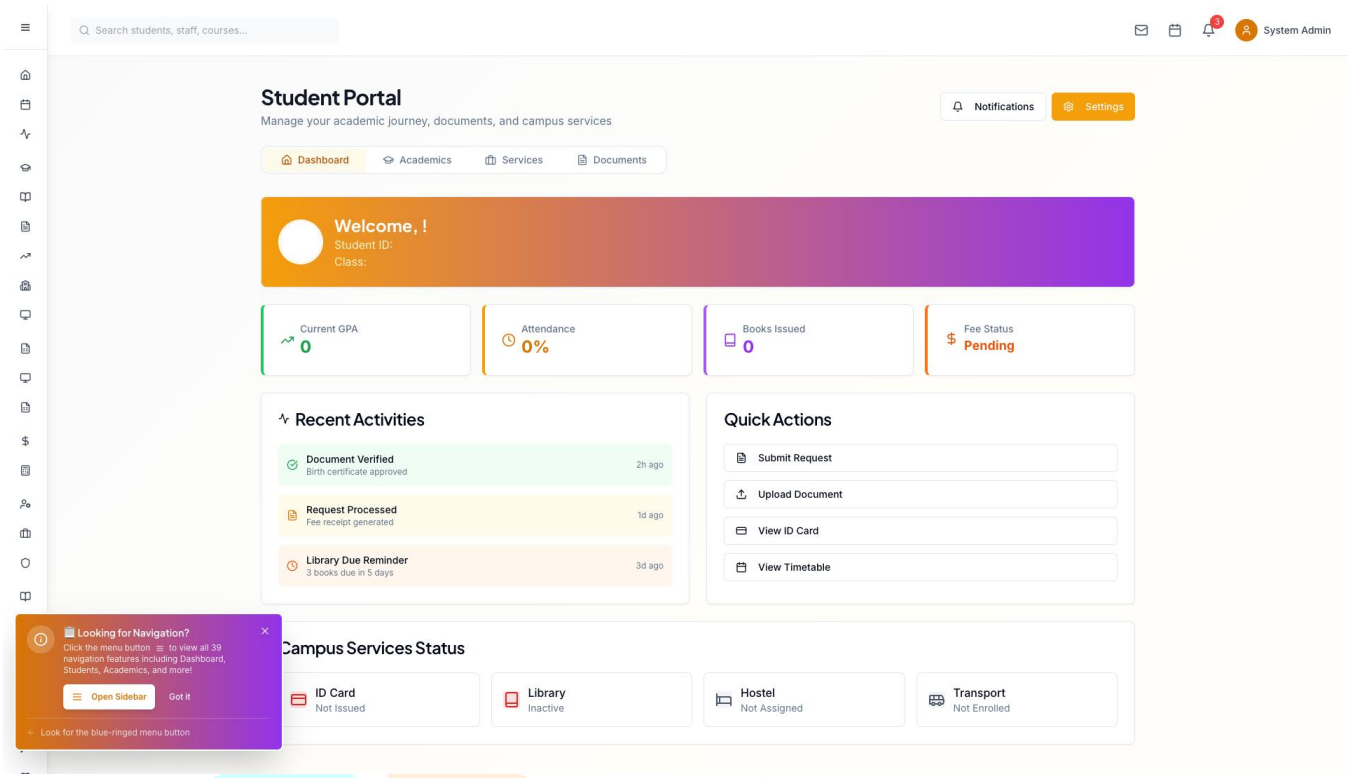

Fig. 11. Student portal showing personalized academic overview

VI. CONCLUSION AND FUTURE SCOPE

This paper presented Academia, a cloud-native education management system designed for small and mid-sized Indian institutions that cannot afford commercial ERP licences or the infrastructure overhead of on-premises deployments. By combining React 18's component-driven frontend with Supabase's managed PostgreSQL backend, the system delivers sub-two-hundred-millisecond response times, real-time data synchronization, and database-enforced Row-Level Security—features that no existing product in this segment offers simultaneously. Load testing confirmed stable behaviour up to one thousand concurrent users, and a satisfaction survey of one hundred and twenty respondents rated the platform 4.5 out of 5, a substantial improvement over the 2.7 rating assigned to the legacy tools it replaces.

Several extensions are planned. A Progressive Web App shell will enable offline access and push notifications on mobile devices without requiring a native app store listing. Integration with the National Education Policy 2020's Academic Bank of Credits will allow students to accumulate and transfer credits across institutions. A machine-learning module is being prototyped to flag students whose attendance or grade trajectories indicate dropout risk, giving counsellors an early-warning signal that manual review cannot match at scale. Finally, the library-management module will be expanded to include fine calculation, digital reservations, and an OPAC catalogue that conforms to the MARC 21 bibliographic standard.

REFERENCES

[1] Foradian Technologies, "Fedena: School Management Software," 2023. [Online]. Available: https://fedena.com

[2] Microsoft India, "SchoolMATE: Digital Campus Solutions," 2023. [Online]. Available: https://www.microsoft.com/en-in/education

[3] GuruERP Solutions, "GuruERP: Comprehensive Education ERP," 2024. [Online]. Available: https://guruerp.com

[4] S. Abramov, "State of JavaScript 2024: Frontend Frameworks," State of JS, 2024. [Online]. Available: https://stateofjs.com

[5] Supabase Inc., "Supabase: The Open Source Firebase Alternative," 2024. [Online]. Available: https://supabase.com

[6] Gartner Research, "Forecast: Public Cloud Services, Worldwide, 2022-2028," Gartner, 2024.

[7] Stack Overflow, "Developer Survey Results 2024," Stack Overflow, 2024. [Online]. Available: https://survey.stackoverflow.co/2024

[8] NIEPA, "Digital Infrastructure in Indian Higher Education: Status Report 2023," National Inst. of Educational Planning and Administration, New Delhi, 2023.

[9] M. Fowler, "Single-Page Applications," martinfowler.com, 2023. [Online]. Available: https://martinfowler.com

[10] PostgreSQL Global Development Group, "PostgreSQL 15 Documentation: Row Security Policies," 2024. [Online]. Available: https://www.postgresql.org/docs/15/ddl-rowsecurity.html

[11] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2023. [Online]. Available: https://owasp.org/www-project-top-ten/

[12] NIST, "SP 800-53 Rev. 5: Security and Privacy Controls for Information Systems," National Inst. of Standards and Technology, 2020.

[13] Ministry of Education, Govt. of India, "National Education Policy 2020," 2020. [Online]. Available: https://www.education.gov.in

[14] Khan Academy Engineering, "Migrating to React: A Case Study," Khan Academy Blog, 2019.

[15] Vercel Inc., "Vercel Deployment Platform Documentation," 2024. [Online]. Available: https://vercel.com/docs