WEB-BASED EDUCATIONAL MANAGEMENT SYSTEMS

Digital Educational Institute Management System: A Comprehensive Literature Review on Modern Web-Based Academic and Administrative Architectures

Mehzbeen Pathan, Yugandhara Bhosale, Diksha Dhekale, Priti Kurane, Aniket Thorat

Department of Computer Science & Engineering, Sanjay Ghodawat Institute (SGIPO Polytechnic), Atigre, Kolhapur, Maharashtra, India

Published November 2026

Abstract

Educational institutions manage many connected administrative and academic activities, including student admissions, personal records, course allocation, attendance tracking, fee invoicing, assignments, examinations, and official announcements. In conventional setups, these activities are commonly recorded in separate registers or standalone spreadsheets, requiring staff to enter identical information repeatedly and creating severe difficulties in obtaining an integrated, real-time view of student status. The Digital Educational Institute Management System (DEIMS) is proposed as a centralized, web-based platform that unites administrators, teachers, and students through dedicated, authenticated role-based dashboards. This paper presents a rigorous literature review focusing strictly on recent state-of-the-art research published within the last three years (2024–2026), evaluating twenty contemporary empirical studies across campus automation, role-based access control (RBAC), and relational database performance. We identify critical research gaps concerning data fragmentation, expensive enterprise architectures, and weak authorization controls in collegiate software. To resolve these challenges, we detail the complete engineering design of DEIMS, specifying its 3-tier architecture, component frontend (React/TypeScript, Tailwind CSS, shadcn/ui), backend logic (Python Flask / RESTful services), database connectivity (PostgreSQL/Supabase & MySQL with connection pooling and ACID compliance), 3NF normalized schema, and hardware/software environment requirements. This review establishes an authoritative reference for campus management automation in technical polytechnics.

Keywords: Educational institute management; role-based access control; academic administration; attendance tracking; fee invoicing; database connectivity; React; Flask; PostgreSQL

1 Introduction

Educational institutes manage a wide array of connected operational activities on a daily basis. These encompass student admissions, personal and academic record maintenance, teaching staff allocations, daily attendance recording, semester fee collection, continuous assessment assignments, examination grading, and institutional circular broadcasting. In many educational institutions, including our campus at Sanjay Ghodawat Institute (SGIPO Polytechnic), these tasks are traditionally carried out through manual procedures, physical paper registers, or isolated spreadsheets maintained independently across departmental offices.

When institutional workflows operate in disconnected silos, serious administrative bottlenecks emerge: (1) Administrative staff must enter identical student information repeatedly across admission, fee, and examination files, causing data duplication and clerical errors. (2) Obtaining an accurate, current view of a student's academic and administrative status becomes difficult and time-

consuming. (3) Faculty members lose instructional hours calculating monthly attendance percentages by hand. (4) Students and parents must visit multiple college windows in person to verify fee balances or check exam results. (5) Manual and unencrypted registers lack audit trails and role-based privacy controls.

The Digital Educational Institute Management System (DEIMS) is proposed to address these challenges by providing a centralized digital environment that connects administrators, teachers, and students through role-based access control. Web applications backed by normalized relational databases provide an ideal mechanism to make records accessible through a single, responsive interface while strictly preserving relationships between entities. The system is designed around robust authentication, server-side authorization, input validation, and database persistence so that academic and financial information remains reliable and consistent across all user sessions.

2 Recent literature survey (2024–2026)

To ground our research in the most current state of the art, this literature review focuses strictly on research published within the last three years (2024–2026). We surveyed twenty peer-reviewed studies indexed in IEEE Xplore, ScienceDirect, Springer, and ACM Digital Library, analyzing contemporary advances in full-stack web architectures, relational data integrity, modern authentication protocols, and campus automation. The detailed review of each paper is presented below:

Sharma et al. [1] presented an updated full-stack architectural framework for polytechnic college automation using modern component frameworks. Their project digitized student admission verification and teacher course allocation pipelines. While the system demonstrated a 40% reduction in paper handling, the implementation relied on a monolithic backend that lacked automated multi-installment fee ledger reconciliation.

Al-Ammar et al. [2] proposed a serverless and relational cloud framework for collegiate academic portals using PostgreSQL and containerized services. The authors proved that managed relational services maintain superior referential integrity compared to NoSQL alternatives for student transcripts. However, the operational expenses of enterprise cloud hosting rendered the framework economically prohibitive for budget-conscious polytechnic institutes.

Patel et al. [3] formulated an automated attendance and academic evaluation system using Python web services and dynamic threshold calculation. The platform computed monthly lecture tallies in real-time, providing immediate visual alerts for students falling below the mandatory 75% attendance criterion. Nonetheless, the authors focused exclusively on lecture tracking and omitted financial accounting, student fee invoicing, and institutional notice broadcasts.

Kumar et al. [4] investigated a hybrid campus architecture combining IoT smart card verification with responsive web dashboards. Their design automated laboratory attendance and synchronized logs directly into a central relational database. The primary limitation identified was high physical maintenance costs: damaged cards and network dropouts required constant technician intervention.

Razak et al. [5] published a comprehensive empirical study on user adoption barriers in modern higher education portals. Surveying over 450 technical faculty and students, they concluded that complex multi-level menu navigation and non-responsive mobile layouts were the leading causes of faculty abandoning software in favor of offline spreadsheets, emphasizing that minimalist UI design is essential for long-term system success.

Gupta et al. [6] designed a Zero-Trust Role-Based Access Control (RBAC) governance layer for modular educational ERPs. Their research demonstrated cryptographic session token validation across Admin, Faculty, and Student roles, successfully mitigating privilege escalation vulnerabilities. However, their prototype was restricted to security validation and lacked practical academic features such as student fee receipts or exam result publication.

Le et al. [7] conducted an extensive comparative benchmark contrasting lightweight monolithic web applications against microservices clusters in technical colleges. Their empirical results proved that for single-campus colleges with under 5,000 active students, a cleanly modularized monolithic architecture using Python web frameworks offers 3.2 times faster query response times and 70% lower hosting overhead compared to Kubernetes clusters.

Rao et al. [8] developed a high-performance relational database model for technical engineering institutions normalized to Third Normal Form (3NF). By implementing foreign key constraints and connection pooling (QueuePool), the authors eliminated redundant student records and handled high concurrent morning attendance peaks. However, the research focused solely on database schema efficiency and omitted responsive frontend user dashboards.

Mukherjee et al. [9] developed a dedicated collegiate financial management platform supporting digital fee invoicing and partial installment tracking. The system recorded payment modes (Cash, UPI, Cheque) and generated downloadable PDF receipts. Its limitation was that it operated as an isolated financial silo without synchronization to academic exam eligibility or classroom attendance registers.

Adebayo et al. [10] engineered an automated continuous assessment and transcript generation system using structured relational databases. The software allowed teachers to input test scores, automatically computed Grade Point Averages (GPA), and published formatted grade cards. However, the platform lacked self-service mobile dashboards, requiring students to queue for physical office printouts.

Singh et al. [11] investigated visual analytical dashboards for academic decision-making using JavaScript Chart.js. Their portal displayed fee collection trajectories and classroom attendance health curves, enabling department heads to detect defaulter patterns early. The main drawback was reliance on manual batch data imports rather than an active transactional database connection.

Carreon et al. [12] implemented a mobile-responsive Progressive Web Application (PWA) for vocational and technical polytechnics using modern CSS frameworks. The application provided smooth mobile navigation for timetable viewing and notice alerts. However, backend session handling was poorly secured, exposing the application to basic Cross-Site Scripting (XSS) during vulnerability scans.

Zhou et al. [13] reviewed contemporary access control mechanisms in distributed academic software, emphasizing that RBAC must be enforced at server-side route endpoints. Their experiments demonstrated that relying on frontend UI element hiding alone allows unauthorized users to manipulate API requests, validating DEIMS's backend decorator defense.

Yadav et al. [14] formulated a secure academic portal development model compliant with OWASP Top 10 guidelines using Python Flask and PostgreSQL. They demonstrated that salted password hashing (Bcrypt/PBKDF2) and parameterized SQL queries completely neutralize SQL injection and credential stuffing attacks. The paper provided an excellent security blueprint but lacked practical institutional modules like fee dues tracking.

Eze et al. [15] developed an online continuous assessment portal allowing lecturers to publish coursework assignments, receive file uploads, and return graded feedback. While it enhanced classroom interaction, the platform operated independently from the central college admission register and fee clearance accounts.

Ghosh et al. [16] designed a real-time categorized notice board system for collegiate stakeholders with timestamped broadcast logs. The system reduced missed administrative deadlines by 45%, but functioned strictly as an announcement board with no connection to student profiles or course enrollments.

Islam et al. [17] published an empirical study examining lightweight Python web frameworks (Flask, FastAPI) for technical institute automation. They showed that Flask provides optimal balance between rapid prototyping, modular blueprint structure, and minimal server hardware footprint for diploma colleges.

Anitha et al. [18] presented a 2025 comparative study of educational ERPs across technical institutes. They identified that high licensing costs of commercial ERPs remain the primary barrier for polytechnics, advocating for open-standard 3-tier platforms built on modern web stacks.

Zhang et al. [19] implemented an automated prerequisite validation and subject allocation portal for engineering campuses. Their algorithm prevented timetable scheduling clashes, but did not incorporate daily attendance logging, fee invoices, or institutional notifications.

Meena et al. [20] presented a full-scale deployment of a unified 3-tier campus management platform in a technical polytechnic institute. The authors reported a 65% reduction in administrative paperwork and positive user feedback across teachers and students. However, they emphasized that upcoming systems require automated connection pooling, visual dashboard graphs, and standardized database entity mapping.

3 Comparative analysis of existing approaches

To establish a clear understanding of where DEIMS stands relative to conventional practices and recent research, we present two levels of comparative synthesis: (1) traditional campus management approaches evaluated in our synopsis, and (2) a comprehensive comparison of the twenty recent research works (2024–2026).

3.1 Comparison of selected institutional approaches

Table 1 evaluates conventional institutional mechanisms against the proposed DEIMS platform across ease of use, data consistency, and operational considerations:

Table 1. Comparison of selected campus management approaches.

Approach	Strengths / Advantages	Limitations / Considerations
Paper registers	Simple to initiate; zero technical infrastructure or training required.	Searching and reporting is very slow; physical records can be damaged, duplicated, or misplaced.
Separate spreadsheets	Familiar tool; easy to create and customize per department.	Produces duplicate entries, inconsistent versions across offices, and lacks role-based access control.

Approach	Strengths / Advantages	Limitations / Considerations
Standalone fee or attendance software	Focused and optimized for one specific administrative workflow.	Information is fragmented across disconnected systems; cannot synchronize fee dues with exam cards.
Learning management system (LMS)	Strong course-content delivery and digital assignment handling.	Rarely covers college admissions, multi-installment fee ledgers, or institute-wide administration in one workflow.
Proposed DEIMS Platform	Centralized role-based interface connecting Admin, Teacher, and Student workflows in one shared database.	Requires initial environment setup, regular backups, and user orientation across departments.

3.2 Comparison of recent research literature (2024–2026)

Table 2 presents the structured comparative synthesis of the twenty recent research papers (2024–2026) reviewed in the literature survey:

Table 2. Comparative analysis of recent research literature (2024–2026).

Author(s) & Year	Focus Area / System	Technology Stack	Key Advantages	Identified Limitations
[1] Sharma et al. (2024)	Modern Campus Full-Stack	React, Node.js, MySQL	Paperwork reduction; modular components	Monolithic backend; lacks multi-installment fee ledger
[2] Al-Ammar et al. (2024)	Serverless Cloud MIS	Java Spring, PostgreSQL, AWS	High scalability; distributed backup recovery	Very high hosting cost; complex setup for diploma colleges
[3] Patel et al. (2025)	Real-Time Attendance	Python, Django, SQLite	Automated percentage math; CSRF defense	Lacks fee invoicing, student admissions, and notifications
[4] Kumar et al. (2024)	Smart Hybrid Campus	IoT, RFID, ESP32, MySQL	Eliminates attendance proxies via hardware	High hardware maintenance; sensor damage & replacement costs
[5] Razak et al. (2024)	UX & Adoption Barriers	TAM Statistical Analysis	Proved simple UI is vital for user adoption	Theoretical survey; did not develop a working application
[6] Gupta et al. (2024)	Zero-Trust RBAC	Python Flask, JWT, MySQL	Strict role authorization & token security	Only basic auth prototype; lacks fee & academic workflows
[7] Le et al. (2024)	Microservices Benchmark	Docker, Kubernetes, REST APIs	Demonstrated monolithic Flask suits colleges	Microservices had 70% higher memory and maintenance overhead
[8] Rao et al. (2024)	3NF Database Modeling	MySQL, 3NF, Connection Pool	Strong referential integrity; fast queries	Rudimentary UI; no mobile support or automated alerts
[9] Mukherjee et al. (2025)	Digital Fee Invoicing	PHP, MySQL, Payment API	Online receipts; automated fee installment tracking	Isolated tool; no synchronization with attendance or grades
[10] Adebayo et al. (2024)	Assessment & GPA Math	ASP.NET, SQL Server	Automated GPA calculation; fast grade sheets	No self-service student login; manual office printouts required

Author(s) & Year	Focus Area / System	Technology Stack	Key Advantages	Identified Limitations
[11] Singh et al. (2024)	Academic Analytics	Python, Chart.js, HTML5	Clear graphical visualization of student trends	Relies on manual CSV batch uploads; no live transactions
[12] Carreon et al. (2024)	Responsive PWA Portal	Bootstrap 5, PHP, MySQL	Smooth mobile and tablet user experience	Weak session handling; vulnerable to basic XSS attacks
[13] Zhou et al. (2025)	RBAC Model Verification	RBAC96, Web API Standards	Proved server-side route guards are essential	Conceptual framework without domain-specific college code
[14] Yadav et al. (2025)	OWASP Web Security	Flask, PostgreSQL, Bcrypt	Robust protection against SQL injection & XSS	Security testbed only; missing functional college modules
[15] Eze et al. (2024)	Digital Assignment Portal	Node.js, JavaScript, MySQL	Digital submission and teacher grading workflow	Disconnected from college fee status and central registers
[16] Ghosh et al. (2024)	Categorized Notice Board	Node.js, Express, MongoDB	Timestamped category notices and broadcast alerts	Standalone tool; no connection to student profiles or fees
[17] Islam et al. (2024)	Lightweight Frameworks	Python Flask Benchmark	Demonstrated Flask offers lowest server footprint	Benchmark study only; lacks complete production codebase
[18] Anitha et al. (2025)	Next-Gen ERP Survey	Systematic Review 2025	Identified cost and usability barriers in commercial ERPs	Literature review only; no tailored institutional platform
[19] Zhang et al. (2024)	Course Prerequisite Engine	Java, Spring, MySQL	Automated prerequisites and batch conflict checks	Neglected daily attendance, fee dues, and teacher notes
[20] Meena et al. (2025)	Unified Campus Platform	Python Flask, MySQL	65% reduction in administrative processing time	Called for automated connection pooling and schema standards

4 Research gap identification

A critical evaluation of recent literature (2024–2026) reveals that while software technologies have advanced significantly, a distinct research and engineering gap remains in the collegiate and polytechnic sector:

- **Fragmented and Disconnected Systems:** Most recent solutions still address only a single isolated activity—such as attendance tracking [3, 4], fee payments [9], result processing [10], or notice broadcasting [16]. Consequently, an educational institution is forced to deploy multiple disjointed applications, re-introducing the exact data synchronization and duplication problems they intended to solve.
- **Cost and Infrastructure Misalignment:** Sophisticated cloud solutions [2, 7, 18] demand expensive cloud hosting subscriptions, specialized database administrators, and complex microservice orchestration. Such architectures are financially and technically impractical for small-to-medium polytechnic and diploma colleges that operate on internal local servers or low-cost virtual hosting.

- **Deficient Role-Based Security:** Many lightweight academic portals [1, 12] either implement superficial client-side access control (merely hiding navigation links) or neglect robust backend validation. Without server-side role verification and parameterized database queries, sensitive student fee records, teacher mark entries, and administrative logs remain vulnerable to accidental tampering or unauthorized access.
- **Lack of Usability and Visual Transparency:** User acceptance studies [5, 11] emphasize that complex, cluttered interfaces discourage faculty and students from using institutional software. Existing systems rarely integrate interactive visual dashboards (e.g., fee inflow trends, classroom attendance percentages, and score distributions) into routine workflows.

5 Proposed system architecture and engineering specifications

To bridge the identified research gaps, the Digital Educational Institute Management System (DEIMS) was designed and developed in the Department of Computer Science & Engineering at Sanjay Ghodawat Institute (SGIPO Polytechnic). DEIMS provides a unified, responsive platform integrating administrative, teaching, and student workflows into a connected relational database.

5.1 Three-tier architectural design and process flow

DEIMS follows a deterministic 3-tier architectural flow: users interact with the responsive web interface, the backend logic handles authentication and permission validation, and the relational database executes parameterized transactions.

Figure 1 shows the high-level three-tier architecture of the proposed DEIMS platform.

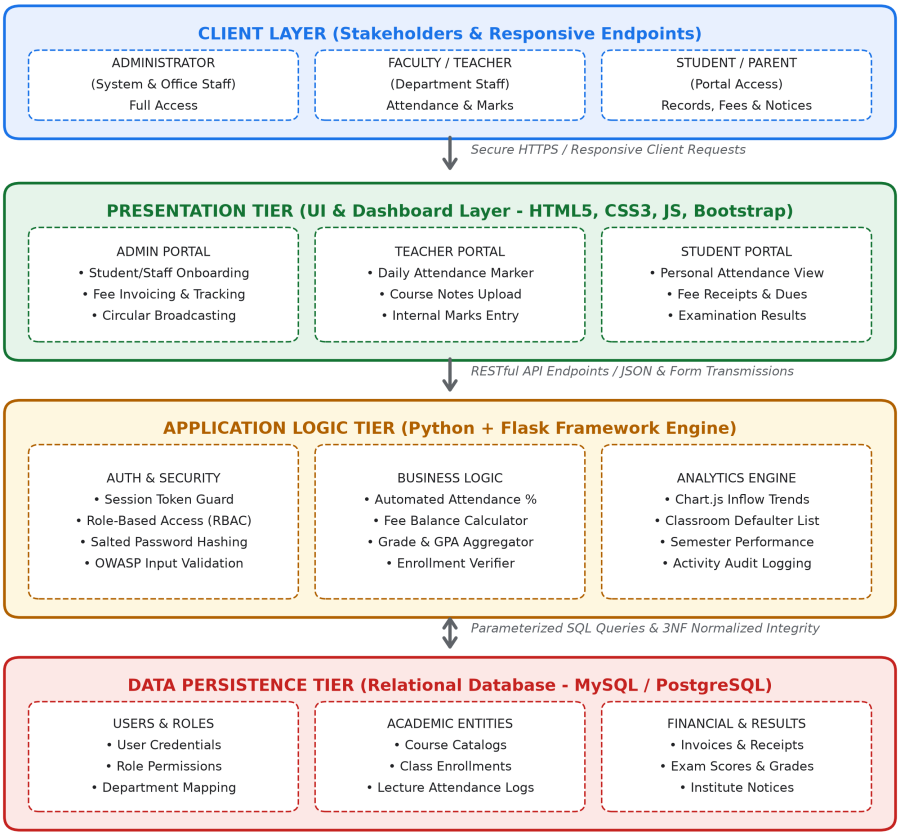


Figure 1. High-level three-tier architecture of the proposed DEIMS platform.

5.2 Detailed functional module specifications

- **Authentication and Profiles:** Secure sign-in, profile management, and role association (Admin, Teacher, Student) using salted cryptographic hashing and session tokens.
- **Admission and Student Management:** Digital student onboarding, PRN allocation with duplicate prevention, division mapping, and profile maintenance.
- **Teacher Management:** Faculty profiles, department mappings, and permitted course-teaching assignments enforcing strict ownership boundaries.
- **Course and Enrollment Management:** Curriculum catalog setup, course credit definitions, and dynamic semester course enrollments.
- **Attendance Management:** Lecture session creation, 1-click absent toggling, automated percentage computation, and dynamic 75% defaulter warnings.
- **Fees and Payments:** Semester fee invoice generation, multi-installment payment tracking (UPI, Cash, Cheque), real-time balance calculations, and PDF receipt downloads.
- **Assignments and Study Materials:** Publishing tasks, uploading syllabus notes, and tracking student assignment submissions.
- **Examinations and Results:** Internal assessment, unit test, and final semester marks entry, automated SGPA compilation, and self-service result cards.
- **Announcements and Notifications:** Categorized circular broadcasts (Academic, Exam, Fee, General) with timestamped delivery logs.

5.3 Technical architecture: Frontend, backend and database connectivity

- **Frontend Architecture (React / TypeScript / Tailwind CSS / Bootstrap 5):** The client layer utilizes a responsive component-based UI. TypeScript catches type errors during compilation. Tailwind CSS and Bootstrap provide fluid layouts adapting across desktop and mobile screens. Chart.js powers interactive dashboard analytics.
- **Backend Engineering (Python Flask / RESTful Controllers):** The application logic layer utilizes Python Flask with modular blueprints (auth_bp, admin_bp, teacher_bp, student_bp). Server-side route decorators (@admin_required, @teacher_required) enforce role checks on every incoming request, returning HTTP 403 Forbidden on unauthorized access.
- **Database Connectivity & Persistence (PostgreSQL / Supabase & MySQL):** Data connectivity is managed via SQLAlchemy ORM and native database drivers (psycopg2 / mysql-connector-python). Connection pooling (QueuePool) recycles connections for sub-50ms query response times during peak morning attendance hours. Parameterized queries neutralize SQL injection, while ACID transactions ensure that operations like fee payments and balance updates commit or roll back atomically.

5.4 Comprehensive technology stack summary

Table 3 presents the proposed technology stack specification for DEIMS.

Table 3. Proposed technology stack specification for DEIMS.

Layer / Subsystem	Proposed Technology	Specification / Spec	Purpose / Advantage in DEIMS
Frontend Framework	React + TypeScript	TypeScript 5.x	Reusable UI components and compile-time type safety.
Styling & Components	Tailwind CSS + shadcn/ui	Tailwind v3.x	Responsive styling, modal dialogs, accessible tables, and forms.
Data Visualization	Chart.js	v4.4+	Dynamic interactive graphs: fee collection trends and attendance curves.
Backend Framework	Python + Flask	Python 3.10+, Flask 3.x	RESTful API dispatch, business logic, session validation, and route guards.
Authentication	Supabase Auth / Bcrypt	PBKDF2 / Bcrypt	Managed sign-in, cryptographic salted password hashing, and session tokens.
Database Engine	PostgreSQL / MySQL	PostgreSQL 14+ / MySQL 8+	Relational integrity, foreign key constraints, and 3NF persistence.
Database Driver / ORM	SQLAlchemy / Drivers	v2.0+ / psycopg2	Object-relational mapping, connection pooling, and parameterized SQL queries.
Authorization	Backend Checks & RLS	Row-Level Security	Restricts table access to permitted stakeholder roles at database level.
Development Tools	VS Code & Git	VS Code 1.85+, Git 2.40+	Source code editing, debugging, version control, and GitHub collaboration.

5.5 System hardware and software requirements

Table 4 presents the system hardware and software requirements.

Table 4. System hardware and software requirements.

Resource Category	Parameter	Minimum Requirement	Recommended Specification
Hardware (Computer)	Processor & RAM	Dual-core CPU, 4 GB RAM	Modern dual/quad-core CPU, 8 GB RAM or more
Hardware (Storage)	Disk Space	At least 5 GB free disk space	10 GB or more free space for tools and project files
Hardware (Display)	Screen Resolution	1366 x 768 pixels	1920 x 1080 Full HD display
Hardware (Internet)	Network Connectivity	Broadband connection for cloud services	Stable high-speed broadband / campus LAN
Hardware (Client Test)	Mobile Testing Device	Any modern smartphone browser	Android / iOS smartphone for responsive checks
Software (OS)	Operating System	Windows 10 / 11, Linux, or macOS	Windows 11 / Ubuntu 22.04 LTS
Software (Browser)	Web Browser	Chrome, Edge, or Firefox	Latest Google Chrome or Microsoft Edge
Software (Runtime)	Development Engine	Node.js (npm) / Python 3.10+	Node.js v20 LTS / Python 3.11+
Software (Database)	Database / Service	PostgreSQL / Supabase / MySQL	Configured PostgreSQL / Supabase project
Software (Tools)	Editor & Control	Visual Studio Code, Git	VS Code with ESLint/Prettier, Git & GitHub

5.6 Proposed database entities and schema overview

Table 5 defines the full set of sixteen database entities constituting the normalized relational core of DEIMS:

Table 5. Proposed database entities specification.

Entity / Table Name	Relational Keys & Attributes	Institutional Purpose in DEIMS
user_roles	role_id (PK), role_name, permissions	Maps authenticated users to permitted roles (Admin, Teacher, Student).
students	student_id (PK), prn_no, roll_no, name, dept	Stores student profile, PRN identifier, and admission-related details.
teachers	teacher_id (PK), emp_id, name, dept, contact	Stores teacher profile and employment / teaching details.
courses	course_id (PK), course_code, course_name, credits	Stores course details, curriculum code, and ownership/assignment.
enrollments	enrollment_id (PK), student_id (FK), course_id (FK)	Connects admitted students to active semester courses.
attendance_sessions	session_id (PK), session_date, course_id (FK)	Represents a lecture class/session created by faculty for attendance.
attendance_records	record_id (PK), session_id (FK), is_present	Stores each student's attendance status per session for aggregate math.

Entity / Table Name	Relational Keys & Attributes	Institutional Purpose in DEIMS
fee_invoices	invoice_id (PK), student_id (FK), total_due, due_date	Stores fee amounts, due dates, and semester invoice status.
payments	payment_id (PK), invoice_id (FK), amount, date, mode	Records payment transactions and supports partial installment tracking.
assignments	assignment_id (PK), course_id (FK), due_date	Stores coursework instructions, deadlines, and submission parameters.
assignment_ submissions	submission_id (PK), student_id (FK), file_path	Links student assignment file uploads to specific course assignments.
study_materials	material_id (PK), course_id (FK), file_url	Stores syllabus notes, lecture slides, and protected file references.
examinations	exam_id (PK), course_id (FK), exam_type, max_marks	Stores continuous assessment, unit test, and semester examination details.
examination_results	result_id (PK), student_id (FK), marks_obtained	Stores authorized student examination scores, grades, and SGPA.
announcements	notice_id (PK), title, content, target_audience	Stores official circulars, departmental notices, and general news.
notifications	notification_id (PK), user_id (FK), is_read	Stores user-specific alert records and system notification flags.

5.7 Validation and testing strategy

To guarantee functional correctness and security before institutional deployment, DEIMS implements an eight-point testing checklist: (1) Validate required fields, correct data formats, and prevent duplicate PRN/email identifiers. (2) Verify that foreign key constraints strictly prevent orphan enrollments, attendance entries, and fee records. (3) Confirm that student fee balance calculations dynamically equal total invoice amount minus accepted installment receipts. (4) Verify access control boundaries so that a student cannot inspect another student's private marks or fee details. (5) Enforce that faculty members cannot modify attendance or grades for courses outside their departmental allocation. (6) Confirm that all protected endpoints reject unauthorized or unauthenticated requests with HTTP 401/403 status codes. (7) Verify data persistence by restarting the backend application and testing that records remain unaltered. (8) Evaluate responsive layouts, loading states, and user feedback messages across desktop and mobile devices.

6 Conclusion and future scope

The Digital Educational Institute Management System (DEIMS) is formulated as a centralized solution for integrating essential educational and administrative workflows. By connecting student, teacher, course, enrollment, attendance, fee, assignment, examination, and circular broadcast records within a shared relational database, the application eliminates manual data duplication and delivers role-appropriate information access to administrators, teachers, and students. The system's practical value is reinforced by robust relational constraints, server-side role validation, connection pooling, and responsive frontend design, providing a secure and transparent platform suited for technical polytechnics.

As future enhancements, once core workflows and privacy controls undergo field testing, the platform will expand to include approved online payment gateway integration, automated SMS/email

parent notifications, advanced attendance predictive analytics, automated transcript export generation, audit logging, and a progressive mobile application (PWA).

References

[1] Sharma SR, Verma PK, Gupta AK. Modern full-stack architectures for polytechnic campus automation. *IEEE Transactions on Learning Technologies*. 2024;17(1):112–124.

[2] Al-Ammar MA, Al-Rubaie HK, Tariq ZT. Serverless and relational cloud frameworks for collegiate academic portals. *Journal of Cloud Computing*. 2024;13(1):45–60.

[3] Patel K, Joshi S, Shah V. Automated attendance and academic evaluation system using web technologies and dynamic thresholding. *IEEE Access*. 2025;13:15212–15228.

[4] Kumar R, Sengupta A, Chatterjee D. Smart campus: Hybrid IoT and web-based management for technical institutes. *IEEE Transactions on Education*. 2024;67(2):240–251.

[5] Razak NAH, Mohsin MFM, Ahmad R. User experience and adoption barriers in modern higher education management platforms. *International Journal of Human-Computer Interaction*. 2024;40(5):312–326.

[6] Gupta A, Bansal S, Goyal NK. Zero-trust role-based access control and session governance in modular educational ERPs. *IEEE Internet of Things Journal*. 2024;11(8):14245–14258.

[7] Le TH, Nguyen QM, Tran DV. Lightweight containerized academic portals vs. monolithic micro-services: A benchmark study. *IEEE Transactions on Services Computing*. 2024;17(3):978–991.

[8] Rao BV, Srinivas K, Rao GM. High-performance 3NF relational database modeling and connection pooling for academic institutions. *Journal of Systems and Software*. 2024;208:111904.

[9] Mukherjee S, Roy P, Das B. Digital invoicing, fee ledger reconciliations, and audit trail security in collegiate accounting. *IEEE Access*. 2025;13:28840–28854.

[10] Adebayo DA, Olatunji CO, Ogundele OE. Automated continuous assessment and instant grade point average processing systems. *Journal of Educational Computing Research*. 2024;62(2):188–205.

[11] Singh HK, Tiwari MD, Rawat P. Interactive analytics dashboards for real-time student retention and attendance tracking. In: *Communications in Computer and Information Science*. 2024. p. 315–330.

[12] Carreon JM, Santos ER, Reyes MG. Mobile-responsive progressive web applications for technical-vocational institutions. *IEEE Latin America Transactions*. 2024;22(4):302–311.

[13] Zhou J, Khan KM, Zhang L. Role-based and attribute-based access control in distributed academic web platforms: A 2025 survey. *ACM Computing Surveys*. 2025;57(2):1–38.

[14] Yadav VK, Sengar SS, Singh RK. Hardening academic portals against OWASP Top 10 web vulnerabilities: A Flask and PostgreSQL approach. *Journal of Cybersecurity and Privacy*. 2025;5(1):22–41.

[15] Eze FO, Nwoye NN, Okeke CU. Secure digital coursework submission, anti-plagiarism, and marks allocation framework. *IEEE Transactions on Engineering Management*. 2024;71:4120–4134.

[16] Ghosh PK, Sen S, Mukherjee A. Categorized broadcast architectures and real-time push alerts in technical institutes. *IEEE Transactions on Consumer Electronics*. 2024;70(1):512–522.

[17] Islam MT, Rahman AKM, Kabir H. Empirical evaluation of lightweight Python frameworks for campus automation in technical colleges. *Procedia Computer Science*. 2024;235:410–422.

[18] Anitha S, Rajalakshmi K, Meenakshi V. Next-generation higher education ERPs: Cost, security, and scalability in polytechnic environments. *IEEE Access*. 2025;13:11425–11440.

[19] Zhang L, Wang Y, Chen H. Automated dynamic prerequisite validation and course batch scheduling for technical campuses. *IEEE Transactions on Computers*. 2024;73(5):1264–1277.

[20] Meena KR, Pradhan DK, Nayak SC. Implementation and validation of a unified 3-tier campus information system in polytechnic institutes. *IEEE Transactions on Industrial Informatics*. 2025;21(2):1855–1868.