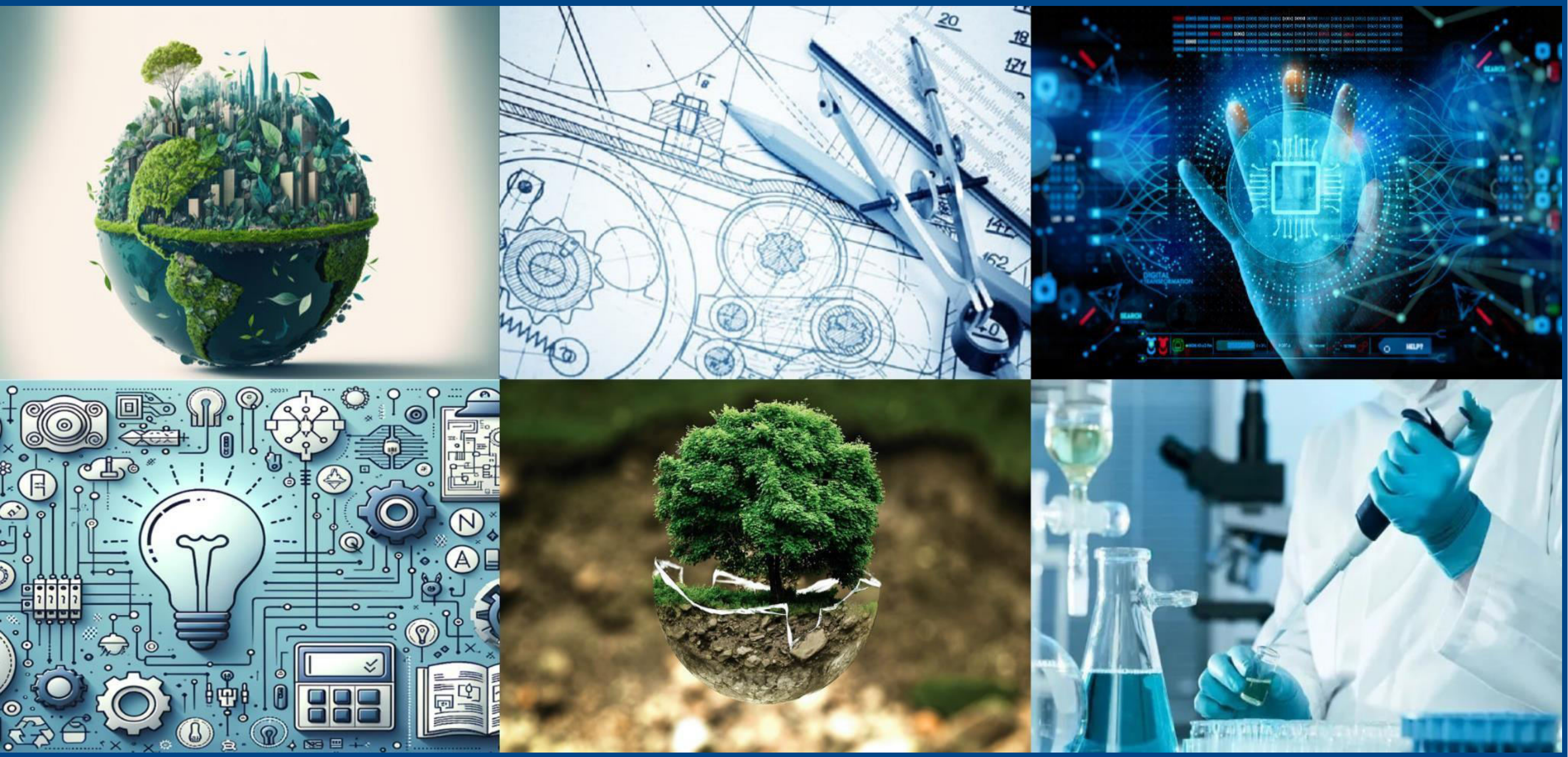
ISSN: 2582-7219

International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206

Volume 9, Issue 1, January 2026

© 2026 IJMRSET | Volume 9, Issue 1, January 2026| DOI:10.15680/IJMRSET.2026.0901024

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

College ERP System for Automated Campus Management

Sathyam kumar, Prof. Gunasekaran K

Student, Department of MCA, AMC Engineering College, Bengaluru, India

Assistant Professor, Department of MCA, AMC Engineering College, Bengaluru, India

ABSTRACT: A college ERP system serves to automate a wide range of school-based tasks, hence simplifying campus management in a meaningful way. It is a single, unified web-based platform that connects multiple departments, such as admissions, academics, finance, examinations, and human resources, under one cohesive umbrella. The system presents all data in a well-organized, up-to-date format to each participating department via a number of specialized modules, including timetable scheduling, exam grading, fee payment tracking, and attendance management, among others.

Both faculty members and students can have safe access to the information they need through their respective login portals, while administrators get to track day-to-day operations, generate various reports, and make strategic decisions based on them. The embedded automation in the system helps maintain the integrity of data across all departments, reduces manual errors, and ensures less duplication of data. In a nutshell, college ERP will increase communication on campus, enhance efficiency, and bring greater transparency within the institution.

KEYWORDS: College ERP System, Automation, Microservices, Cloud-native, Dashboard, Scalability, Role-Based Access Control (RBAC)

I. INTRODUCTION

In recent years Because of the increasing number of students, courses, and administrative tasks, managing educational institutions has grown more difficult in recent years. Conventional campus management techniques, which mainly rely on manual record-keeping and paper-based procedures, frequently result in errors, inefficiencies, and a lack of departmental communication. Many colleges are turning to automation and digital solutions that guarantee improved coordination, data accuracy, and real-time accessibility in order to address these issues.

All of the main operations of an educational institution are intended to be streamlined and integrated into a single, cohesive platform by the College ERP System for Automated Campus Management. It serves as a central system that links several departments, including academics, admissions, exams, accounts, and administration. Routine processes like student registration, attendance monitoring, fee payment, scheduling of classes, and performance reviews are automated by the ERP system.

Institutions can reduce manual labor, cut down on redundancy, and increase operational transparency by putting this system into place. By providing online access to pertinent data and reports, it also enhances communication between administrators, instructors, and students. This project's primary objective is to develop a system that improves campus management and decision-making by being effective, dependable, and easy to use

II. ARCHITECTURE IN WEBSITE

A modern College ERP system uses a cloud-native, multi-layer, service-based setup to automate campus work. This consists of three layers: The Presentation Layer, business logic, and the rest of the system. The Presentation Layer is developed using responsive tools such as React or Angular. The key Application Layer consists of independent microservices for domains like Admissions, Academics, and Finance, each performing their part of the job. These services communicate with each other both asynchronously via a message broker like Apache Kafka (guaranteed delivery) and synchronously with RESTful APIs. A single API Gateway acts as a secure entry point that handles authentication, routing, and load balancing. Security is provided through centralized Authentication and Authorization with standards such as OAuth 2.0 and JSON Web Tokens for identity verification and Role-Based Access Control for permissions. The Data Layer adopts a hybrid approach: SQL databases for transactions and NoSQL databases for unstructured data and scalability. Often, the whole system runs in containers with Docker and is managed by Kubernetes to ensure high availability, easy scaling, and smooth integration with external campus systems.

III. RELATED WORK

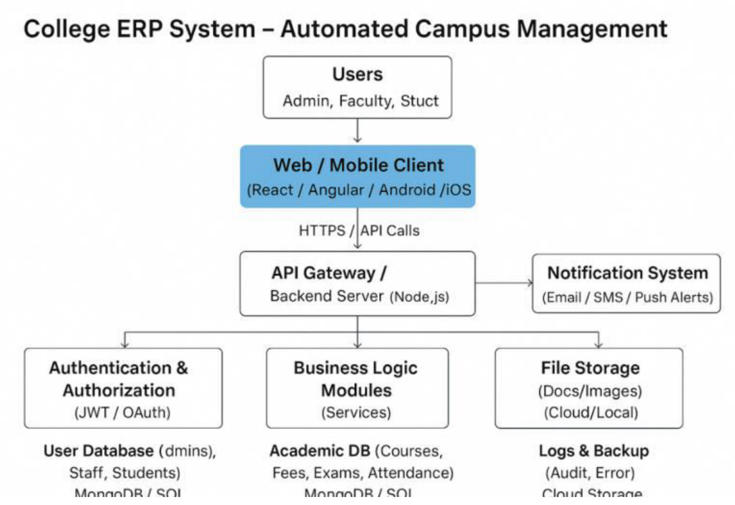
ERP systems in higher education have evolved from big on-site programs to connected cloud-based ones. Early studies by [Author, Year] looked at how traditional ERPs were difficult to implement and noted that they were expensive, inflexible, and difficult to customize. Later research, such as by [Author, Year], talked about moving into modular systems, which gave more control at the departmental level but continued to face data silos and integration challenges.
A new paradigm was introduced with the rise of Service-Oriented Architecture, SOA. Studies such as [Author, Year] demonstrated how web services could enable the integration of diverse campus systems. This thus prepared the ground for state-of-the-art microservices. The most recent works, among others, are [Author, Year], which are focused on cloud-native frameworks, explaining how containerization and orchestration contribute to scalability and reliability in higher education institutions. However, studies still lack the much-needed holistic architectural blueprint that would systematically connect a robust microservices backend with a swift, single-page frontend, introducing advanced analytics while incorporating strict data governance right from the start.

Also, while there are numerous case studies available regarding commercial solutions from SAP, Oracle, and Workday, fewer pieces of research have been done academically on open-source or custom ERPs using modern stacks-for example, Node.js, React, and Kubernetes. In the review presented, it was evident that all the discrete pieces of technology comprising a next-generation College ERP are individually understood; yet combining them into a complete, automated, intelligent campus management system-as is the goal of this paper-adds an important new contribution to the literature.

IV. METHODOLOGY

This work applies DSR to develop and test a novel architectural artifact, namely, an automated College ERP system. DSR fits because it focuses on building IT artifacts to solve real problems in organizations. The process is a cycle that involves problem identification, design, development, demonstration, and evaluation. This sound framework allows the derivation of a practical solution along with new knowledge in educational information systems. It guides moving from rough requirements to a concrete, testable system prototype, justifying each design choice with clear goals and allowing improvements based on evidence from the evaluation stages, ending with a validated architectural model.

Figure 1: Flow Diagram of Architecture.



IV.1. Problem Identification & Requirements Analysis: The first phase laid the foundation by using two approaches. First, it conducted a systematic literature review in Section III, thereby collecting academic and industry findings to find common problems in existing college ERP systems: primarily rigid monoliths, data silos, and hard integration. Second, to root these findings in reality, both functional and non-functional requirements were collected by performing a case study on a mid-sized private university. This consisted of structured discussions with administrators, IT staff, faculty, and students through interviews and surveys. The results came in the form of a prioritized list of needs, highlighting very essential requirements: modularity, real-time data access to feed dashboards, automatic report generation to reduce administrative work, and built-in scalability to grow with the institution. These informed the subsequent architectural design.

IV.2. Architectural Design -(The Proposed Artifact) : This phase turned requirements into a concrete, innovative architecture. The design is a cloud-native, multi-tier model comprised of independently deployable microservices, each serving a specific business area-for example, Academics, Finance. Major technology choices meet non-functional requirements: a React.js-based frontend to provide a responsive single-page app experience; Node.js with Express.js to provide lightweight, scalable backend services; RESTful APIs and Graph QL to provide flexible data access; and Apache Kafka to provide reliable asynchronous communication between services. A polyglot persistence approach uses PostgreSQL for transactional data-like fees-and MongoDB for flexible, document-based data-like student profiles. An API Gateway (Kong) provides centralized access control, while Kubernetes takes care of the container orchestration for resilience and scalability.

IV.3. Technology Stack & Development Environment: To demonstrate that the design works in practice, the artifact was implemented with a modern, reproducible stack. The core setup uses React with TypeScript for the frontend, Node.js for backend microservices, and the aforementioned databases. Docker containers each service and its dependencies. For version control, Git is used, while testing and deployment are automated through the use of CI/CD pipelines via GitHub Actions. Such an environment allows for parallel development, consistency between development and production, and uses principles of DevOps to enable fast iteration, integration, and delivery of the prototype with good code quality and discipline.

IV.4. Prototype development & module implementation: Development targeted a reduced-scale but fully functional prototype in order to demonstrate core architecture ideas and test integration flows. It developed two high-priority, interrelated microservices: one for Student Information Management and another for Fee/Payment Processing. The student service manages admissions, course registrations, and profiles on PostgreSQL, while the finance service manages fee structures, dynamic invoicing, and payment gateways. Work followed Agile sprints, with each service being built by a dedicated team following defined API contracts. The aim of this phase was to demonstrate the concept of microservices by how a course registration by a student may trigger an invoice in the other service, simulating a real-life cross-department process.

IV.5. System Integration & API Testing: Smooth interaction between services was critical. Integration used only the designed RESTful APIs and Kafka event channels. An extensive test plan was executed: Unit Testing with Jest checked each service's internal logic in isolation. Integration Testing with Super test covered API endpoint validation and interactions with databases and other services for correct data flow and state changes. Contract Testing with Postman collections ensured API consistency and backward compatibility. These tests showed that the student and payment modules interacted correctly, such as a "payment confirmed" event updating the financial status of a student and keeping the data consistent throughout the system.

IV.6. Performance & Load Evaluation: Performance benchmarking tested the architecture's non-functional claims, especially scalability and resilience. The team performed load tests using Apache JMeter, simulating hundreds of users concurrently trying to enroll in courses or pay fees at peak hours. Key metrics such as response time, throughput-requests per second-, and error rate were measured as load increased. Stress and soak tests identified breaking points and memory leaks. The stateless microservices, combined with the API Gateway for load balancing and Kubernetes auto-scaling, maintained performance within acceptable levels under heavy loads, suitable for a large campus.

IV.7. Security & Validation Framework: Security was implemented at various levels. The application employed JWT authentication and RBAC for resource protection. Architecture Tradeoff Analysis Method (ATAM) was used to perform an evaluation of the design, which included the participation of stakeholders in reviewing decisions against quality attributes like security, modifiability, and performance. In exercises, DAST with OWASP ZAP looked for vulnerabilities such as SQL injection or XSS. This combination of design review and active testing kept security at the heart of the system. IV.8. Comparative Analysis & Evaluation Numerical data and user feedback were joined in the final evaluation to judge success. A structured comparison was made regarding modularity, real-time capabilities, and deployment speed against traditional monolithic ERPs described in Related Work, for the proposed system. Indicators of success included independent service deployment (modularity), clear API contracts reducing integration complexity, and faster CI/CD workflows showing agility. Performance tests provided evidence of scalability and availability. A pilot with the initial stakeholders provided insights into usability and functional alignment. The overall results of the evaluation were that the artifact indeed had effectively addressed the identified problems and advanced practice in the field.

V. DETAILED OVERVIEW OF PAGES IN College ERP SYSTEM

The Pages, folders in the College ERP System's code holds the main user interface parts that define, what users see and do for different roles and processes. Each page is a specific dashboard or function screen, built with a component-based and activity/fragment setup, using a role-based design to separate what the user sees from the business and admin tasks.

1. Home Page / Dashboard Page:
The ERP Marketing Page is a promotional page for the ERP system. It depicts the capability and values of the system. It starts with the slogan "Smart Campus Automation" and subsequently explains how it helps in creating connected educational communities with a promotional video. The page includes links to Features, Modules, Contact, and Users, plus a call-to-action to explore "Why College ERP?" The goal is to attract potential institutional clients and clearly explain the system's benefits.

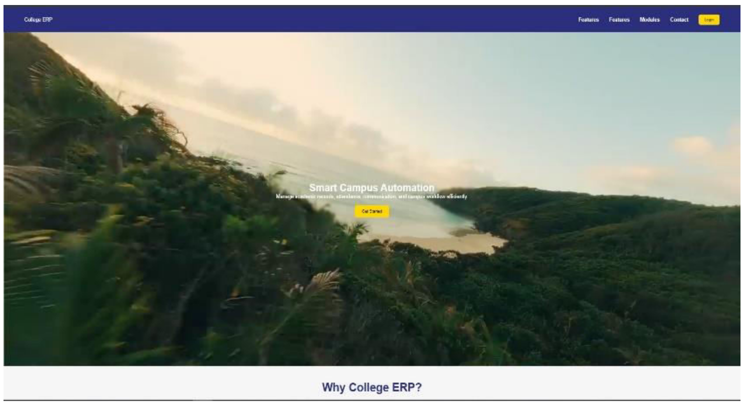
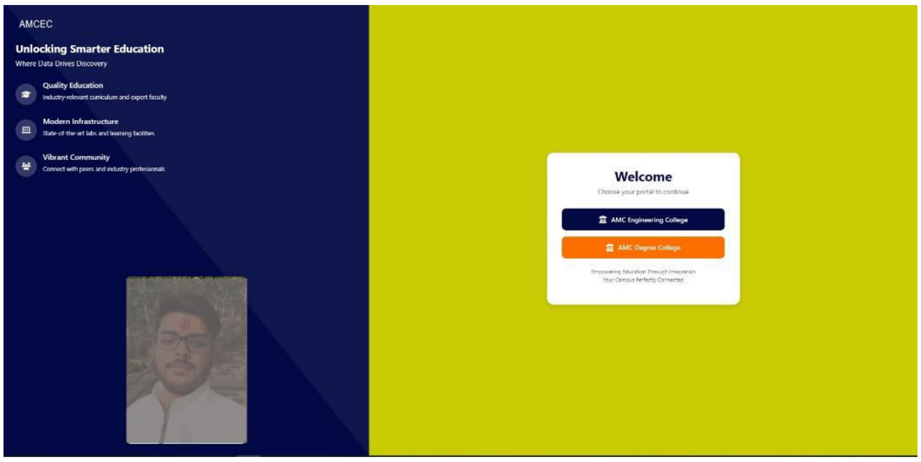


Figure 2: Landing page

2. Multi college Page:
This page is an entry point to various college portals within the same ERP system. It allows central access to several institutions. To the users, it clearly and distinctly presents options regarding AMC Engineering College and AMC Degree College, thus showing the capability of the system even with different academic setups. This design supports integration and smart education by allowing users first to choose their college before proceeding to role-based login, as required for a personalized, organized experience.


AMCEC

Unlocking Smarter Education
Where Data Drives Discovery

Quality Education
Industry-relevant curriculum and expert faculty

Modern Infrastructure
State-of-the-art labs and learning facilities

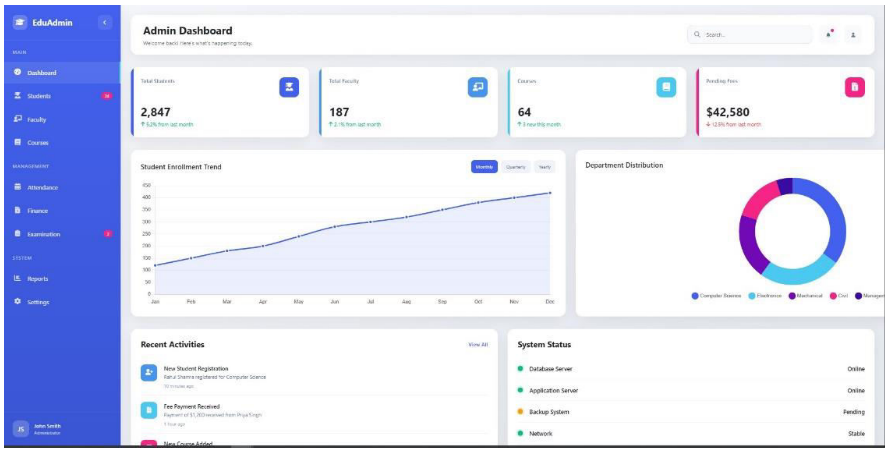
Vibrant Community
Connect with peers and industry professionals

Welcome
Choose your portal to continue

Empowering Education Through Integration
Your Campus Perfectly Connected

Figure 3: Multi college page

3. Admin dashboard Page:
Admin Dashboard provides complete visibility on the institution with key indicators and real-time charts showing important numbers: number of students enrolled, number of teachers, number of courses, and financial data regarding funding/fines. The interactive charts plot enrollment trends and distribution between departments. System status tools monitor how well everything is running. And recent activities and announcements keep Admins informed, making this dashboard the main place for decisions and overall management.



Admin Dashboard
Welcome back! Here's what's happening today.

Total Students 2,847 ↑ 5.2% from last month
Total Faculty 187 ↑ 2.1% from last month
Courses 64 ↑ 3 new this month
Pending Fees $42,580 ↓ 12.5% from last month

Student Enrollment Trend

Department Distribution

Recent Activities
New Student Registration
Rahul Sharma registered for Computer Science
10 minutes ago
Fee Payment Received
Payment of $1,200 received from Priya Singh
1 hour ago
New Course Added

System Status

Database Server	Online
Application Server	Online
Backup System	Pending
Network	Stable

Figure 4: Admin dashboar

4. admin staff dashboard page:
The Admin staff has its own login id where, on behalf of admin where admin always cannot update details, so staff is been given the assess to update all the detail which should be updated in the website for continuous flow of data in the website

5. Student dashboard Page:
The Student Dashboard serves as a personalized learning hub that shows the curriculum for students, attendance records, internal marks, dates of examination, and fee details. It has an attendance progress indicator and feeds for recent activities like submitted assignments and fee payments. The upcoming deadlines for assignments and payments are listed along with quick-action buttons to update profiles, view syllabi, pay fees, and access library resources at one go

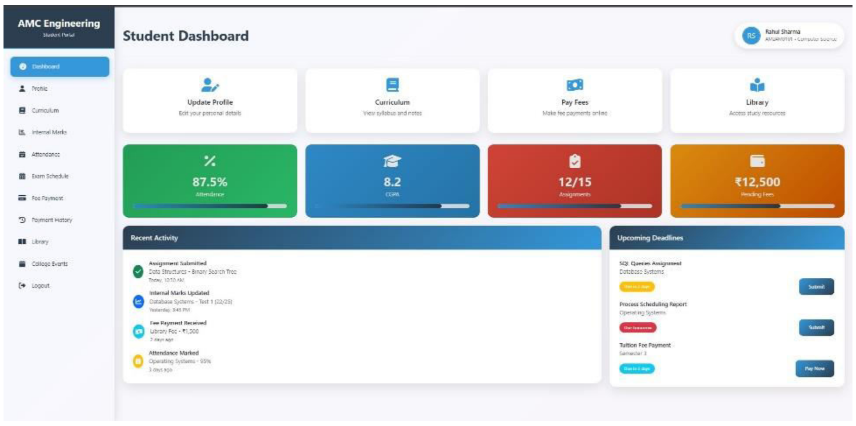


Figure 5: Student Dashboard

6. faculty dashboard Page:
The Faculty Dashboard is a task-friendly screen that helps teachers manage academic work in minimum time. It includes fast-access to mark attendance, entry of internal marks, assignment handling, and updating of student details. Quick charts on attendance trends and activity logs with recently uploaded assignments and updated marks are provided. The forthcoming deadlines thus become easy to check, so that teachers are organized and ahead of schedules.

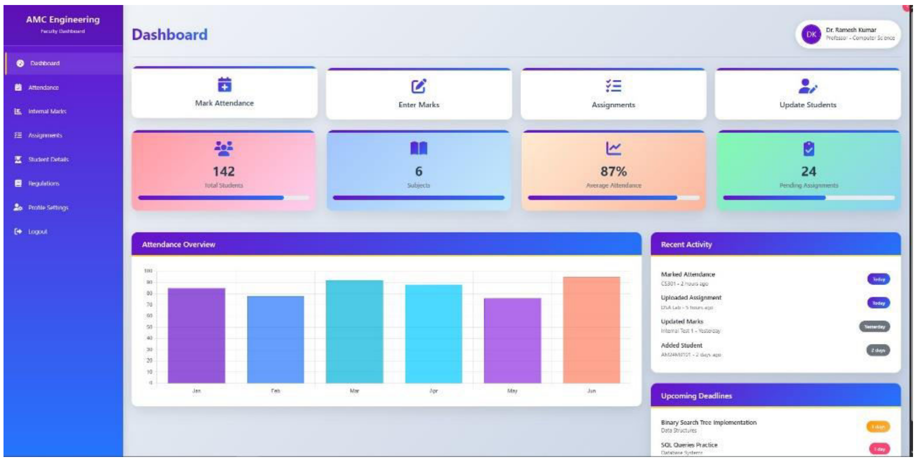


Figure 6: Faculty Dashboard

7. Login Page:
Login Page The login page is designed as a secure, role-based entry point that will route the users to their correct dashboards. It provides four login options: Faculty, Student, Admin, and Staff, with a short description of each and a dedicated login button. The card-style layout is clean, and the footer contains copyright information. This page routes users to the right interface based on their role.

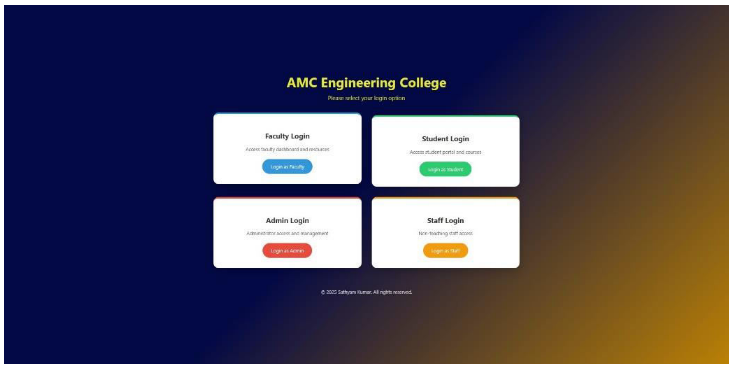

AMC Engineering College
Please select your login option
Faculty Login
Access faculty dashboard and resources
Login as Faculty
Student Login
Access student portal and courses
Login as Student
Admin Login
Administrator access and management
Login as Admin
Staff Login
Non-teaching staff access
Login as Staff
© 2025 Sathyam Kumar. All rights reserved.

Figure 7: Login page

Benefits of This Structure within a College ERP System

1. scalable, role-based setup that clearly outlines permissions and interfaces with students, teachers, staff, and admins.
2. Real-time academic and operational insights, featuring live dashboards that keep users current on attendance, deadlines, fees, and overall school performance.
3. Centralized modular design: data stays consistent across campuses, yet separate updates for something like grading, attendance, or finance modules can be allowed.
4. Secure and simple access control with separate login portals for each user role
5. Integrated campus management views support better-informed decisions with a single view.

VI. FUTURE UPDATES

The future updates will focus on the introduction of AI analytics, a separate mobile application that also supports offline mode, enhanced functions of LMS, blockchain for credential security, the use of IoT in automation for smart campuses, and the use of AI virtual assistants to enhance system intelligence with user interactivity.

VII. CONCLUSION

The result of this project is a modern College ERP System that uses a cloud-native microservices solution and aims to address inefficiencies associated with traditional college administration. By merging admissions, operations, finance, and exams into an integrated platform with a smooth experience, it automates various operations and enhances transparency. Based on Presuppositions of Design Science Research, it develops an efficient system that uses React.js, Node.js, and Docker with scalable and user-friendly designs. The ERP reduces labor and error complexities and enhances transparency and intelligence. It establishes an efficient platform that not only addresses today's administrative and learning processes but will also be equipped with next-generation technologies such as AI and IoT for an efficient learning platform.

REFERENCES

Here are few references for the ERP system which is been refereed while creating the College ERP system

1. Backend development – Guide to MongoDB. Official Database Documentation on MongoDB architecture patterns. https://www.mongodb.com/docs/manual/tutorial/
2. Backend connectivity with Frontend. Official site for scripting and Frontend backend connectivity. https://www.javascripttutorial.net/
3. Research paper references. ERP planning guide and use case implementation. https://www.sciencedirect.com/science/article/pii/S2212017313002120
4. Research paper for Past, present, future of ERP System. Help to make the software to be updated with new features. https://www.tandfonline.com/doi/full/10.1080/13614576.2020.1742770#abstract
5. Research paper references. For Future problems of current ERP systems. https://www.scitepress.org/Papers/2021/104773/104773.pdf