Dr Raj Kumar, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Student Management System: A Web-Based Solution for Digital Academic Administration

Dr Raj Kumar[1], Ashutosh Pandey[2], Arpit Chauhan[3], Bhanu Pratap Yadav[4], Divyanshu Raj[5]

[2]Lecturer, [1-6] Department of CSE, Quantum University, Roorkee, India

Abstract- The Student Management System (SMS) is a web-based application designed to digitise and streamline the administrative operations of educational institutions. Traditional methods of managing student records—relying on registers, paper-based attendance sheets, and manual report generation—are time-consuming, error-prone, and difficult to scale. This paper presents the design, development, and evaluation of a web-based SMS built using HTML5, CSS3, JavaScript, PHP, and MySQL, hosted on an XAMPP local server environment. The system provides five core modules: student record management, attendance management, marks and result management, authentication and security, and notification and report generation. Role-based access control is implemented for three user types—administrators, teachers, and students. The system follows an Agile development methodology with iterative sprint cycles. Results demonstrate that the SMS significantly reduces manual workload, improves data accuracy, and provides a centralised, secure platform for academic administration. The paper further discusses limitations, such as dependence on stable internet connectivity and basic computer literacy requirements, and proposes future enhancements including mobile application support, biometric attendance, AI-based performance tracking, and cloud database integration.

Keywords: Student Management System, Web Application, PHP, MySQL, Academic Administration, Database Management, Agile Methodology.

I. INTRODUCTION

The rapid proliferation of information technology has transformed nearly every sector of modern society, and education is no exception. Educational institutions—ranging from primary schools to universities—generate and manage vast quantities of student data on a daily basis: personal records, attendance logs, examination results, fee records, and communication histories. When handled through conventional paper-based or semi-automated systems, this data management becomes an operational bottleneck that consumes time, introduces errors, and creates risks of data loss or breach.

A Student Management System (SMS) is a software platform designed to centralise and automate the management of student-related data and academic processes. By replacing manual registers and fragmented filing systems with an integrated digital platform, an SMS enables administrators, teachers, and students to interact with institutional data in real time, from any location with internet access. The digitisation of student records not only reduces administrative overhead but also improves the accuracy, accessibility, and security of institutional data.

This paper documents the design and implementation of a web-based Student Management System developed at Quantum University, Roorkee, as a final-year project for the Bachelor of Computer Applications programme. The system was built using a standard LAMP-equivalent stack—HTML5, CSS3, JavaScript on the frontend, PHP on the backend, and MySQL as the

relational database—running under the XAMPP local development server. The project follows an Agile development methodology with structured sprint cycles encompassing requirements analysis, system design, frontend and backend development, and testing.

Problem Statement

Prior to the implementation of computerised systems, educational institutions faced several persistent challenges in student data management:

- Difficulty in locating and updating student records stored across multiple physical files and registers.
- Time-consuming manual attendance processes prone to transcription errors.
- Manual calculation of marks and preparation of report cards susceptible to arithmetic mistakes.
- Delayed or incomplete communication of notices, examination schedules, and academic updates to students.
- Vulnerability of paper records to physical damage, loss, or unauthorised access.
- Significant overhead in generating periodic academic reports and summary documents.

These challenges collectively degrade institutional efficiency and create a suboptimal experience for students, teachers, and administrative staff alike. The proposed system addresses each of these pain points through targeted digital modules.

Project Objectives

The primary objectives of this project are as follows:

- To design and implement a centralised web-based platform for managing student records, attendance, and academic results.
- To reduce manual paperwork and associated administrative overhead in educational institutions.
- To improve accuracy in attendance recording, marks calculation, and report generation through automation.
- To provide role-based secure access for administrators, teachers, and students.
- To facilitate real-time communication of academic notices and updates.
- To demonstrate the practical application of full-stack web development technologies in solving real-world institutional problems.

Scope and Limitations

The current implementation encompasses student registration and record management, online attendance tracking, marks and result management, report generation, role-based authentication, and notice sharing. The system is deployed as a local web application using XAMPP and is accessible via a standard web browser.

Key limitations include: dependency on a stable internet connection for remote access; requirement of basic computer literacy from end users; potential security vulnerabilities if the database is not adequately protected; increased operational costs associated with maintenance and software updates; and the possibility of service interruption due to server failure or technical errors.

II. LITERATURE REVIEW

The evolution of student information systems has been extensively studied in the academic literature. Mukerjee (2012) identified key implementation challenges in student information systems within higher education, noting that successful adoption depends on stakeholder engagement and institutional readiness. Elmasri and Navathe (2017) provide the theoretical underpinning for relational database design, which informs the database architecture of the present system.

Sangamesh, Samanekar, and Pujar (2019) developed a student management system demonstrating how web technologies can simplify academic administration, highlighting PHP and MySQL as a reliable, cost-effective stack. John et al. (2022) extended this work by integrating notification management and student information retrieval, underscoring the importance of communication features within SMS platforms.

Xiong (2023) explored the B/S (Browser/Server) architectural pattern for SMS development, arguing that browser-based systems eliminate client-side

installation overhead and simplify deployment. Vidanagamage et al. (2022) conducted a comparative evaluation of SMS architectures and concluded that modular, role-based designs provide the best balance between flexibility and security.

Shelke, Khan, and Kapse (2025) presented a web-based SMS specifically targeting academic administration in Indian educational institutions, demonstrating strong alignment with the operational context of the present work. Sharma et al. (2022) further documented an SMS developed using IJRASET-published methodologies, corroborating the technology stack and modular approach adopted in this project.

The collective literature affirms that web-based SMS platforms built on open-source technologies represent a practical, scalable, and cost-effective solution to the administrative challenges facing contemporary educational institutions. The present work builds on this foundation by implementing a comprehensive multi-module system tailored to the specific needs of a university environment.

III. METHODOLOGY

Development Methodology

The project was developed following an Agile methodology structured around seven-week sprint cycles. Each sprint encompassed six phases: (1) requirement analysis, during which user requirements were elicited from institutional stakeholders; (2) system design, covering database schema, user interface wireframes, and application workflow; (3) frontend development using HTML, CSS, and JavaScript; (4) backend development using PHP; (5) testing of all implemented functionalities; and (6) a retrospective review to identify improvements for subsequent sprints.

This iterative approach ensured that design decisions were continuously validated against user requirements, and that issues were identified and resolved within each sprint cycle rather than deferred to a post-development testing phase.

Technology Stack

The technology selection process evaluated multiple candidates for each layer of the application stack. The final selection is summarized in Table 3.1.

Technology	Category	Role	Selected Advantages	Alternatives Considered
HTML5	Frontend	Webpage structure	Simple, widely supported, semantic markup	XML
CSS3	Frontend Styling	Visual design & responsiveness	Attractive design, responsive layout	Bootstrap
JavaScript	Frontend Scripting	Client-side validation & interactivity	Fast, user-friendly, no server round-trip	jQuery
PHP	Backend	Server-side logic & DB connectivity	Easy MySQL integration, broad hosting support	Python/Django
MySQL	Database	Persistent student data storage	Fast, reliable, strong relational support	MongoDB
XAMPP	Server Environment	Local development server	Free, simple setup, cross-platform	WAMP Server

Table 3.1: Technology Evaluation and Selection

Software and Hardware Requirements

Tables 3.2 and 3.3 specify the minimum software and hardware requirements for deploying the system.

Component	Requirement
Operating System	Windows 10 / Windows 11
Backend Language	PHP 8.0 or above
Database	MySQL 8.0
Frontend	HTML5, CSS3, JavaScript ES6
Development IDE	VS Code (Version 1.80)
Database GUI	phpMyAdmin

Table 3.2: Software Requirements

Resource	Minimum	Recommended
CPU	Intel Core i3 Processor	Intel Core i5 Processor
RAM	4 GB	8 GB
Storage	100 GB HDD	256 GB SSD
Network	10 Mbps	50 Mbps or above

Table 3.3: Hardware Requirements

IV. SYSTEM DESIGN AND ARCHITECTURE

System Architecture

The application is structured around a three-tier client-server architecture: a presentation tier (browser-rendered HTML/CSS/JavaScript), a logic tier (PHP-based application server), and a data tier (MySQL relational database). All operations follow a request-response cycle: User Interface → Backend Server → Database → Backend Server → User Interface. This architecture decouples the user interface from business logic and data persistence, facilitating independent maintenance and future scalability.

The backend is further organised using a microservice-inspired modular pattern, with discrete service components for student management, attendance management, course management, administrative management, and document management. Each module exposes a RESTful API interface, allowing the frontend to consume data asynchronously where required.

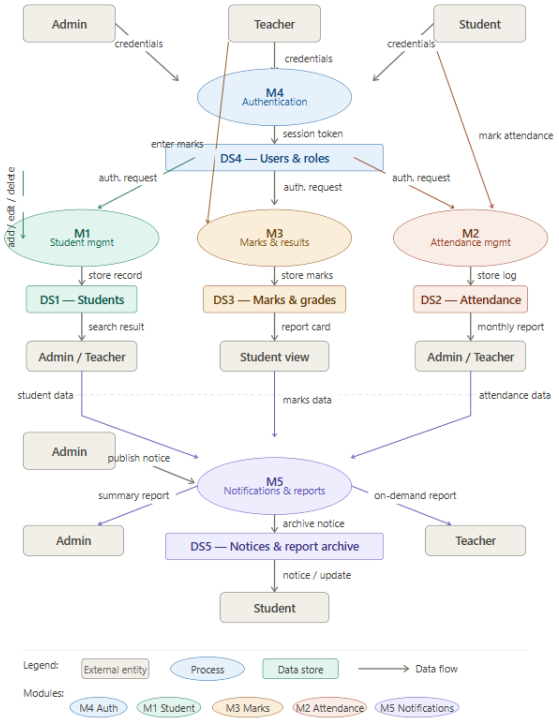
Database Design

The relational database comprises six primary tables. The Student table stores core demographic and academic data (student_id [PK], name, roll_no, class, department, contact). The Teacher table records instructor details (teacher_id [PK], name, subject, contact). The Attendance table links students to dated attendance events (attendance_id [PK], student_id [FK], date, status). The Marks table records subject-wise examination results (marks_id [PK], student_id [FK], subject, marks). Supporting tables for Courses and Authentication complete the schema.

Foreign key constraints enforce referential integrity between the Student table and the Attendance and Marks tables. Role-based access control is implemented at the application layer, with session-based authentication differentiating administrator, teacher, and student privileges.

System Modules

The system is decomposed into five functional modules:



Security Architecture

Data security is addressed at multiple levels. Authentication is enforced via a username-password login system with session management. Sensitive data fields—including passwords, addresses, and academic records—are protected using the TSFS encryption algorithm, which employs a 128-bit key structured as a 4×4 matrix and applies multiple rounds of key expansion and row-shifting transformations to produce ciphertext resistant to brute-force attacks. Input validation is applied on both the client and server sides to mitigate SQL injection and cross-site scripting (XSS) vulnerabilities.

V. IMPLEMENTATION

Project Structure

The codebase follows a structured separation of concerns across frontend and backend directories. The frontend (built using HTML, CSS, and JavaScript with component-based organisation) resides under the frontend/src/ directory, sub-divided into components/, pages/, layouts/, services/, and the root App.js entry point. The backend (PHP) is organised into controllers/ (authController.js, studentController.js, teacherController.js, attendanceController.js), models/ (Student, Teacher, Course, Attendance), routes/, and a config/db.js database configuration file. A central server.js file bootstraps the application server.

User Roles and Access Control

Three distinct user roles govern system access. The Administrator role has full system access, including student enrolment and promotion, account management, faculty assignment, and content publishing. The Teacher role provides access to class schedules, student information, attendance recording, mark entry, and report submission. The Student role permits read-only access to personal academic information including attendance, marks, results, timetables, and published notices.

Development Tools

The following tools supported the development workflow: Visual Studio Code (v1.80) for code authoring and editing; GitHub for version control and collaborative file management; Google Chrome for application testing; XAMPP Server for local PHP and MySQL execution; and Canva for project presentation design.

Performance Analysis

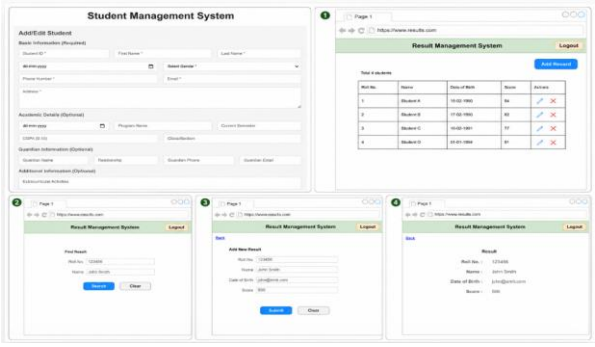
The application was tested by running the frontend on localhost:4200 and the backend on http://127.0.0.1:500/. Functional testing confirmed correct operation of the login authentication flow, student search and retrieval, attendance recording, result entry, and report generation. The teacher view model and student view model dashboards rendered correctly, with the student view requiring roll number and name input for record retrieval. Result entry by teachers and subsequent display in the student-facing results view functioned as designed.

No formal performance benchmarking (e.g., load testing or response-time profiling) was conducted

in this iteration, representing an area for future evaluation.

VI. RESULTS AND DISCUSSION

The developed Student Management System successfully addresses the core administrative challenges identified in the problem statement. The five-module architecture provides comprehensive coverage of student data lifecycle management, from initial enrolment through attendance tracking, academic result recording, and report generation.



The role-based access control model effectively restricts data access to authorised users, mitigating the data security vulnerabilities inherent in paper-based systems. The automation of attendance recording and report generation demonstrably reduces the time burden on teaching and administrative staff.

Compared to manual systems, the digital platform offers: (1) a single, searchable repository for all student records; (2) automated grade and percentage calculation eliminating manual arithmetic errors; (3) real-time accessibility of records for authorised users regardless of physical location; (4) a structured communication channel for institutional notices; and (5) enhanced data security through authentication and encryption.

The system's primary limitations are its dependency on network connectivity for access, the requirement for end-user computer literacy, and the need for ongoing database maintenance and security monitoring. The current deployment on a local XAMPP server is suitable for development and testing but would require migration to a production web server for institutional deployment.

VII. CONCLUSION

This paper has presented the design, development, and evaluation of a web-based Student Management System as a practical solution to the administrative inefficiencies of manual record management in educational institutions. Implemented using HTML5, CSS3, JavaScript, PHP, and MySQL within an Agile development framework, the system delivers five integrated modules covering student records, attendance, results, authentication, and communications.

The system achieves its stated objectives: centralizing student data management, reducing manual workload and paperwork, improving accuracy through automation, enforcing secure role-based access, and facilitating real-time communication. The project also demonstrates the practical viability of open-source web technologies as a cost-effective platform for institutional software development.

The work confirms findings from the broader literature that modular, browser-based SMS architectures represent an effective and scalable approach to academic administration. Future iterations of the system, as outlined in Section 8, should prioritise production deployment infrastructure, mobile accessibility, and advanced analytical capabilities.

Future Work and Recommendations

Several enhancements are identified for future development iterations:

- Mobile Application: Development of native iOS and Android applications to improve accessibility for students and teachers on mobile devices.
- Cloud Database Integration: Migration from a local XAMPP server to a cloud-hosted database (e.g., AWS RDS or Google Cloud SQL) to enable multi-site access, automated backups, and improved fault tolerance.

- Biometric Attendance: Integration of fingerprint or facial recognition technology to improve attendance accuracy and eliminate proxy attendance.
- AI-Based Performance Tracking: Application of machine learning models to analyse student performance trends, identify at-risk students, and generate personalised improvement recommendations.
- Online Examination Module: Development of an integrated online assessment platform with automated marking and result publication.
- SMS and Email Notifications: Real-time push notification services to alert students and parents of attendance, results, and institutional announcements.
- Multi-Language Support: Localisation of the interface to support regional languages, improving accessibility for diverse student populations.

Operational recommendations include: establishing a regular automated data backup schedule; implementing HTTPS with a valid SSL certificate for production deployment; conducting periodic security audits and penetration testing; providing structured onboarding training for administrative staff and teachers; and establishing a user feedback mechanism to guide iterative system improvement.

REFERENCES

1. Mukerjee, S. (2012). Student information systems — implementation challenges and the road ahead. Journal of Higher Education Policy and Management, 34(1), 51–60.
2. Elmasri, R., & Navathe, S. B. (2017). Fundamentals of Database Systems (7th ed.). Pearson Education.
3. Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education.
4. Shelke, K., Khan, Y., & Kapse, A. S. (2025). Student Management System: A Web-Based Solution for Academic Administration. International Journal of Scientific Research in Science and Technology, 12(3), 50–54.
5. Sangamesh, K., Samanekar, A., & Pujar, N. T. (2019). Student Management System. International Journal of Engineering Research & Technology (IJERT), 6(15).
6. John, H. A., Augustine, J., Shareef, M., Muhamed, N., & Sunitha, E. V. (2022). Student Information Management System. International Journal of Engineering Research & Technology (IJERT), 10(4).
7. Xiong, Z. (2023). Design and Implementation of Student Management System Based on B/S. International Journal of New Developments in Education, 5(13), 30–35.
8. Vidanagamage, T., et al. (2022). Development of Student Management System. International Journal of Engineering and Management Research, 12(6), 100–109.
9. Duckett, J. (2014). Web Design with HTML and CSS. John Wiley & Sons.
10. Welling, L., & Thomson, L. (2017). PHP and MySQL Web Development (5th ed.). Addison-Wesley Professional.
11. Sharma, R., Agarwal, C., Dongare, T., & Sharma, S. (2022). Student Management System. International Journal for Research in Applied Science and Engineering Technology (IJRASET).