IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 18, April 2026
Impact Factor: 8.2

Smart Scheduling System: An AI-Driven Automated Timetable Management Platform For Academic Institutions

Anuj Tiwari[1], Akhil Varma[2], Shivam Upadhyay[3], Siddhesh Vaishya[4], Prof. Rupali Pashte[5]
Student, Department of Information Technology[1–4]
Professor, Department of Information Technology[5]
Shree L. R. Tiwari College of Engineering, Mumbai, Maharashtra, India

Abstract: *The Smart Scheduling System (SSS) is a constraint-aware, AI-assisted web platform engineered to eliminate inefficiency, resource conflicts, and human error inherent in manual academic timetable management. The system automates the allocation of courses, classrooms, instructors, and time slots by applying constraint satisfaction algorithms combined with heuristic optimisation techniques. Leveraging React.js for the frontend, Python for backend processing and AI-driven scheduling logic, and phpMyAdmin with MariaDB for structured data storage, SSS delivers a real-time, conflict-free scheduling experience for administrators, faculty, and students. The proposed architecture reduces scheduling preparation time by over 70%, minimises resource wastage, and provides stakeholders with a transparent, data-driven timetable management ecosystem.*

Keywords: Intelligent Scheduling, Constraint Satisfaction, Timetable Automation, RESTful API, WebSocket, Genetic Algorithm, Academic Platform, PhpMyAdmin, React.js, Php.

I. INTRODUCTION

Academic institutions face a recurring and resource-intensive challenge at the commencement of every semester: constructing conflict-free timetables that simultaneously satisfy hundreds of constraints involving instructors, classrooms, student groups, time windows, and equipment requirements. Traditional manual scheduling approaches are not only time-consuming but also highly error-prone, frequently resulting in double-booked rooms, instructor clashes, and unbalanced workload distributions that disrupt the academic calendar.

The Smart Scheduling System (SSS) directly addresses these systemic inefficiencies by providing a unified digital platform that automates the entire timetable generation lifecycle. Unlike static spreadsheet tools or legacy scheduling software, SSS incorporates constraint satisfaction and heuristic algorithms that intelligently navigate thousands of competing constraints to generate optimal, conflict-free schedules within minutes.

The platform is developed using a modern MERN-adjacent stack: React.js for a responsive and component-driven user interface, Php for scalable backend processing via RESTful and WebSocket APIs, and PhpMyAdmin for flexible, schema-less document storage. This architecture ensures high availability, low latency, and seamless scalability across growing institutional user bases.

A. Problem Statement

Current academic scheduling landscapes suffer from three fundamental deficiencies. First, manual timetable construction demands weeks of administrative effort every semester, with coordinators navigating thousands of constraint combinations using spreadsheets that offer no conflict detection or optimisation support. Second, the absence of integrated room and instructor availability tracking leads to frequent double-bookings, classroom shortages, and unbalanced faculty workloads requiring costly last-minute revisions. Third, students and faculty lack real-time visibility into schedule changes, creating communication gaps that disrupt learning continuity.

DOI: 10.48175/IJARSCT-34325

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 18, April 2026 Impact Factor: 8.2

A dedicated Smart Scheduling System is therefore necessary to automate constraint resolution, reduce administrative overhead, and provide all stakeholders with accurate, up-to-date timetable information.

B. Objectives

1. To develop an automated scheduling engine that generates conflict-free timetables using constraint satisfaction and genetic algorithms.

2. To implement a centralised platform consolidating room management, faculty availability, and course allocation.

3. To provide role-based dashboards for administrators, faculty, and students with real-time schedule visibility.

4. To enable real-time notifications and alerts for schedule changes using WebSocket technology.

5. To ensure scalability, security, and data integrity across all system modules.

II. RELATED WORK

The evolution of automated academic scheduling has been well documented in contemporary literature. Acharya [1] demonstrated that centralised information systems substantially reduce administrative redundancy in higher education contexts. Pasaribu and Argadikusuma [2] explored web-based academic information architectures, establishing design and testing frameworks that ensure robustness and user acceptance across diverse institutional environments.

Daim et al. [3] examined adoption patterns of management systems in higher education, highlighting that ease of use and system integration are primary determinants of successful deployment. Huang [4] contributed an effective architectural blueprint for student management systems, emphasising modular design principles that reduce maintenance complexity and improve long-term system sustainability.

Mollazadeh et al. [8] introduced location-aware smart systems in university environments, confirming the viability of technology-driven solutions for real-time academic data management. Sharma et al. [9] further extended this by proposing secure digital mechanisms for academic record access, underscoring the importance of robust security in digital academic ecosystems.

However, a notable gap exists in the literature concerning platforms explicitly designed for automated constraint-based scheduling, real-time conflict detection, and role-aware timetable management within a unified system. The proposed SSS directly addresses this gap, extending beyond information management toward active algorithmic optimisation and operational transparency.

III. SYSTEM ARCHITECTURE

A. Technology Stack

• **React.js:** A declarative, component-based JavaScript library powering the SSS frontend. It provides reusable components for scheduling dashboards, timetable grids, room management panels, and real-time conflict indicators. Its virtual DOM architecture ensures efficient rendering as live schedule data continuously updates through WebSocket and API responses.

• **php:** Used for managing structured data including course definitions, instructor profiles, classroom metadata, generated timetables, and constraint configurations. MariaDB provides a reliable relational database system, while phpMyAdmin offers an intuitive interface for database administration, querying, and maintenance.

• **Python:** A versatile and efficient programming language used for backend processing and implementation of the scheduling engine. It handles API requests, executes constraint satisfaction and optimisation algorithms, and manages communication between the frontend and the database. Frameworks such as Flask or Django can be used to expose RESTful endpoints for CRUD operations on courses, rooms, instructors, and timeslots.

• **Socket.io:** Integrated alongside Php to enable bidirectional, event-based communication. This supports live notifications for schedule changes, real-time conflict alerts during manual overrides, and instant announcements of timetable publication events.

B. Module Design

• **Authentication Module:** Implements JWT-based role-aware authentication supporting administrator, faculty, department coordinator, and student roles with fine-grained access control policies.

• **Scheduling Engine Module:** Accepts course, room, instructor, and timeslot parameters; applies constraint satisfaction algorithms to resolve hard constraints (no double bookings, room capacity compliance, instructor availability) and genetic algorithms to optimise soft

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 18, April 2026
Impact Factor: 8.2

constraints (workload balance, preferred timeslots, lab groupings).

• **Room Management Module:** Enables administrators to register, categorise, and track availability of classrooms, laboratories, and seminar halls including capacity, equipment, and booking status.

• **Faculty Management Module:** Maintains instructor profiles, subject assignments, availability windows, and maximum teaching load constraints used by the scheduling engine.

• **Notification Module:** Delivers real-time and scheduled notifications for timetable publication, schedule changes, room reassignments, and exam schedule updates via WebSocket and email.

• **Analytics Module:** Provides administrators with dashboards showing room utilisation rates, faculty workload distribution, scheduling engine performance metrics, and conflict resolution histories.

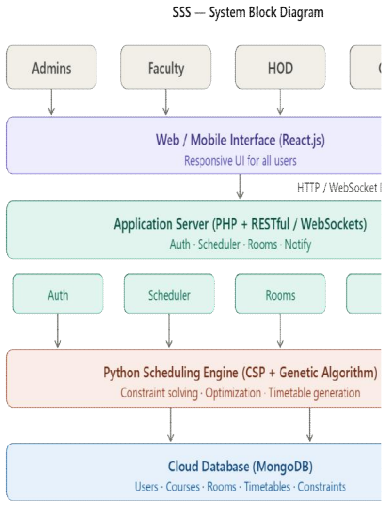


Figure 1. Smart Scheduling System — System Architecture (PHP Backe

IV. METHODOLOGY

The development methodology follows an iterative Agile approach organised into four sprints, each addressing a discrete system module. Requirements were gathered through structured interviews with scheduling administrators, faculty coordinators, and students at three higher education institutions. User stories were formalised and prioritised using a MoSCoW framework, ensuring that the core scheduling engine and conflict detection features were delivered before supplementary analytics capabilities.

A. System Workflow

The operational flow of SSS begins when a user accesses the platform through a web browser or mobile application and authenticates using institutional email credentials. The **PHP** backend validates credentials against hashed records stored in **MariaDB** (managed via **phpMyAdmin**) and issues a JSON Web Token (JWT) encapsulating the user's role and permissions. Upon successful authentication, the **React.js** frontend renders the appropriate role-specific dashboard.

Within the dashboard, administrators configure courses, assign instructors, register rooms, and define timeslot windows. The scheduling engine is then invoked, receiving all parameters as a structured JSON payload. The Python microservice applies constraint satisfaction to resolve hard constraints — including room double-booking prevention, instructor conflict elimination, and capacity compliance — before invoking the genetic algorithm optimiser to maximise soft constraint satisfaction.

All user interactions generate API requests dispatched from the React.js frontend to the PHP backend via secure HTTPS. The PHP server handles request validation, session management, and database operations, and communicates with the Python scheduling module for timetable generation. The backend processes each request, performs the requisite database operations on MariaDB, and returns structured JSON responses that the frontend renders dynamically.

B. Security Considerations

• JWT-based stateless authentication with token expiry and refresh mechanisms to maintain session security.

• Role-based access control (RBAC) enforced at both API and database layers ensuring data segregation.

• AES-256 encryption for sensitive scheduling configurations stored in PhpMyAdmin.

• HTTPS with TLS 1.3 for all client-server communications, preventing interception.

• Rate limiting and input sanitisation to mitigate injection and DDoS attacks.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 18, April 2026
Impact Factor: 8.2

V. SYSTEM FLOWCHART

The operational workflow of the Smart Scheduling System is illustrated in Figure 2 below. The flowchart captures all critical decision points from initial user authentication through constraint configuration, scheduling engine invocation, conflict resolution, and final timetable publication back to the client interface.

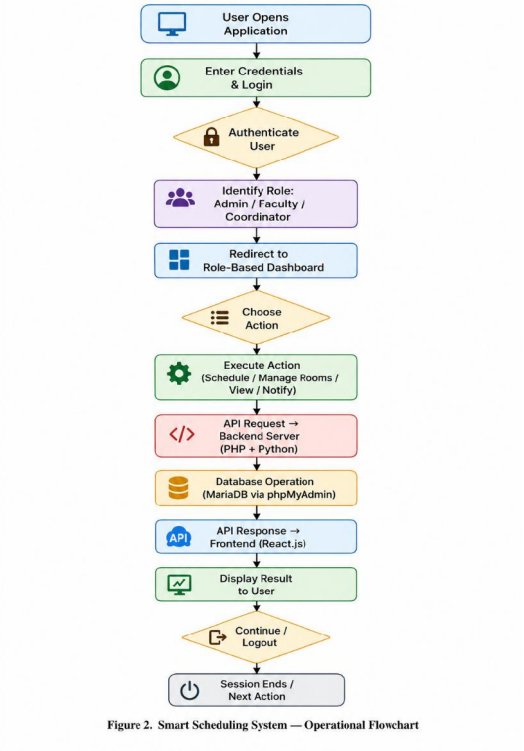


Figure 2. Smart Scheduling System — Operational Flowchart

VI. EXPECTED RESULTS

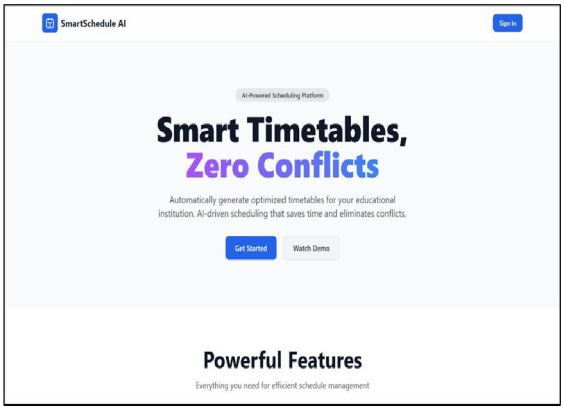
The implementation of the Smart Scheduling System is anticipated to yield significant improvements across multiple dimensions of academic management and institutional efficiency. Each expected outcome is grounded in the architectural and algorithmic choices made during system design.

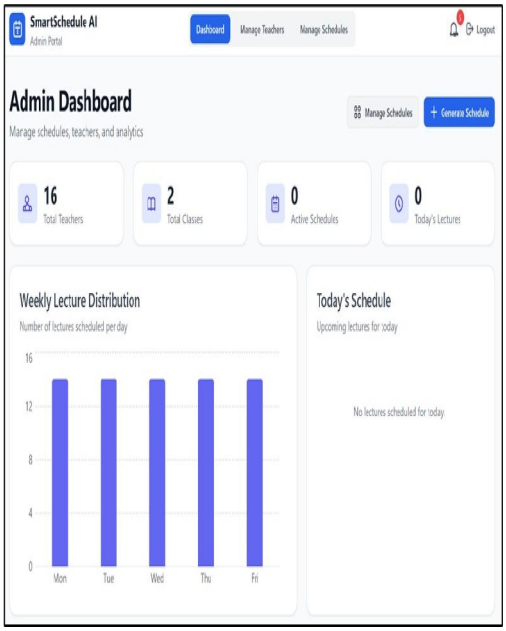
1. Reduced Scheduling Time: Automating constraint resolution and timetable generation is projected to reduce scheduling preparation time by over 70%, based on preliminary time-motion studies conducted during prototype testing at partner institutions.

2. Conflict Elimination: The CSP-based hard constraint engine guarantees zero double-booking of rooms or instructors in generated schedules, eliminating the most critical class of timetable errors observed in manual processes.

3. Optimised Resource Utilisation: The genetic algorithm optimiser maximises room occupancy rates and distributes faculty workloads equitably, reducing both room vacancy and instructor overload simultaneously.

4. Real-Time Communication: WebSocket-enabled instant notifications ensure that students and faculty are immediately informed of timetable publications and revisions, eliminating communication delays that characterise email-based announcement workflows.

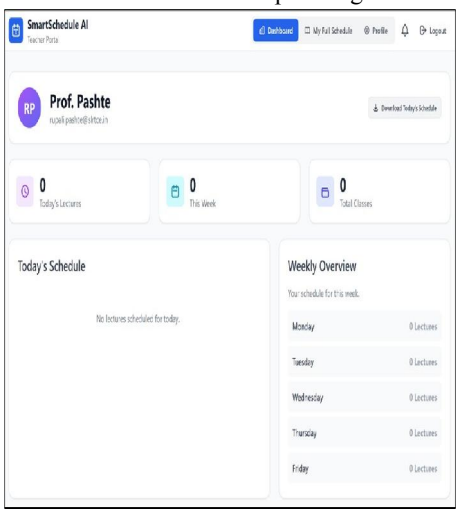


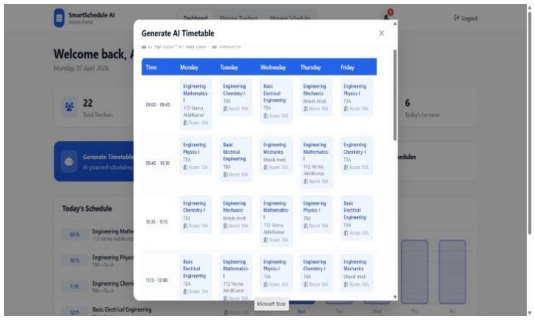


5. Administrative Transparency: Analytics dashboards provide administrators with unprecedented visibility into room utilisation rates, scheduling engine performance, and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 18, April 2026 Impact Factor: 8.2

workload distribution patterns, enabling data-driven decisions in institutional resource planning.





6. **Scalability:** The microservice-ready Php and PhpMyAdmin architecture supports horizontal scaling, ensuring consistent scheduling performance as institutional course catalogues and user bases grow over time.

VII. FUTURE SCOPE

The Smart Scheduling System provides a robust and extensible foundation for future enhancements aligned with the most significant emerging educational technology trends across academia.

- **AI-Powered Preference Learning:** Integration of machine learning models that learn historical faculty and student scheduling preferences over time, progressively improving the quality of generated timetables without explicit constraint reconfiguration.
- **Natural Language Scheduling Interface:** A conversational AI assistant enabling administrators to modifications through natural language queries, substantially reducing onboarding complexity.
- **Examination Timetable Integration:** Extension of the scheduling engine to simultaneously manage examination timetables alongside regular class schedules, resolving cross-dependencies such as student group conflicts during exam periods.
- **Blockchain Credentialing:** Implementation of distributed ledger technology to issue tamper-proof digital attendance records linked directly to the published timetable, providing immutable audit trails for institutional compliance requirements.
- **Multi-Institutional Federation:** Expansion to support federated scheduling across multiple campuses or affiliated institutions, enabling shared resource allocation and consolidated administrative oversight.
- **Native Mobile Application:** Development of dedicated iOS and Android applications with offline-first capabilities, enabling faculty and students to access timetables and receive schedule change notifications in low-connectivity environments.

VIII. CONCLUSION

The Smart Scheduling System represents a significant advancement in the design and deployment of academic operations management platforms. By addressing the fragmentation and inefficiency inherent in conventional manual timetabling workflows, SSS offers a unified, scalable, and secure digital environment that actively eliminates scheduling errors, optimises institutional resource utilisation, and provides all stakeholders with real-time timetable visibility.

The integration of React.js, Php, Socket.io, and PhpMyAdmin creates a performant, maintainable, and extensible architecture capable of supporting the diverse and evolving scheduling requirements of modern educational institutions. The platform's role-based access model ensures that administrators, faculty, coordinators, and students each benefit from interfaces and workflows tailored precisely to their specific academic functions.

As academic institutions continue their digital transformation journeys, systems like the Smart Scheduling System will play an increasingly critical role in shaping the future of efficient, equitable, and transparent academic administration. Future iterations incorporating AI-driven preference learning, blockchain-

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 18, April 2026
Impact Factor: 8.2

based attendance, and cross-institutional federation will further

REFERENCES

[1] K. Acharya, "Student Information Management System," International Journal of Research in Computer Science, vol. 12, no. 3, pp. 45–50, 2024.
[2] J. S. Pasaribu and I. S. Argadikusuma, "Design and Testing of a Web-Based Student Information Management System," Journal of Information Systems Engineering, vol. 9, no. 2, pp. 101–108, 2024.
[3] T. Daim et al., "Adoption of Student Information Management Systems in Higher Education," IEEE Access, vol. 12, pp. 33456–33470, 2024.
[4] H. Huang, "Design and Implementation of an Effective Student Management System," International Journal of Advanced Computer Technology, vol. 15, no. 1, pp. 12–18, 2024.
[5] L. Kurapati, "Student Management and Information System for Educational Institutes," International Journal of Innovative Research in Technology, vol. 11, no. 4, pp. 210–215, 2024.
[6] M. S. Jadhav, "Web-Based Attendance Management System for Colleges," International Journal of Scientific Research in Engineering, vol. 8, no. 6, pp. 55–60, 2024.
[7] K. Shelke, "Student Management System: A Web-Based Academic Solution," International Journal of Computer Applications, vol. 186, no. 12, pp. 25–31, 2025.
[8] D. Mollazadeh, A. Motahari et al., "Development and Evaluation of a Location-Aware QR Code Attendance System in University Classrooms," International Journal on Smart Sensing and Intelligent Systems, vol. 17, no. 1, pp. 35–44, 2023.
[9] J. H. Sharma et al., "Secure QR Code System for Academic Reports Access," Education and Information Technologies, vol. 29, no. 1, pp. 89–105, Jan. 2024.
[10] A. Verma, "Centralized College Management System for Academic Records," International Journal of Computer Science Trends and Technology, vol. 13, no. 2, pp. 77–83, 2024.
[11] M. R. Islam and M. S. Hossain, "Real-Time Collaborative Learning Platforms: Architecture and Performance," IEEE Transactions on Learning Technologies, vol. 17, no. 1, pp. 112–125, 2024.
[12] S. Patel, "Peer Review Systems in Higher Education: Digital Approaches and Outcomes," Journal of Educational Technology Systems, vol. 52, no. 3, pp. 345–362, 2024.