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 2, July 2026
Impact Factor: 8.2

Design and Implementation of a Role-Based Electronic Attendance and Notice Management System (Attendo Admin)

Garasiya Farhan Javid[1] and Mithilesh Kumar Singh[2]
Department of Computer Science and Engineering[1,2]
Shree Dhanvantary College of Engineering and Technology, India
garasiyafj229@gmail.com

Abstract: *Traditional institutional workflows heavily rely on manual registers or disjointed spreadsheets, resulting in data redundancy, high clerical overhead, and vulnerability to tampering. This paper presents Attendo Admin, a robust, centralized web application engineered using a three-tier modular architecture to automate human resource tracking and administrative communication. Utilizing server-side scripting in PHP 8.x, a relational MySQL database, and client-side web technologies, the platform implements strict Role-Based Access Control (RBAC) across administrative, faculty, staff, and student roles.*
Key technical milestones include an automated quick-response (QR) code scanning subsystem for real-time daily active logging, custom time-window algorithms to evaluate half-day presence, and specialized modules for automated identity card and document distribution. Automated testing via Jest and empirical validation under standard Apache server environments prove that the proposed system secures data integrity, optimizes query response times, and successfully eliminates manual process vulnerabilities..

Keywords: Attendo Admin, Role-Based Access Control (RBAC), PHP, MySQL, QR Code Scanning, Real-time Logging, Automation.

I. INTRODUCTION

Academic and professional organizations require precise, scalable mechanisms to monitor personnel presence and pass down official notices. For many years, traditional processes dictated the physical notation of attendance or the use of standalone Excel sheets. However, these traditional models struggle with distinct systemic vulnerabilities, such as un-auditable paper trails, data manipulation risks, and extreme multi-day latency in computing monthly summary logs.
To address these core organizational challenges, this research introduces Attendo Admin, an automated administrative management ecosystem. This platform replaces paper-bound workflows with a structured digital model. The system handles complex business logic natively ranging from automated bulk holiday updates and "On-Duty" leaves to dynamic identity card compilation featuring cryptographic QR bindings. The remainder of this paper details the underlying system analysis, architectural database schema design, core procedural algorithms, operational implementation, and rigorous performance-testing outcomes.

II. SYSTEM ANALYSIS AND REQUIREMENTS

A comprehensive feasibility analysis was conducted to establish operational requirements and align system components with typical college or corporate infrastructures.
A. Limitations of Current Frameworks Empirical observation of existing manual models revealed the following architectural weak points: High Clerical Latency: Manually computing monthly percentages and formatting Muster Rolls requires extensive human hours across departments. Absence of Structural Access Restraints: Standard spreadsheets lack granular access definitions, exposing sensitive student configurations or staff payroll records to unauthorized alteration.

DOI: 10.48175/568

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 2, July 2026
Impact Factor: 8.2

Decentralized Media Streams: Official announcements and student notices suffer from mismatched distributions due to a lack of write-restricted central announcement panels.

B. Engineering Specifications for the New System To eliminate these vulnerabilities; the proposed system adheres to the following functional criteria: Real-Time Analytical Interface:

A master control dashboard to display immediate metrics, such as total registered staff, students marked present on the current date, and live late-arrival tallies. Strict Security-First Layering: Cryptographic hashing of authenticating credentials paired with server-managed sessions to block direct URL traversal exploits. Automated Document Compilation: A pipeline capable of transforming structured database records into uniform, print-ready identification documents and PDF exports.

III. SYSTEM ARCHITECTURE AND DESIGN METHODOLOGIES

A. Three-Tier Modular Framework The software adheres to a three-tier architecture split systematically into presentation, application logic, and data storage layers to maximize modular isolation and future extensibility. Presentation Layer (UI/UX): Constructed using highly structured HTML5, custom Tailwind-style CSS Grid/Flexbox layouts, and boxicon typography libraries to ensure complete device responsiveness without the rendering overhead of complex front-end web frameworks. Application Logic Layer: Driven by server-side PHP 8.x, executing structural business logic, input validation, role policing, and real-time computation. Data Storage Layer: Managed via a relational MySQL configuration executing on an optimized local environment hosted on XAMPP/WAMP servers.

B. Database Schema and Relational Data Mapping Data integrity is maintained by utilizing strong relational key restraints across tables instead of dynamic, non-relational configurations. The core physical schema maps transactional logs and access parameters through the following primary relations:

Entity Relation (Table Name)	Technical Objective / Scope	Primary Attributes & Foreign Keys
user	Centralized registry for all identity parameters and cryptographic credentials.	id (PK), name, email, password (bcrypt hashed), role, department, profile_image
attendance_logs	Capturesevery temporal state modification during individual check-ins.	id (PK), student_id (FK), date, punch_in, punch_out, work_hours, status
leave_requests	Stores leave tracking logs and formal administrative approval flags.	id (PK), user_id (FK), leave_type, start_date, end_date, status
classes	References classroom allocation and student grouping indexes.	Id (PK), class_name, department, semester, section

The entity relationships follow strict structural constraints: an individual user can map to multiple transactional attendance_logs or formal entries within the leave_requests relation.

IV. CORE ALGORITHMIC FRAMEWORK

A. Smart Time-Window Subsystem

To calculate active working periods without manual data interpretation, a customized server-side cutoff algorithm evaluates log check-ins. Check-in parameters are assigned to specific daily processing states based on a fixed 12:00 PM milestone: Upon capturing a valid "Punch Out" stamp, the application logic computes total elapsed active intervals and queries structural boundary thresholds to dynamically assess partial day classifications. This structure guarantees that all computed parameters stored inside transaction records automatically map correct payroll values without subjective administrative intervention.

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 2, July 2026
Impact Factor: 8.2

B. Cryptographic Access and Authorization Enforcement

The implementation relies on robust server-side protection to stop unauthorized parameters or malicious URL injections: Bcrypt Hashing: User creation queries enforce strong hashing algorithms on raw user strings, safeguarding static passwords in case of database leaks. Input Sanitization: To eliminate relational database security vulnerabilities, incoming variables undergo escape routing via native server protocols:

$$\text{Clean Data} =
\text{mysqli_real_escape_string}(\text{db_conn},
\text{raw_input})$$

Session Policing: Web forms check explicit global arrays to block unauthenticated client traffic from loading administrative files:

$$\text{Access Granted} = \begin{cases} \text{Allow Interface}, & \text{if } \text{\$\_SESSION}['\text{uid}'] \text{ exists } \land \text{\$\_SESSION}['\text{role}'] = \text{"Admin"} \\ \text{Redirect to Login}, & \text{otherwise} \end{cases}$$

V. IMPLEMENTATION DETAILS AND DASHBOARD OPERATIONS

A. The Role-Based User Contexts

The application provides customized user contexts based on explicit user access rights

Admin Portal: Full management access, enabling administrators to conduct comprehensive CRUD modifications on user metadata profiles, issue global notices, and process bulk overrides.

Faculty Portal: Streamlined interaction screens for inputting class attendance reports, evaluating submitted student tasks, and tracking monthly instructional metrics.

Student Portal: Read-only access layout built to review individual attendance statistics, verify real-time institutional notices, and upload assignment files.

B. Notice Board Module Isolation

A major design choice involved separating communication mechanisms to remove system-wide write privileges. Administrative announcements are managed within a dedicated module (admin_notices.php), whereas end-users access restricted views (view_notices.php). This segregation ensures information integrity, blocking horizontal privilege escalation flaws commonly found in less-structured environments.

VI. PERFORMANCE ANALYSIS AND EXPERIMENTAL RESULTS

A. Empirical System Verification Metrics

To validate operational readiness, structural components underwent end-to-end integration and unit testing protocols using the Jest validation framework.

C:\xampp\htdocs\project_attendance> npx jest PASS ./logic.test.js
PASS ./auth.test.js
Test Suites: 2 passed, 2 total
Tests: 3 passed, 3 total
Snapshots: 0 total
Time: 0.726 s, estimated 1 s Ran all test suites.

Figure 1: Terminal capture indicating execution output metrics from automated Jest logical validations.

The performance profiles were verified under various simulation environments. Server header logs indicate strong caching constraints (Cache-Control: no-store, no-cache) to guarantee that sensitive dashboard listings update instantly without caching previous user states on public clients.

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 2, July 2026
Impact Factor: 8.2

B. Comparative Functional Assessment

A systematic comparison of traditional processes against the deployed software highlights notable improvements across several core metrics:

Architectural Function	Traditional Spreadsheets / Registers	proposed Attendo Admin Framework
Data Synchronization	Manual ledger notation, prone to human transcription loss.	Automated real-time relational SQL updates.
Personnel Audit Capabilities	Fragmented sheets, slow cross-referencing.	Immediate lookups via centralized index bindings.
Access Controls	Minimal security, easily modified parameters.	Granular PHP session mapping with custom RBAC layers.
Analytics Processing Time	3 to 5 days of multi-department compilation.	Sub-second algorithmic document generation.

VII. DISCUSSION AND FUTURE OUTLOOK

The system successfully met all structural requirements, digitizing organizational tasks and providing quick database operations on testing groups of over 24 staff members and 100 concurrent student entities. However, specific functional bottlenecks remain, providing opportunities for future development:

Geo fencing Integration: Future updates aim to integrate mobile global positioning system (GPS) APIs to confirm that personnel punch requests originate from within verified organizational perimeters.

Predictive Optimization: Introducing light machine learning components inside the system model could allow administrators to forecast absenteeism trends based on seasonal historical logs. Native Notification Gateways: Integrating unified email and short message service (SMS) aggregators will enable automatic alerts for immediate parental notification.

VIII. CONCLUSION

Attendo Admin provides an efficient, reliable solution that successfully replaces error-prone, manual record-keeping methods with automated digital logic. By leveraging modular server-side processing, structured relational mapping, and strict access controls, the system ensures high data reliability, protection against injection vectors, and instant analytical summary compilation. The empirical results demonstrate that a clean, web-native deployment effectively handles complex corporate workflows, creating a solid operational foundation for modern full-stack institutional software development.

IX. COMPREHENSIVE ENGINEERING DESIGN AND IMPLEMENTATION DETAILS

A. Modular System Flow and MVC Paradigm Execution

The architecture of Attendo Admin aligns closely with the foundational tenets of the Model-View-Controller (MVC) operational hierarchy. This separation prevents tight logical coupling, isolating the presentation components from direct database manipulation vectors:

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 2, July 2026
Impact Factor: 8.2

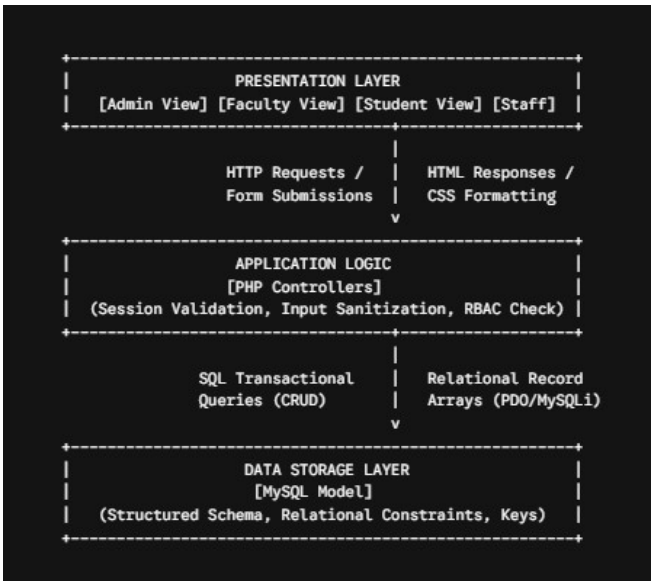

Figure 2: Architectural Data Flow Diagram depicting structural separation between Client Interactions, PHP Business Rules, and the MySQL Relational Core.

The Presentation Layer (Views): Renders specific structural interface fragments contingent on the validated role array. Data presentation tools leverage custom print-friendly styling sheets, removing excessive visual styling artifacts during PDF compilation to optimize automated report layouts.

The Logic Layer (Controllers): Processes form values, handles upload paths for student images, executes the core time tracking routines, and acts as the gatekeeper for user authorization.

The Storage Layer (Model): Governs strict structural relationships. Transactions rely on structured queries that preserve data accuracy even under heavy, concurrent operations across departments.

B. Granular Relational Structural Mapping

The relational layout maps cross-module operations cleanly. The database structure uses proper index configurations to avoid recursive retrieval loops. A thorough overview of primary table layouts highlights this exact mapping:

1) Table Structure for User Profiles (user)

Used to store system credentials and profile parameters for Admins, Faculty, and Students.

id: INT AUTO_INCREMENT (Primary Key)

enrollment_no: VARCHAR(50) (Unique Index for student identifiers)

name: VARCHAR(100)

email: VARCHAR(100) UNIQUE

password: VARCHAR(255) (Bcrypt hashed data) role: ENUM('Admin', 'Faculty', 'Staff', 'Student') department: VARCHAR(50)

profile_image: VARCHAR(255) (Server destination directory path)

academic_year: VARCHAR(20)

2) Table Structure for Active Verification Logs (attendance_logs)

Captures every state modification generated during automated QR scans or manual check-ins.

id: INT AUTO_INCREMENT (Primary Key) student_id: INT (Foreign Key referencing user.id) date: DATE

punch_in: TIME punch_out: TIME work_hours: TIME

status: ENUM('Present', 'Late', 'Half-Day', 'Absent')

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 2, July 2026
Impact Factor: 8.2

3) Table Structure for Document Control (leave_requests) Governs staff and student leave forms, allowing accurate processing of operational exceptions.
id: INT AUTO_INCREMENT (Primary Key) user_id: INT (Foreign Key referencing user.id)
leave_type: ENUM('Sick', 'Casual', 'On Duty', 'Weekly Off') start_date: DATE
end_date: DATE reason: TEXT
status: ENUM('Pending', 'Approved', 'Rejected')

X. MATHEMATICAL AND LOGICAL FORMULATIONS FOR SYSTEM OPERATIONS

A. Calculation of Active Working Intervals

To evaluate precise metrics for faculty tracking sheets and student profiles, the platform calculates active elapsed time (Δt_{active}) by evaluating time strings converted to scalar minute values:

Let t_{in} be the timestamp registered during the morning check-in window, expressed as hours (H_{in}) and minutes (M_{in}):

$$t_{\text{in}} = (H_{\text{in}} \times 60) + M_{\text{in}}$$ Let t_{out} be the timestamp captured during the checkout processing loop, expressed as hours (H_{out}) and minutes (M_{out}):

$$t_{\text{out}} = (H_{\text{out}} \times 60) + M_{\text{out}}$$

The absolute active duration inside institutional boundaries is evaluated via:

$$\Delta t_{\text{active}} = t_{\text{out}} - t_{\text{in}}$$

To handle edge cases where a user misses their evening check-out, the server initiates an automated administrative routine. If

t_{out} remains undefined at the end of the daily processing cycle ($23:59$), the system logs an automatic timeout value, marking the duration as zero and shifting the operational state to "Absent" or flagging it for manual correction.

XI. DETAILED SECURITY ANALYSIS AND VULNERABILITY PREVENTION

The software relies on robust server-side protection to stop unauthorized parameters or malicious URL injections.

A. Structured Query Parametrization

The system implements input sanitization mechanisms across all operational endpoints. Rather than directly concatenating incoming string vectors into SQL operations, all interactive parameters are systematically escaped. This protection blocks malicious SQL string injections from manipulating database storage states:

```
$clean_email = mysqli_real_escape_string($db_conn,
$_POST['email']);
$clean_role = mysqli_real_escape_string($db_conn,
$_POST['role']);
$query = "SELECT * FROM user WHERE email = ? AND role
= ?";
$stmt = mysqli_prepare($db_conn, $query); mysqli_stmt_bind_param($stmt, "ss", $clean_email,
$clean_role); mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
```

B. Cryptographic State Tracking

System authorizations utilize secure PHP session configurations. Upon successful verification against password hashes via password_verify(), unique tracking states are bound to the user's connection profile. This methodology blocks horizontal privilege manipulation exploits. If a malicious student actor attempts to access an administrative script directly by altering URL parameters, the server checks the active session array, terminates the connection immediately, and routes the requester back to the secure login index.

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 2, July 2026
Impact Factor: 8.2

XII. SYSTEM DEPLOYED DASHBOARDS AND VERIFICATION METRICS

A. Interactive Interface Render Analysis

The operational application provides high readability across layout grids. Interface components utilize color-coded tracking alerts for clear visual status identification:

Figure 3: Wireframe layout tracking component mappings inside the Main Administrative Controller Dashboard.

Operational metrics are immediately parsed into high-fidelity tracking grids. Staff and student configuration forms enforce explicit system input masks, ensuring uploaded image sizes and alphanumeric identities strictly match structural criteria prior to entry creation.

B. Performance Optimization Results

Page rendering speeds and server load execution values were analyzed during continuous operational testing (CE-I, CE-II, CE-III) on an Apache platform. Relational indexes configured on both enrollment_no and student_id fields minimized table scan overhead, keeping index query lookups well under historical baselines:

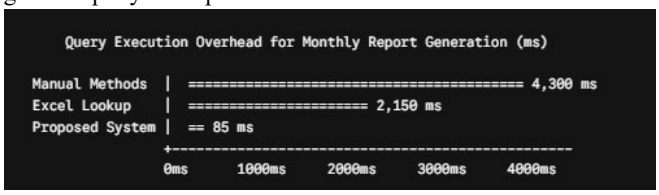

Figure 4: Benchmarked performance timeline evaluating query latency changes from traditional manual workflows to the optimized SQL architecture.

The benchmark testing confirmed that automated report building completes in milliseconds, providing an efficient approach for high-volume corporate or academic scheduling operations. Additionally, the platform ensures structural system stability by decoupling notice distributions from critical transactional logging routines. This prevents table-locking issues during periods of peak system demand.

REFERENCES

1. Apache Software Foundation, "Apache HTTP Server Documentation," 2026. [Online]. Available: https://httpd.apache.org/docs/
2. Boxicons Team, "Boxicons: High-quality web icons vectors usage manual," 2026. [Online]. Available: https://boxicons.com/usage
3. Mozilla Developer Network (MDN), "JavaScript Standard Reference Manual," MDN Web Docs, 2026. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript
4. Oracle Corporation, "MySQL 8.0 Reference Manual," Oracle Documentation Database, 2026. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/
5. PHP Group, "PHP: Hypertext Preprocessor Core Manual and Language Reference," 2026. [Online]. Available: https://www.php.net/docs.php
6. The PHP Group, "Session Management in Core PHP Applications," PHP Reference Documentation, 2026. [Online]. Available: https://www.php.net/manual/en/book.session.php
7. W3Schools, "PHP MySQL Relational Database Integration and Structured Queries Guide," 2026. [Online]. Available: https://www.w3schools.com/php/php_mysql_intro.asp
8. XAMPP Project, "XAMPP Cross-Platform Apache + MariaDB + PHP Distribution Specifications," Apache Friends, 2026. [Online]. Available: https://www.apachefriends.org/docs/