WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

LEAVE NOTIFICATION SYSTEM

Ms. Kaviya Bharathi K[1] , Ms. Nivetha J[2], Ms. Pooja C[3], Ms. Vijayalakshmi S AP/IT[4]

Assistant Professor, Department of Information Technology,
Sri Shakthi Institute of Engineering and Technology, Coimbatore-641062, Tamil Nadu, India.
Email ID: thisisnivethaj@gmail.com

ABSTRACT :

Leave management in colleges is often handled through paper forms, verbal permission, and informal messages to parents. This makes it hard for class advisors to keep a reliable record of who is on leave, whether the leave was approved, and whether the parents were informed. A particular difficulty arises in emergency situations, where a student may obtain permission from the class advisor and leave the campus without logging in to any portal, and the advisor may forget to enter the details afterwards. This paper presents the Leave Notification System, a web-based application that manages normal leave applications, emergency leave reports, student login activity, mentor monitoring, and parent notifications through a single interface. The system combines role-based student and mentor dashboards, leave approval with remarks, an emergency permission workflow with pending-verification tasks and advisor reminders, login activity tracking, a combined monitoring table, and SMS or email alerts to registered parents with logged outcomes. The application is built with HTML, CSS, and Vanilla JavaScript on the frontend, Node.js and Express.js on the backend, and a MySQL database. The system does not claim to detect a physical departure that nobody has reported. It records emergencies that are reported by an advisor or by an authorized exit source, marks incomplete records for verification, and avoids concluding that a student has left campus only because no login was recorded. Functional and integration testing can be used to evaluate the individual modules and their interaction, with further evaluation required on real college data.

Keywords: Leave Management, Emergency Leave, Parent Notification, Login Monitoring, Verification Workflow, Audit Trail, Node.js, MySQL.

1. INTRODUCTION

Accurate leave records are an important administrative requirement in colleges. Class advisors must know which students are on approved leave, parents expect to be informed when their ward is away from campus, and the institution needs a record that can be reviewed later. In many colleges these tasks are still handled through paper letters, phone calls, and messaging applications, which are slow, hard to search, and easy to forget.

Web applications can reduce this effort by letting students submit leave requests online, letting mentors approve or reject them with remarks, and sending notifications to parents automatically. Student login activity can also give mentors a simple view of who has used the portal on a given day. However, an application can only act on information that has been entered into it, and real campus situations do not always follow the normal procedure.

1.1 Existing challenges in leave management

Existing approaches include paper leave forms, phone calls to parents, group messages, and basic attendance registers. Although these methods work for routine cases, they have several weaknesses. Records are scattered across several places, approvals are not always traceable, and parent communication is not logged. Mentors also find it difficult to see in one place the normal leaves, the emergency leaves, and the notifications that failed.

The most difficult case is emergency leave. A student may face a family emergency, obtain permission from the class advisor, and leave without logging in to the portal. If the advisor does not record the permission afterwards, the system has no information about the event. Software cannot detect an event that no person or system has reported. A realistic solution must therefore focus on making reporting easy, reminding staff about incomplete records, and clearly stating what the system can and cannot know.

1.2 Role of verification workflows and login monitoring

A verification workflow treats every emergency report as a record with a status rather than as a final fact. When an advisor grants emergency permission, or an authorized person reports an exit, the system creates a pending record with the original report time. The advisor then completes the details, and the record moves to a verified state. If the record remains incomplete, the system creates a reminder and, where configured, escalates the task to an administrator.

Login monitoring complements this workflow by recording successful student logins with a timestamp. The dashboard can then show how many unique

students logged in today and which students have no login recorded. A missing login is only an absence of data. It is not evidence that the student is absent or outside the campus, and the system labels such students as "No Login Recorded" without drawing further conclusions.

1.3 Problem statement and objectives

Many existing approaches handle leave requests, emergency permissions, login tracking, and parent notification as separate activities. This makes it difficult for mentors to obtain a reliable and complete view of student leave status. There is therefore a need for an integrated application that handles normal and emergency leave, tracks login activity, and informs parents, while being honest about the limits of what can be known automatically.

The main objective of this work is to propose a web-based Leave Notification System that combines leave management, emergency leave reporting, login monitoring, and parent notification in one platform. The specific objectives are:

i. To provide secure, role-based login for students, mentors or class advisors, and administrators.

ii. To support normal leave applications with approval, rejection, remarks, and history.

iii. To provide an emergency leave reporting workflow in which an advisor records permission or an authorized staff member reports an exit.

iv. To create pending verification tasks, reminders, and escalation for incomplete emergency records.

v. To record student login activity and show unique-student statistics from real database data.

vi. To notify registered parents by SMS or email and to keep a log of every attempt and its outcome.

vii. To maintain an audit trail of important actions and to protect records through authentication and authorization.

viii. To evaluate the application through functional, integration, and usability testing.

2. MATERIALS AND METHODS

2.1 Development approach

The project follows a structured software development approach consisting of problem identification, requirement analysis, system design, implementation, integration, testing, and evaluation. During problem identification, the main difficulties of manual leave handling and of unreported emergency leave are considered. Requirement analysis identifies the major functions required by the application, including authentication, leave application, approval, emergency reporting, login tracking, notification, and audit logging.

System design defines the relationship between the user interface, application logic, database, and notification service. The implementation stage develops the required modules using web technologies and backend services, and integration connects the modules so that they exchange data correctly. Testing is then used to identify errors and determine areas requiring improvement.

2.2 System architecture

The proposed system is organized into the user interface layer, application logic layer, authentication layer, notification layer, and data storage layer, as shown in Fig. 1. The user interface layer provides the student and mentor dashboards through which users log in, apply for leave, review requests, and monitor records. The application logic layer processes requests and applies business rules such as validation, duplicate prevention, and status changes. The authentication layer verifies the user's identity and role, while the notification layer sends SMS or email alerts to registered parents, or mock alerts in demo mode, and records the outcome.

The data storage layer maintains students, staff, leave records, emergency reports, login events, verification tasks, reminders, notification logs, and audit entries. The modules are kept logically separate so that, for example, the notification provider or an optional gate-register integration can be changed without redesigning the complete application.

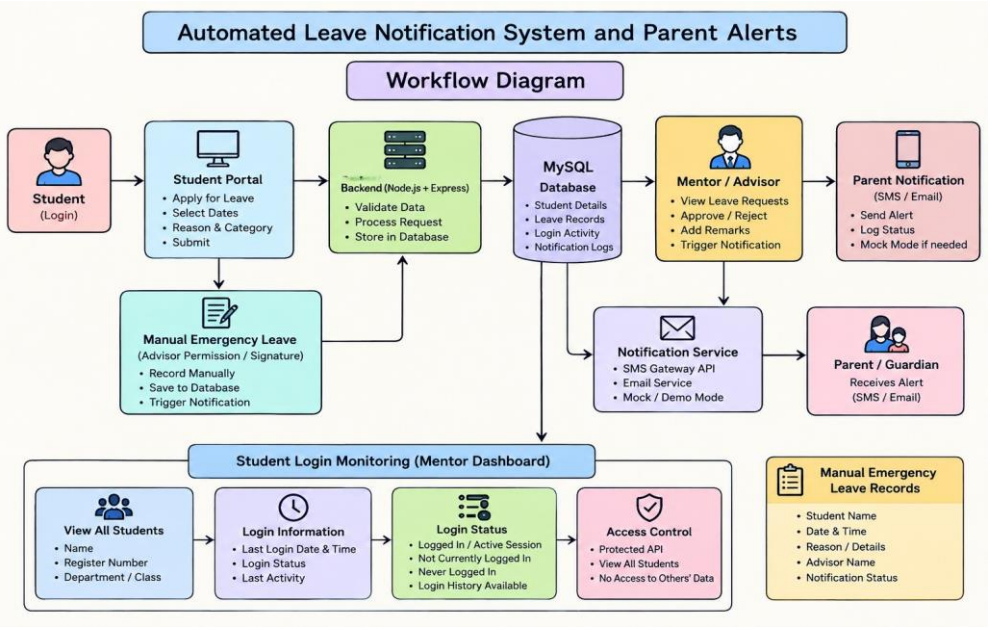


Fig. 1. Workflow and system architecture of the Leave Notification System.

2.3 Technology stack and development environment

The frontend uses HTML, CSS, and Vanilla JavaScript for the student and mentor dashboards. The backend uses Node.js and Express.js to expose authenticated routes. MySQL is used as the relational database, with parameterized queries and transactions for data integrity. Database connection settings, the server port, and notification credentials are read from environment variables, and secrets are never exposed to the browser.

Layer	Technology	Purpose
Frontend	HTML, CSS, Vanilla JavaScript	Student and mentor dashboards
Backend	Node.js, Express.js	Routes, authentication, business rules
Database	MySQL	Persistent records, audit history
Notifications	SMS/email service with MOCK mode	Parent alerts and delivery log

Table 1. Technology stack of the proposed system.

2.4 Data collection and processing

The system processes information required for its functions, such as student and staff details, leave applications, emergency reports, login events, parent contact details, and notification outcomes. Personal information is collected only when necessary. Passwords are never stored in plain text, and parent contact details are retrieved on the server only when a notification is required. All inputs are validated on the server, and the identity of the staff member who records or verifies an emergency entry is taken from the authenticated session and not from a value submitted by the browser.

2.5 Database design

The database design reuses the existing tables for students, staff, and leave applications and adds only the tables needed for the new functions. These include login activity, emergency permission reports, exit or attendance verification records, pending verification tasks, reminder history, and audit history. Foreign keys and indexes are used where appropriate, and schema changes are applied as safe migrations that do not drop existing tables or data. Every emergency record keeps two timestamps: the time the emergency or permission occurred and the time the information was entered into the system. When a record is entered late, both values are preserved so that delayed reporting is visible.

2.6 Emergency leave reporting and verification

The central design question is how the system can know about an emergency leave when the student did not log in and the advisor forgot to record it. The honest answer is that it cannot, unless someone or some system reports it. The proposed workflow therefore provides two reporting routes.

Route A: Advisor records the permission. The advisor creates an emergency permission record with the student register number, reason, permission time, expected departure time, permission method, and the advisor identity taken from the session.

Route B: Independent authorized exit report. If the college has a gate register, security desk, or attendance system, authorized staff can report that a student has left. The source, verification status, and timestamp are recorded. This route is optional and no gate system is assumed to exist.

When an emergency is reported but the leave details are incomplete, the system creates a pending verification task, shows a prominent alert on the mentor dashboard, and displays how long the task has remained pending. A configurable reminder is generated for the responsible advisor without sending repeated duplicates, and overdue tasks can be escalated to an administrator if the college enables an escalation policy. The resolution and the responsible staff member are recorded in the audit trail. The system distinguishes clearly between the situations listed in Table 2.

Situation	How the system treats it
Student did not log in	Shown as "No Login Recorded"; no conclusion drawn
Normal leave submitted	Handled through the approval workflow
Emergency permission reported	Pending record created and verified by advisor
Independent exit event recorded	Stored with source; flagged if leave entry is missing
Emergency record incomplete	Marked "Incomplete Entry"; reminder generated
Physical departure not verified	Marked "Pending Verification"
Departure never reported	Cannot be detected automatically

Table 2. Distinction between login status, reported events, and verification status.

2.7 Login activity monitoring and parent notification

Successful student logins are recorded with the student identifier, timestamp, and result. The dashboard counts unique students rather than repeated logins, uses the college time zone for "today", and shows students with no successful login as "No Login Recorded". These students are not labelled absent or outside the campus without independent evidence.

When a normal leave or a verified emergency leave requires parent notification, the server retrieves the registered parent contact and calls the configured SMS or email service. Each attempt and its outcome are logged, failed notifications can be retried, and duplicate messages are prevented. A message is described as delivered only when the provider confirms delivery, and in MOCK mode it is labelled as a mock notification. For an unverified emergency, an authorized staff member may send a clearly labelled preliminary alert instead of presenting the leave as fully verified.

2.8 Mentor dashboard, security, and testing

The mentor dashboard shows summary cards for registered students, logged in today, students with no login recorded, online applications, pending requests, approved and rejected requests, manual emergency leaves, and parent alerts sent or failed. A unified monitoring table lists the student, register number, event type, date and time, source, approval status, verification status, recorded-by staff member, parent notification status, and remarks, with filters for record type, status, date range, and register number. All figures are derived from database records, and empty states are shown when no data exists.

Students can view only their own records, advisors can access records within their scope, and routes are protected by authentication and role-based authorization. Important actions are written to an audit trail. Testing includes unit checks of individual functions, integration testing between the interface, backend, and database, and system testing of the complete workflow. Error conditions such as invalid credentials, missing fields, duplicate records, unauthorized requests, and failed notifications are also examined. Any quantitative claim about performance is reported only when supported by actual measurements.

3. RESULTS AND DISCUSSION

3.1 Proposed system functionality

The proposed application brings several leave management functions into a centralized interface. These include role-based login, normal leave application and approval, emergency leave reporting with verification tasks, student login tracking, parent notification with a delivery log, and a combined monitoring dashboard. The integration of these functions aims to reduce the effort required to move between paper records, messaging applications, and separate registers.

Fig. 2 shows the Faculty Advisor Portal of the implemented prototype. The top bar shows the notification mode and database status, the header card shows the advisor details and the monitoring cohort, and the summary cards show the number of registered students, students who logged in today, online applications, pending, approved and rejected requests, manual emergency leaves, and parent alerts sent or failed. The values in this screenshot come from a demonstration cohort of 50 students captured in simulated (MOCK) notification mode, and they illustrate the interface rather than measured performance in a real college.

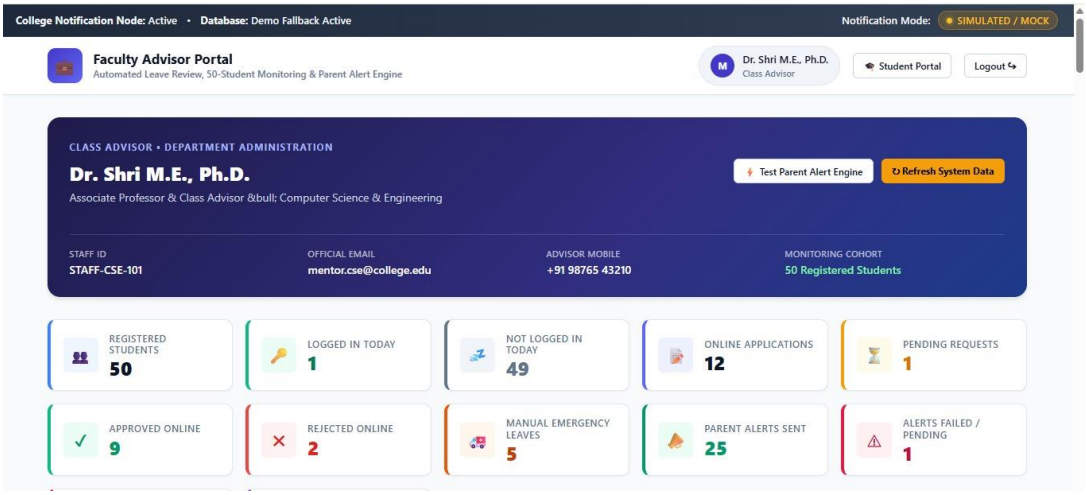


Fig. 2. Faculty Advisor Portal showing the monitoring summary cards (demonstration data, MOCK mode).

3.2 Functional evaluation

Functional evaluation examines whether the implemented modules behave according to their requirements. Representative test scenarios are summarised in Table 3. Results should be documented from the observed behaviour of the application, and untested features should be identified as pending evaluation.

ID	Module	Test scenario	Expected behaviour
T1	Authentication	Login as student and as mentor with valid and invalid credentials	Valid user accepted; invalid user rejected with message
T2	Normal leave	Student submits leave; mentor approves	Status updated; history and remarks stored
T3	Emergency entry	Advisor records emergency permission	Record created with both timestamps
T4	Verification	Emergency reported with incomplete details	Pending task and dashboard alert generated
T5	Reminder	Task remains pending beyond interval	One reminder generated; no duplicate spam
T6	Login tracking	Same student logs in twice	Counted once as a unique student
T7	Login status	Student with no login	Shown as "No Login Recorded", not absent
T8	Notification	Parent alert in MOCK mode	Logged and labelled as mock
T9	Retry	Notification fails and is retried	Retry recorded; no duplicate message
T10	Authorization	Student calls a mentor-only route	Request rejected

Table 3. Representative functional test scenarios for the proposed system.

Any failed tests should be investigated and repeated after corrections are made.

3.3 Comparison with conventional approaches

Conventional approaches rely on paper forms, phone calls, and group messages. The proposed system aims to bring leave approval, emergency reporting, login monitoring, and parent notification into a common interface with an audit trail, potentially improving traceability and reducing missed records. The proposed system should not be considered more effective than conventional methods without comparative testing in a real college setting.

3.4 Discussion

The proposed architecture provides a framework for managing normal and emergency leave in a way that is realistic about what software can know. Its main strength is that an incomplete or forgotten emergency entry becomes a visible, time-stamped task instead of silently disappearing.

The system also has limitations. A departure that is never reported by an advisor or an independent source cannot be detected automatically. Login records show portal usage only and are not proof of physical presence. The usefulness of the verification workflow depends on staff responding to reminders, and automated notifications depend on the reliability of the SMS or email provider. Privacy and security are important because the system handles student and parent contact data, so authentication, authorization, and careful handling of credentials are necessary.

4. CONCLUSION

The Leave Notification System presents a technology-based approach to managing student leave, emergency permissions, login activity, and parent communication through a centralized application. The proposed system combines leave approval, an emergency reporting and verification workflow, login monitoring, and parent notification, with an audit trail that records who did what and when.

The design avoids unsupported claims. It does not assume that a student has left campus because no login exists, and it does not claim to detect unreported departures. Its practical effectiveness depends on correct implementation, staff participation, and testing of its individual components.

Future development can focus on integration with a college attendance or gate register, mobile app support, push notifications, richer analytics for administrators, and support for additional notification channels. Further testing and user evaluation are required to determine the reliability of the system in a real college environment.

5. ACKNOWLEDGMENTS

The authors would like to thank Sri Shakthi Institute of Engineering and Technology for providing the academic environment and facilities required for this project. The authors also express their sincere gratitude to the Head of the Department of Information Technology, the project guide, and the faculty members for their guidance, encouragement, and support throughout the development of the project.

6. REFERENCES

1. D., Nandy, S., Kumar, A., and Kumar, A. (2019). “Development of an Attendance System for Students with SMS Notification to Parents.” International Advanced Research Journal in Science, Engineering and Technology.
2. Bakhri, F., Mohd Ekhsan, H., and Hamid, J. N. (2020). “Students’ Attendance Monitoring System with SMS Notification.” Journal of Computing Research and Innovation, 5(1), 19–24. https://doi.org/10.24191/jcrinn.v5i1.159
3. Go, D. A. B., Deiparine, C. J. M., Cuizon, J. C., and Canonigo, B. L. (2018). “Presence: An Integrated Mobile Solution for Truancy Detection using RFID and Cloud-based Notification Services.” Journal of Science, Engineering and Technolog.
4. Abbas, Z., Ahmed, B., Abbas, J., Akbar, A., and Batool, R. (2023). “A Fingerprint Based Students Attendance System with SMS Alert to Parents.” Iraqi Journal of Information and Communication Technology.
5. Joshi, R., and Shete, V. V. (2015). “Android Based Smart Learning and Attendance Management System.” International Journal of Advanced Research in Computer and Communication Engineering.
6. Pranav Pisal et al. (2025). “An Automated Students Attendance System with SMS Notification.” International Journal of Scientific Research in Engineering and Management.
7. Node.js. Node.js Official Documentation. https://nodejs.org/docs/latest/api/
8. Express.js. Express Web Framework Documentation. https://expressjs.com/
9. Oracle. MySQL Reference Manual. https://dev.mysql.com/doc/
10. Mozilla Developer Network (MDN). Introduction to Express/Node.js. https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/Introduction
11. Mozilla Developer Network (MDN). HTML: HyperText Markup Language. https://developer.mozilla.org/en-US/docs/Web/HTML
12. Mozilla Developer Network (MDN). CSS: Cascading Style Sheets. https://developer.mozilla.org/en-US/docs/Web/CSS
13. Mozilla Developer Network (MDN). JavaScript Guide. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
14. OWASP Foundation. SQL Injection Prevention Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
15. OWASP Foundation. Node.js Security Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Nodejs_Security_Cheat_Sheet.html
16. R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.
17. I. Sommerville, Software Engineering, 10th ed. Harlow, U.K.: Pearson, 2015.