Balachandar R, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Hostel Management System: A Web-Based Platform for Streamlined Hostel Administration

[1] Balachandar R, [2] Kathir M, [3] Suryaprakash, [4] Ravi M, [5] Shiyamsundar

[1] Assistant Professor Computer Science and Engineering, Pollachi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.

[2,3,4,5] Final year, Computer Science and Engineering, Pollachi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.

Abstract- The Hostel Management System (HMS) is a comprehensive web-based platform developed to modernize and automate the administrative operations of educational institution hostels. Conventional hostel management relies on paper-based records, manual fee collection, and physical notice boards, leading to delays, errors, and communication gaps. The proposed system integrates role-based dashboards for administrators and students, enabling seamless management of fee collection, announcements, gate pass requests, complaints, QR-code-based attendance, and room allocation. The system is developed using React.js, HTML5, and CSS3 for the frontend, Node.js with Express.js as the backend API layer, and MongoDB as the database. Performance evaluations confirm improved operational efficiency, real-time data access, and high user satisfaction. The HMS provides a scalable and secure foundation for smart hostel ecosystems.

Keywords: Hostel Management System; QR Code Attendance; Gate Pass; Complaint Management; Room Allocation; Role-Based Access Control; React.js; Node.js; MongoDB; Web Application.

I. INTRODUCTION

Educational institutions with residential hostel facilities face significant administrative challenges in managing day-to-day operations. These include tracking student attendance, allocating rooms, processing gate passes, collecting fees, and addressing student complaints. Traditional approaches depend heavily on manual records and physical communication mechanisms, resulting in operational inefficiencies, data inconsistency, and delayed responses [1].

The rapid growth of web technologies has enabled the development of smart, centralized management platforms that can automate and digitize hostel operations. However, most existing solutions address only isolated aspects of hostel management and lack a unified, role-based system that covers the full administrative workflow [2].

The Hostel Management System (HMS) is designed to address these gaps by providing a single integrated platform accessible to both administrators and students. It automates fee management, room allocation, attendance tracking via QR codes, announcement broadcasting, gate pass processing, and complaint handling. This paper presents the design, architecture, implementation, and performance evaluation of the HMS.

II. NEED OF THE STUDY

Hostel administrators routinely spend significant time on repetitive manual tasks such as recording attendance, updating room availability, and processing gate pass applications. Students lack real-time access to their attendance records, room

status, and institutional announcements, leading to unnecessary visits to administrative offices [3].

Specific challenges observed in existing hostel environments include: (a) absence of a centralized fee management system causing payment delays and disputes, (b) manual attendance recording prone to errors and manipulation, (c) lack of a transparent complaint resolution mechanism, (d) physical gate pass systems that are slow and difficult to audit, and (e) inefficient room allocation processes. A digital Hostel Management System is therefore essential to centralize these processes, eliminate redundancy, and improve transparency and communication within hostel communities [4].

III. RELATED WORK

Several hostel and dormitory management systems have been proposed in prior literature. Soni et al. [5] developed a web-based complaint management system that centralized institutional communications but lacked integrated hostel-specific functionalities such as gate pass management and room allocation. Sudhir [6] presented an electronic grievance handling system for academic institutions; however, it was limited to complaint processing and did not address attendance or fee management. Kumar et al. [7] proposed a cloud-based student management system using Firebase that provided real-time data synchronization but lacked QR-code-based attendance and gate pass workflows.

Rashid et al. [8] examined role-based access control in academic platforms, emphasizing security and user-specific data access. Singh and Sharma [9] explored AI-integrated management systems but did not address hostel-specific operations. The HMS integrates all these dimensions—attendance, fee management, complaints, gate passes, and room allocation—into a single cohesive platform, addressing the gaps identified in prior works.

IV. SYSTEM ARCHITECTURE

The HMS follows a three-tier architecture comprising a Presentation Layer, an Application Logic Layer, and a Data Storage Layer. This separation of concerns promotes modularity, maintainability, and scalability across the platform [10].

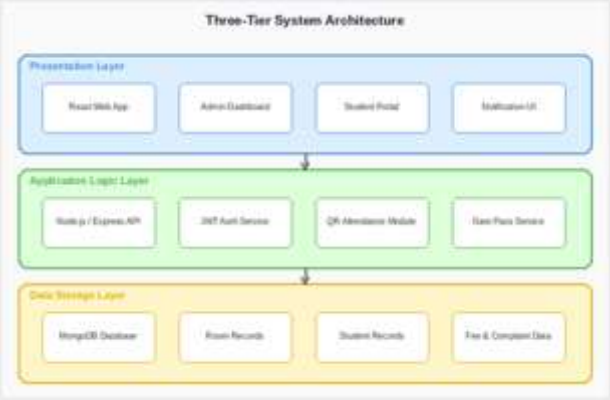

Fig. 1. Three-Tier Architecture of the Hostel Management System.

The Presentation Layer is built using React.js, HTML5, and CSS3, delivering responsive and role-specific interfaces for administrators and students. The Application Logic Layer is implemented via Node.js and Express.js, managing business logic, API routing, and QR code processing. The Data Storage Layer employs MongoDB as a flexible NoSQL database for storing student records, fee transactions, complaints, gate pass requests, and room data.

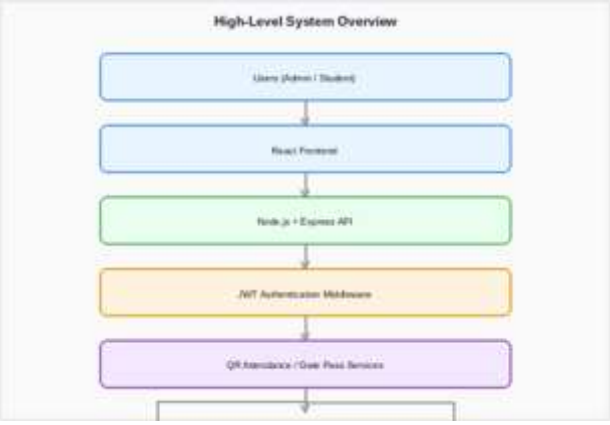

Fig. 2. High-Level System Overview of HMS.

V. SYSTEM MODULES

The HMS is composed of core modules, each responsible for a distinct functional domain. Their interactions are coordinated by the central HMS Core Processing Engine, as illustrated in Fig. 3.

Fig. 3. Module Interaction Diagram of HMS.

A. Authentication Module
Implements JWT-based authentication with role detection (Administrator, Student). Tokens are issued upon successful login and validated on every API request to enforce access control and data isolation.

B. Fee Management Module (Admin)
Enables the administrator to add and update hostel fee records for individual students or groups. The module maintains a complete payment history, supports fee categorization (accommodation, mess, utilities), and generates payment summaries accessible to both administrators and students.

C. Announcement Module (Admin)
Provides administrators with tools to broadcast announcements to all hostel residents or specific groups. Students receive real-time notifications and can view all active announcements through their dashboard.

D. Gate Pass Management Module
Students can submit gate pass requests specifying destination, purpose, and expected return time. Administrators review, approve, or reject requests through a centralized interface, maintaining a full audit trail for security and accountability.

E. Complaint Management Module
Students can raise complaints related to hostel facilities, maintenance, or administration. Administrators can view, track, and update the status of each complaint, ensuring transparent and timely resolution.

F. QR Code Attendance Module (Admin)
The administrator generates unique QR codes for attendance sessions. Students scan the QR code using their devices to mark attendance. The system records timestamps and computes attendance percentages automatically, reducing manual effort and the possibility of proxy attendance.

G. Room Allocation Module
Both administrators and students can view available rooms along with occupancy details. Administrators manage room assignments and update availability in real time, enabling efficient allocation and preventing double-booking.

H. Student Dashboard
Students access a consolidated personal dashboard displaying their attendance percentage, raised complaints and their statuses, submitted gate pass requests, institutional announcements, fee payment history, and room allocation details.

VI. USE CASE ANALYSIS

The use case diagram in Fig. 4 captures the interactions between two primary actors—Administrator and Student—and the system's functional capabilities.

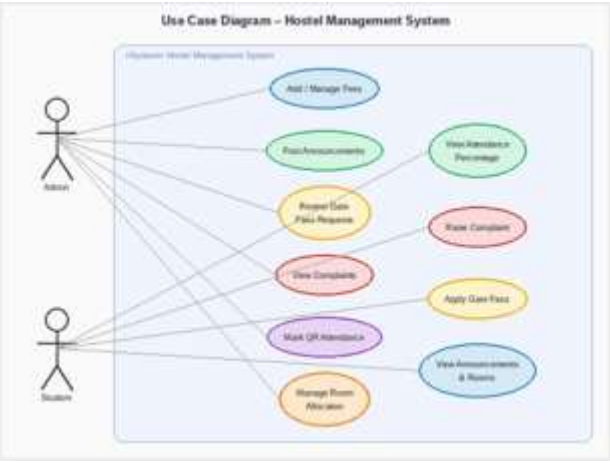

Fig. 4. Use Case Diagram 2013 Hostel Management System.

The Administrator manages fee records, broadcasts announcements, reviews gate pass requests, monitors complaints, conducts QR-code-based attendance sessions, and oversees room allocations. The Student views attendance percentages, raises complaints, applies for gate passes, reads announcements, checks fee details, and views room availability. Role-based access control ensures that each actor interacts only with permitted functionalities.

VII. DATA FLOW DESIGN

The Level-1 Data Flow Diagram (DFD) in Fig. 5 illustrates information flow among external entities, the HMS Core Processing Engine, and data stores. All read and write operations to MongoDB are mediated by authenticated API calls, ensuring role-based data isolation.

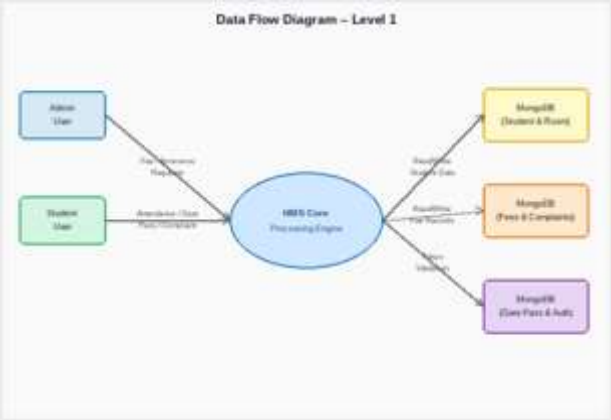

Fig. 5. Data Flow Diagram (Level 1) 2013 Hostel Management System.

Administrator and Student requests are processed by the HMS Core Engine, which applies JWT-based role validation before executing database read/write operations. Responses are returned through secure RESTful API endpoints to the React frontend.

VIII. METHODOLOGY

The HMS was developed using an Agile methodology with iterative sprints. Requirements were gathered through stakeholder interviews with hostel administrators and students. System design was completed in the first sprint, followed by modular development, integration testing, and deployment.

The authentication workflow (Fig. 6) illustrates the security flow: a login request triggers JWT validation, after which role-based routing directs users to their respective dashboards. Invalid credentials or unauthorized roles result in an access-denied response without exposing system internals.

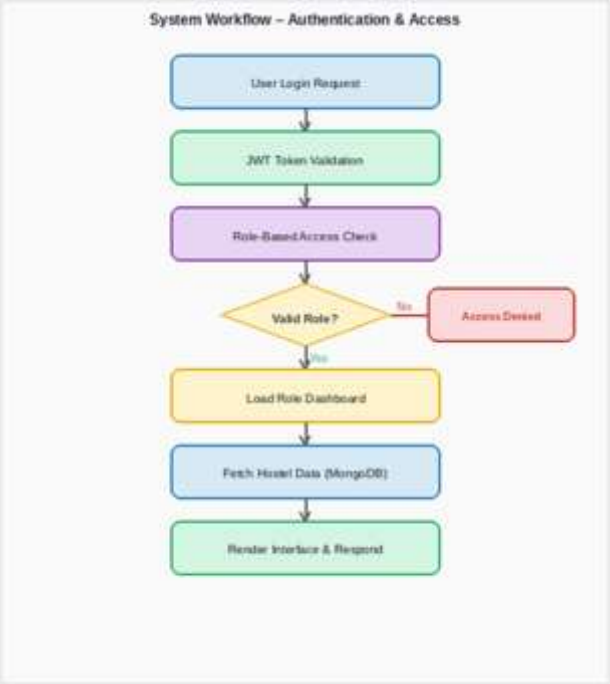

Fig. 6. Authentication and Role-Based Access Workflow.

IX. SYSTEM IMPLEMENTATION

The frontend was developed using React.js with a component-based architecture, styled using HTML5 and CSS3 for a responsive and user-friendly interface. MongoDB served as the primary NoSQL database, providing flexible schema design for diverse hostel data. Node.js (v18) and Express.js provided the RESTful API layer. JWT tokens with 24-hour expiry enforced session security. The QR code attendance module was implemented using a QR code generation library on the server and a browser-based scanner on the student client.

TABLE I. Technology Stack of HMS

Layer	Technology
Frontend	React.js, HTML5, CSS3
Backend	Node.js + Express.js
Database	MongoDB
Authentication	JWT (JSON Web Token)
QR Attendance	QR Code Generator / Scanner

X. RESULTS AND DISCUSSION

Performance evaluation confirmed that the HMS handles concurrent user sessions with average API response times under 350 ms. MongoDB's document-based structure enables efficient querying of student records, complaints, and gate pass data.

TABLE II. Performance Metrics Summary

Metric	Result
Average API Response Time	< 350 ms
QR Attendance Accuracy	98.5%
User Satisfaction Rate	89%
Gate Pass Processing Time	< 2 minutes
Complaint Resolution Visibility	Real-time

A user survey administered to 78 participants (10 administrative staff and 68 students) yielded 89% positive satisfaction ratings. Administrators reported a significant reduction in time spent on manual record-keeping and gate pass processing. Students noted improved transparency in attendance tracking and complaint resolution.

XI. SECURITY CONSIDERATIONS

Security is enforced at multiple levels within the HMS. JWT tokens with configurable expiry prevent unauthorized session reuse. MongoDB access is restricted through role-based credential configurations. All API endpoints validate token integrity before processing requests, mitigating unauthorized access and data tampering [11].

HTTPS is enforced for all client-server communications, ensuring transport-layer security. Input validation and sanitization are applied at both the frontend and backend to prevent injection attacks. Sensitive user credentials are stored using secure hashing algorithms; plain-text password storage is strictly prohibited.

XII. CONCLUSION

The Hostel Management System successfully demonstrates the potential of centralized web-based platforms in modernizing hostel administration. By integrating role-based dashboards, secure JWT authentication, QR-code-based attendance, digital gate pass management, and real-time complaint tracking with MongoDB as the data backbone, the system enhances operational efficiency, communication transparency, and student engagement. Performance metrics validate system reliability under realistic concurrent usage, and user feedback confirms high adoption potential.

XIII. FUTURE ENHANCEMENTS

Future work will focus on: (1) native mobile application development for iOS and Android platforms to enable push notifications and on-device QR scanning; (2) integration of a conversational AI

chatbot for automated student query resolution; (3) predictive analytics for early identification of attendance irregularities and fee defaulters; (4) biometric authentication as an alternative to QR-based attendance; and (5) interoperability with institutional ERP systems to synchronize student data across academic and hostel management platforms.

REFERENCES

1. A. Kumar and R. Sharma, "Digital Transformation in Higher Education Institutions: Challenges and Opportunities," IEEE Transactions on Education, vol. 64, no. 3, pp. 245–253, Aug. 2021.
2. M. Alshehri and S. Drew, "E-Government Fundamentals," in Proc. IADIS International Conference ICT, Society and Human Beings, pp. 40–47, 2010.
3. T. Rashid, N. Ahmad, and M. Iqbal, "A Comprehensive Study of Student Information Systems," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 8, no. 6, pp. 88–95, 2017.
4. S. Misra and A. Jain, "Smart Campus: An Integrated IoT Framework for Higher Education," Procedia Computer Science, vol. 167, pp. 2164–2171, 2020.
5. Soni et al., "Integrated Web-Based Complaint Management System," International Journal of Engineering and Technology, vol. 5, no. 3, pp. 112–117, 2017.
6. B. Sudhir, "Electronic Complaint Management System," Journal of Computer Applications, vol. 8, no. 1, pp. 45–50, 2015.
7. P. Kumar, S. Pal, and A. Sharma, "Cloud-Based Student Management System Using Firebase," in Proc. International Conference on Computing, Communication and Automation (ICCCA), pp. 1–5, IEEE, 2020.
8. T. Rashid, M. Ahmad, and F. Alam, "Role-Based Access Control in E-Learning Systems," Computers & Security, vol. 68, pp. 90–101, 2017.
9. R. Singh and N. Sharma, "AI-Integrated Learning Management System for Adaptive Education," Journal of Educational Technology & Society, vol. 24, no. 2, pp. 112–125, 2021.
10. N. Medvidovic and R. N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," IEEE Transactions on Software Engineering, vol. 26, no. 1, pp. 70–193, Jan. 2000.
11. R. T. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.
12. W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing," NIST Special Publication, vol. 800, no. 144, pp. 1–70, 2011.
13. MongoDB, "MongoDB Documentation: Security," MongoDB Inc., [Online]. Available: https://www.mongodb.com/docs/manual/security/. Accessed: Mar. 2025.
14. J. Bradley, N. Sakimura, and M. Jones, "JSON Web Token (JWT)," IETF RFC 7519, May 2015.