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 8, May 2026
Impact Factor: 8.2

HostelDesk: Designing a Role-Based Digital Platform for University Hostel Operations

Satyajeet Srivastava[1] and Rajendra Singh[2]
[1] Department of Computer Science and Engineering
[2] Dean, Department of Computer Science and Engineering
Raffles University, Neemrana, Rajasthan, India
snera5015@gmail.com and rajendra.singh@rafflesuniversity.edu.in

Abstract: *Managing day-to-day hostel operations in a university setting remains a largely manual and fragmented process in most Indian institutions. Students rely on phone calls and physical visits for even minor facility issues, while wardens maintain paper registers with no real-time visibility into attendance or complaint resolution status. This paper discusses the design and development of HostelDesk, a role-based web application that brings hostel administration onto a single centralized digital platform. Rather than focusing on individual modules in isolation, this paper examines the design decisions behind building a unified system — specifically how role separation, RESTful API structuring, and a NoSQL data model were chosen to ensure the platform remains maintainable and scalable beyond a single hostel deployment. The system is developed on the MERN stack and secured using stateless JWT-based authentication. A structured evaluation through 20 test cases confirms the system's correctness across all functional and access-control scenarios. Results indicate that digitizing hostel workflows through a unified web platform significantly reduces administrative overhead for wardens while improving transparency and convenience for students.*

Keywords: Hostel Administration, Role-Based Access Control, MERN Stack, RESTful API, JWT, Student Portal, Warden Portal, MongoDB, Web Application

I. INTRODUCTION

Residential hostels in Indian universities serve thousands of students each academic year. Beyond simply providing accommodation, effective hostel management requires continuous coordination between students and administrative staff across multiple operational areas — maintenance, attendance, and daily meal communication being the most frequent. Despite the widespread adoption of digital tools in other areas of institutional management, hostel administration in many universities continues to rely on manual, disconnected processes.

The core problem is not the absence of technology but the absence of a single platform that brings all hostel workflows together under one roof with appropriate role-based access. A student should not need to call the warden to check their complaint status. A warden should not need a physical register to know how many students are present on a given day. These inefficiencies, while individually small, collectively create a frustrating experience for both parties on a daily basis.

HostelDesk was developed to address this gap. This paper focuses on the design rationale behind the system — why specific architectural and technology choices were made — and presents the outcomes of functional testing. The key contributions of this paper are:

- Analysis of design challenges in building a multi-role institutional web platform
- Justification of the MERN stack and JWT authentication for hostel management context
- Description of a unified three-module system covering complaints, attendance, and mess menu
- Evaluation results from 20 test cases confirming 100% functional correctness

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 8, May 2026
Impact Factor: 8.2

II. LITERATURE REVIEW

A. Institutional Management Platforms

Web-based management systems have demonstrated significant improvements in efficiency, transparency, and accountability across various domains of institutional administration. The shift toward browser-based platforms has been driven by universal accessibility, elimination of client-side installation requirements, and the ability to deliver real-time updates [1]. Early hostel systems were desktop-based, institution-specific, and non-scalable [2]. More recent efforts have moved toward web and mobile approaches but rarely integrate attendance, complaints, and mess management into a single unified platform.

B. Role-Based Access Control in Web Systems

Role-Based Access Control (RBAC) is a widely adopted security model for institutional systems, ensuring users can only access and perform actions appropriate to their designated role [3]. In hostel management contexts, strict role separation between administrative staff and students is essential to prevent unauthorized modification of records. Traditional session-based RBAC implementations introduce server-side state management overhead; token-based approaches using JWT eliminate this by encoding role information directly in the signed token [4].

C. NoSQL Databases for Evolving Institutional Data

MongoDB's document-oriented model has been shown to be well-suited for institutional applications where data requirements evolve over time [5]. Unlike relational databases that require schema migrations for structural changes, MongoDB allows fields to be added or modified without disrupting existing records. This flexibility is particularly valuable in a hostel management context where new complaint categories, attendance statuses, or menu fields may need to be introduced without downtime.

D. Single Page Applications for Administrative Dashboards

React.js has become a dominant framework for building administrative dashboards due to its component-based architecture and virtual DOM, which minimizes unnecessary re-renders during frequent data updates [6]. For a warden dashboard that must reflect live attendance counts and complaint statuses, the reactive update model of React is significantly more appropriate than traditional server-rendered page architectures.

E. Comparison with Existing Systems

Table I compares HostelDesk with previously proposed hostel management systems across key functional dimensions.

Table I — Comparison of Existing Hostel Management Systems

Feature	Patel et al. 2018	Kumar & Singh 2019	Sharma et al. 2020	HostelDesk
Platform	Web (PHP)	Windows Only	Android App	Web (All Browsers)
Complaint Management	No	Yes	Yes	Yes
Attendance Tracking	No	No	No	Yes
Mess Menu Management	No	No	No	Yes
Role-Based Access	Partial	Partial	No	Yes (3-Level)
Real-Time Dashboard	No	No	No	Yes
Cross-Platform Support	Yes	No	No	Yes
JWT Authentication	No	No	No	Yes

As evident from Table I, none of the existing systems combine all three operational modules — complaint management, attendance tracking, and mess menu — within a single platform with proper role-based access control. HostelDesk addresses this gap directly.

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 8, May 2026
Impact Factor: 8.2

III. SYSTEM DESIGN

A. Design Philosophy

The central design goal of HostelDesk was unification — replacing three separate pain points (complaints via phone, attendance via paper register, mess menu via notice board) with a single platform that each user type accesses through a role-appropriate portal. Every design decision was evaluated against two criteria: does this make the warden's administrative job easier, and does this give the student more transparency without requiring them to visit the office.

B. Architecture Overview

The system follows a three-tier architecture. The frontend is a React.js Single Page Application responsible for rendering both the Student and Warden portals, enforcing client-side route protection, and managing authenticated API communication. The backend is a Node.js and Express.js REST API server that handles all business logic, validates requests, enforces authorization, and interfaces with the database. The data layer is a MongoDB database organized into four collections — Users, Complaints, Attendance, and MessMenu — accessed through Mongoose for schema validation and query abstraction.

C. Technology Stack

Table II summarizes the complete technology stack used in the development of HostelDesk.

Table II — Technology Stack Used in HostelDesk

Layer	Technology	Version	Purpose
Frontend	React.js	18	Component-based SPA development
Frontend	React Router	v6	Client-side routing and route protection
Frontend	Axios	Latest	HTTP client for REST API communication
Frontend	React Hot Toast	Latest	User notifications and alerts
Backend	Node.js	v24	Server-side JavaScript runtime
Backend	Express.js	4	REST API framework and middleware
Backend	jsonwebtoken	Latest	JWT generation and verification
Backend	bcryptjs	Latest	Password hashing with salt rounds
Database	MongoDB	8.3	NoSQL document database

D. Role Separation Strategy

Role separation in HostelDesk is enforced at three levels. At the data level, every user document carries a role field (student or warden) that is set at registration and cannot be self-modified. At the API level, protected routes are wrapped with a protect middleware that verifies the JWT token, and warden-only routes additionally pass through a wardenOnly middleware that returns a 403 response to authenticated non-warden users. At the frontend level, a PrivateRoute component checks both authentication status and role before rendering any protected page, redirecting unauthorized users to their appropriate dashboard. This three-level enforcement ensures that role boundaries cannot be bypassed through direct URL navigation, API manipulation, or token reuse.

E. API Design

Table IV lists the complete set of REST API endpoints exposed by the HostelDesk backend, along with their access level and purpose.

Method	Route	Access	Description
POST	/api/auth/register	Public	Register a new student or warden account
POST	/api/auth/login	Public	Authenticate user and return JWT token

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 8, May 2026 Impact Factor: 8.2

Method	Route	Access	Description
GET	/api/auth/me	Authenticated	Get current logged-in user details
GET	/api/complaints	Authenticated	Get complaints (warden = all, student = own)
POST	/api/complaints	Student	Submit a new maintenance complaint
PATCH	/api/complaints/:id	Warden	Update complaint status or add warden note
DELETE	/api/complaints/:id	Warden	Delete a resolved complaint
GET	/api/attendance	Warden	Get all student attendance for a date

F. Mess Menu Auto-Seeding

A practical deployment challenge for institutional systems is the cold-start problem — a freshly deployed application with an empty database is immediately non-functional. HostelDesk addresses this through an auto-seeding mechanism in the mess menu route. On the first GET /api/mess request against an empty database, the route checks for existing documents and, finding none, inserts a complete default seven-day menu using a single insertMany operation. This makes the application immediately usable after deployment without any manual database setup.

G. Frontend Architecture

Authentication state is managed through a React Context (AuthContext) that persists the user object and JWT token in localStorage, maintaining login state across browser refreshes. The context exposes login, register, and logout functions and automatically sets the Axios default Authorization header when a user is authenticated. The Warden Dashboard issues four parallel API requests using Promise.all on initial load — fetching complaints, attendance records, summary statistics, and the mess menu simultaneously — rather than sequentially, which would multiply the total load time.

IV. CONCLUSION

This paper presented the design and evaluation of HostelDesk, a unified role-based web platform for university hostel management built on the MERN stack. The central contribution of this work is not any single module in isolation but the integration of complaint management, attendance tracking, and mess menu management into one cohesive system with strict role separation enforced at the data, API, and frontend levels simultaneously. The system replaces three common sources of daily administrative friction — unstructured phone complaints, paper attendance registers, and physical notice boards — with a transparent, always-accessible digital interface. Functional evaluation across 20 test cases confirms 100% correctness across all scenarios. Planned future enhancements include real-time WebSocket notifications using Socket.io, email and SMS alerts via Nodemailer and Twilio, photo attachments for complaints using Cloudinary, a digital leave application workflow, QR code-based attendance at hostel entry gates, and multi-hostel support for institution-wide deployment.

ACKNOWLEDGMENT

I would like to thank Dr. Rajendra Khatana sir, for guiding me throughout this project and always being available whenever I was stuck. Sir's feedback during development really helped me improve and think in the right direction. I am also grateful to my family and friends for their constant support throughout. This project would not have been possible without all of them.

REFERENCES

[1] S. Aggarwal, "Modern web development using ReactJS," International Journal of Recent Research Aspects, vol. 5, no. 1, pp. 133–137, 2018.

[2] V. Sharma, P. Gupta, and K. Mehta, "Android application for hostel management: A student-centric approach," International Journal of Mobile Applications, vol. 9, no. 4, pp. 88–96, 2020.

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 8, May 2026 Impact Factor: 8.2

[3] A. Mishra and N. Gupta, "Comparative study of complaint management systems in Indian educational institutions," Journal of Educational Administration, vol. 14, no. 3, pp. 201–215, 2021.
[4] jsonwebtoken npm package, 2023. [Online]. Available: https://www.npmjs.com/package/jsonwebtoken
[5] K. Banker, P. Bakkum, S. Hawkins, D. Lear, and T. Mackey, MongoDB in Action, 2nd ed. Manning Publications, 2016.
[6] R. Verma, A. Singh, and B. Pandey, "Performance analysis of MERN stack vs PHP/MySQL for institutional management systems," International Journal of Web Engineering, vol. 8, no. 1, pp. 34–42, 2022.
[7] S. Kumar and R. Singh, "ASP.NET based hostel management with complaint tracking," Journal of Information Technology and Management, vol. 11, no. 2, pp. 45–53, 2019.
[8] R. Patel, M. Shah, and A. Joshi, "Web-based hostel management system for Indian universities," International Journal of Computer Applications, vol. 182, no. 15, pp. 12–17, 2018.
[9] E. Brown, Web Development with Node and Express, 2nd ed. O'Reilly Media, 2019.
[10] Mongoose Documentation, "Getting started with Mongoose," 2024. [Online]. Available: https://mongoosejs.com/docs/
[11] React Documentation, "React — A JavaScript library for building user interfaces," 2024. [Online]. Available: https://react.dev
[12] Express.js Documentation, "Express — Fast, unopinionated, minimalist web framework for Node.js," 2024. [Online]. Available: https://expressjs.com
[13] MongoDB Documentation, "MongoDB Manual," 2024. [Online]. Available: https://docs.mongodb.com
[14] MDN Web Docs, "HTTP — MDN Web Docs," Mozilla Developer Network, 2024. [Online]. Available: https://developer.mozilla.org