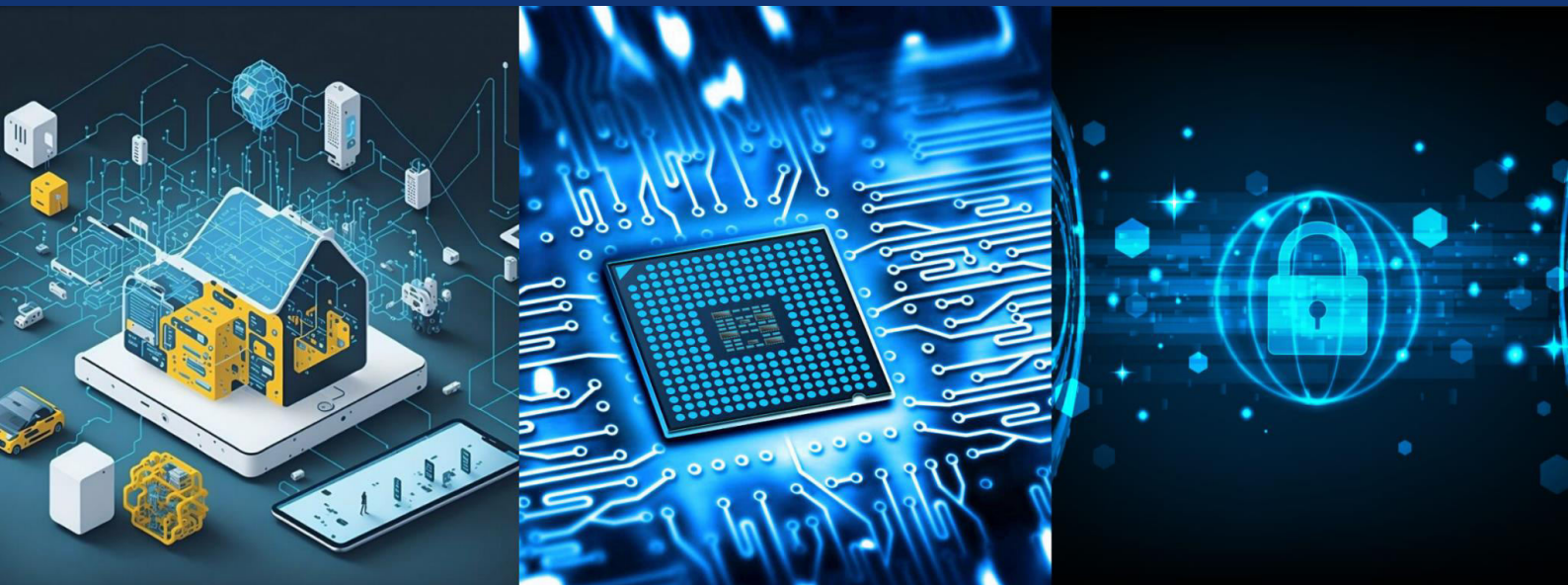
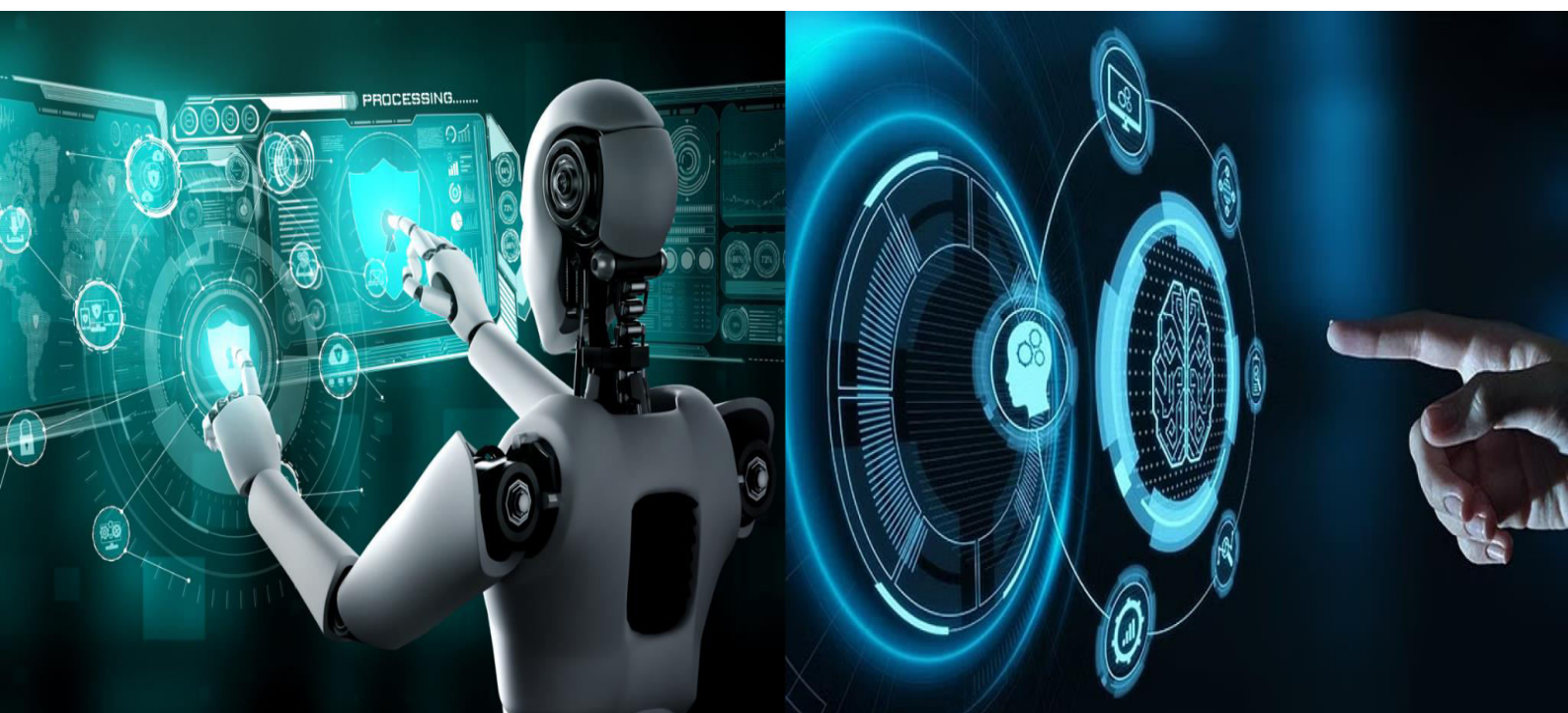




International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

HostelHub: A Progressive Web Application for Smart Hostel Management

Aishwarya Sanjeev K, Dr. Deepa A

School of Science and Computer Studies, CMR University, Bangalore, Karnataka, India

School of Science and Computer Studies, CMR University, Bangalore, Karnataka, India

ABSTRACT: Student hostels traditionally rely on manual, paper-based processes for records, leave applications, complaints and notices — slow, error-prone and non-transparent. This paper presents HostelHub, a Progressive Web Application built on the MERN stack (React/TypeScript, Node.js/Express.js, MongoDB/Mongoose) with two role-based interfaces: an administrative control centre for student, room, notice and complaint/leave management, and a student dashboard for profile, leave, complaints and canteen-menu access. Security relies on JWT authentication and Bcrypt hashing, with React Query optimizing data synchronization. As a PWA it installs like a native app and performs reliably on slow networks. Unit, integration, system and acceptance testing confirmed all major functions, reducing administrative workload and increasing transparency.

KEYWORDS: Hostel Management System, Progressive Web Application, MERN Stack, JWT Authentication, Role-Based Access Control

I. INTRODUCTION

HostelHub is a Progressive Web Application (PWA) designed to digitalize the day-to-day operations of a college hostel, bridging the communication gap between administrators and the students living there. Built as a PWA, it installs directly on mobile phones and desktops for fast loading and native-like performance, even on unstable internet connections.

Manual hostel administration suffers from several recurring problems. Physical ledgers can be misplaced, damaged, or accessed by unauthorized individuals, compromising sensitive student data. Wardens spend hours manually verifying records and leave statuses instead of focusing on core management tasks. Without digital data, administrators cannot easily analyse trends such as recurring maintenance issues or peak leave periods. Students receive no real-time status updates and must repeatedly approach staff, emergencies make it slow to determine who is on campus, and printing physical forms for minor requests is wasteful.

The objectives of HostelHub are to digitalize student and room records in a secure, centralized database; centralize communication through a digital notice board; provide a real-time ticketing system for maintenance complaints; replace paper leave forms with a transparent digital approval workflow; make daily information such as the canteen menu accessible from a phone; and reduce administrative workload so wardens can focus on hostel management rather than paperwork. Within scope are student/room information management, a digital notice board, complaint tracking, a leave application/approval workflow and canteen-menu updates, all delivered as a responsive PWA; online fee collection, visitor-gate tracking, biometric hardware integration and asset tracking are identified as candidates for future extension.

A PWA was chosen over a native mobile application because a single codebase runs consistently across Android, iOS, desktop and browser while still supporting home-screen installation, offline caching of key views and push-style notifications — reducing development and maintenance overhead within the limited timeline of a capstone project, while still delivering an app-like experience for both students and administrators.

II. LITERATURE SURVEY

Early hostel-management systems relied on manual, paper-based processes that were time-consuming, error-prone and difficult to scale. Standalone computer programs, followed by web-based systems that centralized data storage, improved record-keeping but were designed primarily for administrators, offering students little or no direct access.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Recent studies show growing adoption of cloud-based, multi-device systems: React and Node.js provide fast, scalable architectures, while Progressive Web Apps add offline capability, push notifications and consistent cross-device presentation. Forecasting techniques are increasingly applied to facility management — analysing repair records and occupancy trends to anticipate maintenance needs, estimate occupancy and plan food procurement. Artificial Intelligence is also reshaping hostel management through NLP-based chatbots, machine-learning models that triage maintenance issues, and IoT/camera-based systems supporting safety and occupancy tracking.

Table I summarizes how the conventional manual approach compares with HostelHub across key dimensions.

TABLE I. Manual System vs. HostelHub

Feature	Manual System	HostelHub
Data Storage	Paper/Excel, loss-prone	Centralized MongoDB
Leave Mgmt.	Physical signatures	Digital, real-time tracking
Complaints	Verbal/written, often lost	Ticketing with image upload
Accessibility	Office hours only	24/7, any device
Data Retrieval	Manual file search	Instant search & charts
Communication	Notice board, easily missed	Real-time digital alerts
Security	Low, open physical access	JWT auth, RBAC, hashing

Across the surveyed work, three trends emerge: a shift from desktop to cloud/web architectures, growing interest in predictive analytics for resource planning, and early exploration of AI-driven automation. Few systems, however, combine all of these with a genuinely student-facing interface — most remain administrator-centric with reporting added afterward. Prior systems also typically neglect timely, bidirectional communication, track complaints only one-directionally, leave digital leave authorization unsupported, and remain inconsistent in usability across devices. HostelHub addresses these gaps by combining real-time notifications, document-backed leave approval, visual complaint tracking and a responsive interface within one accessible platform.

III. METHODOLOGY / APPROACH

A. Requirements and Feasibility

Functional requirements cover secure, role-based Authentication for Students and Admins; a Student Module with profile/room/notice viewing, leave application (dates, reason, PDF attachment) and complaint raising (image attachment, status visibility); and an Admin Module with dashboard analytics, student-record management, leave review with approve/reject actions, complaint resolution tracking and notice/emergency-banner management. Non-functional requirements target performance (Vite-based loading, React Query state synchronization), security (Mongoose ODM against NoSQL injection, JWT authentication, Bcrypt hashing), reliability (consistent status tracking for leaves/complaints), usability (responsive Material UI/Tailwind interface) and maintainability (modular React components, separated Express routes).

A feasibility study found the project technically feasible (mature, well-supported MERN stack runnable on standard hardware), economically feasible (free open-source tools; free-tier hosting via MongoDB Atlas/Vercel/Render) and operationally feasible (an intuitive Material UI design usable without specialized training).

B. System Design

The system design (Fig. 1) models Student and Admin actors and their use cases — profile/dashboard access, leave application and status tracking, complaint submission and tracking, notice viewing and hostel-information download for students; and student, room, leave and complaint management plus notices and reporting for administrators. Supporting Class, Sequence, State-Transition, Entity-Relationship and Data-Flow diagrams guided the object, schema and process



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

design, including how a Leave Application or Complaint moves between Pending, Approved/Resolved and Rejected states.

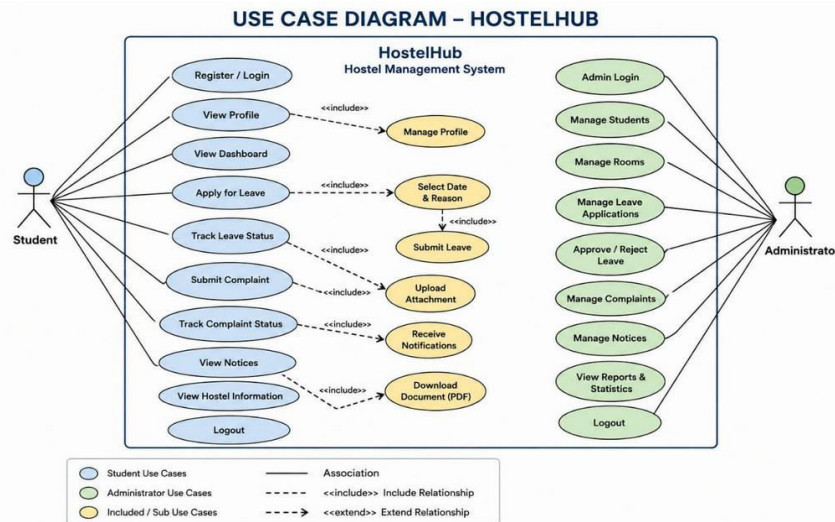


Fig. 1. Use case diagram — Student and Admin actors.

C. Database Schema

The MongoDB schema (via Mongoose ODM) is organized around six primary collections. User stores common authentication fields (email, hashed password, role) shared by Students and Admins. Student extends user data with personal details, hostel block and an allocated Room reference. Room records room number, block, capacity and current occupancy. LeaveApplication stores the requesting student, from/to dates, reason, an optional attachment path and a status field (Pending, Approved, Rejected). Complaint stores the reporting student, description, optional image attachment and a status field (Open, In-Progress, Resolved). Notice stores announcement content, publish date and an emergency flag.

D. Implementation and Modules

Implemented using the technology stack in Table II, the frontend organizes React 18 components by feature (authentication, dashboard, leave, complaints, notices, rooms) with reusable components and custom data-fetching hooks. React Query manages server-state caching, background refetching and optimistic updates. The backend exposes a modular REST API through Express.js route modules, each backed by a corresponding Mongoose model and controller.

TABLE II. Technology Stack by Layer

Layer	Technology
Frontend	React 18, TypeScript, Vite
UI	Material UI, Tailwind CSS
Backend	Node.js, Express.js
Database	MongoDB, Mongoose ODM
Auth/Security	JWT, Bcrypt
File Upload	Multer

Core modules are: Authentication (JWT session security, Bcrypt password encryption); Student Management (personal details, hostel block, room allocation); Admin Dashboard (live counts of students, pending leaves, open complaints and



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

room occupancy); Complaint Management (image-attached tickets with real-time status); Leave Management (apply/approve/reject with live status updates); Room Management (number, block, capacity, occupancy); and Notification, Report-Generation and Database-Management modules.

Security follows defence-in-depth: passwords are salted and hashed with Bcrypt before persistence; every protected API route validates a JWT whose payload encodes the user's role, letting Express middleware reject cross-role requests; Mongoose schema typing and parameterized queries mitigate NoSQL injection; and Multer-handled file uploads are restricted by type and size.

IV. RESULTS AND DISCUSSION

HostelHub was validated across four stages: Unit Testing of components such as authentication logic, form validation and API endpoints; Integration Testing of the React–Express–MongoDB interaction; System Testing of complete end-to-end workflows across modules; and Acceptance Testing against stakeholder requirements. All eight system-level scenarios (Table III) and all eight acceptance-test cases (Table IV) returned a Pass result.

TABLE III. System-Level Test Results

Test Function	Scenario	Result
User Login	Valid credentials	Pass
Student Mgmt.	View student details	Pass
Complaints	Submit complaint	Pass
Leave Mgmt.	Apply for leave	Pass
Notices	View notices	Pass
Canteen Menu	View updated menu	Pass
Auth Guard	Access protected page	Pass
Logout	Session termination	Pass

TABLE IV. Acceptance Test Results

User	Acceptance Criteria	Result
Student	Register and log in	Pass
Student	Submit leave application	Pass
Student	Register a complaint	Pass
Student	View notices and menu	Pass
Admin	Manage student records	Pass
Admin	Approve leave / resolve complaints	Pass
Admin	Publish notices, update menu	Pass
All Users	Navigate all modules	Pass



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

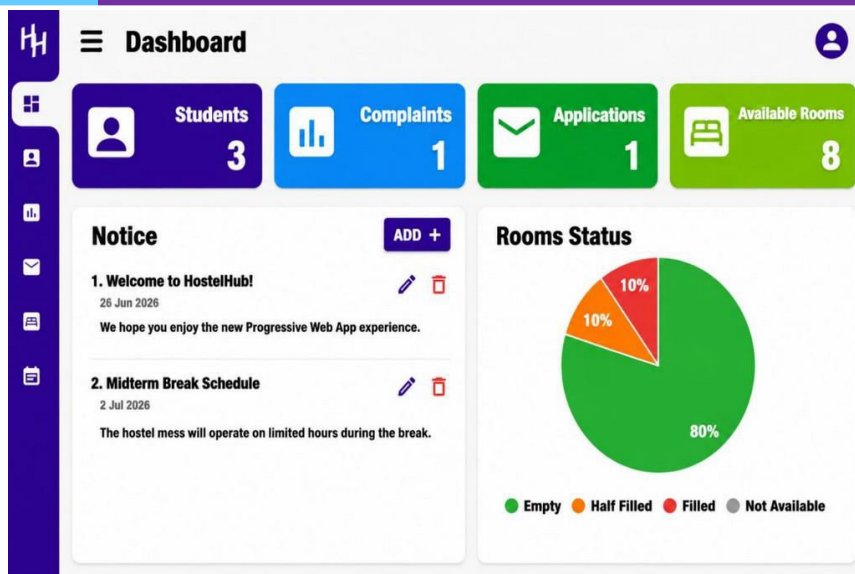


Fig. 2. Admin dashboard showing live hostel statistics.

Screenshots collected during testing show a clean, responsive interface: login screens enforce role separation, the admin dashboard (Fig. 2) surfaces live counts of students, pending leaves and open complaints together with a room-occupancy chart, and the student views for leave application, complaint raising and status tracking rendered correctly across desktop and mobile breakpoints.

The results indicate that a MERN-based PWA is a practical fit for institutional hostel management: the stack's maturity kept implementation time low while still supporting role-based dashboards, file attachments and real-time status tracking. Compared with the manual process it replaces, HostelHub shortens the leave-approval and complaint-resolution cycle from a multi-step, in-person process to a few clicks, and gives administrators visibility that was previously unavailable without manually tallying paper records. The main trade-off observed during development was scope control — the project deliberately limited itself to the core communication and workflow gaps identified in the literature survey rather than attempting payments, biometrics and visitor logs at once.

V. CONCLUSION

HostelHub was successfully designed and implemented as a Progressive Web Application that digitalizes student and room records, centralizes communication through a digital notice board, and automates leave approval and complaint resolution with secure JWT/Bcrypt authentication and role-based access control, delivered through a responsive React/Material UI/Tailwind interface. Extensive unit, integration, system and acceptance testing confirmed stable performance, accurate data handling and a consistent experience across devices.

Current limitations are that the system requires a stable internet connection (complete offline functionality is not yet supported), is designed for single-hostel rather than multi-hostel deployment, has no dedicated native mobile app, and does not yet integrate online fee payment, SMS/email notifications or biometric attendance. Planned future work includes native mobile applications with automated fee-payment receipts, QR-code/biometric attendance with digital visitor gate-passes, AI chatbot support with smart room allocation and an advanced analytics dashboard, and multi-hostel management integrated with college ERP systems.

REFERENCES

- [1] React Documentation, React.dev, 2024.
- [2] Vite Documentation, vitejs.dev, 2024.
- [3] Node.js Documentation, nodejs.org, 2024.



International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- [4] MongoDB / Mongoose Documentation, mongodb.com, 2024.
- [5] Material UI Documentation, mui.com, 2024.
- [6] Tailwind CSS Documentation, tailwindcss.com, 2024.
- [7] TanStack React Query Documentation, tanstack.com, 2024.
- [8] Google Developers, Progressive Web Apps Guide, web.dev, 2024.
- [9] Mozilla Developer Network, Web APIs Documentation, developer.mozilla.org, 2024.
- [10] Express.js Documentation, expressjs.com, 2024.
- [11] Axios Documentation, axios-http.com, 2024.
- [12] React Router Documentation, reactrouter.com, 2024.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

 9940 572 462  6381 907 438  ijircce@gmail.com



www.ijircce.com

Scan to save the contact details