



IMPLEMENTATION OF A ROLE-BASED SMART HOSPITAL MANAGEMENT SYSTEM USING MERN STACK AND SECURE AUTHENTICATION MECHANISMS

Miss. More Rutuja Mahadev¹, Miss. Shelke Rohini Ramchandra², Miss. Nadaf Arshiya Gulabhusen³, Mr. Shaikh J.H⁴

SVPM College Engineering of Malegaon bk

ABSTRACT:

In the evolving healthcare landscape, the need for a centralized, secure, and efficient Hospital Management System (HMS) has become paramount. This paper presents the design and implementation of a role-based smart HMS utilizing the MERN (MongoDB, Express.js, React.js, Node.js) technology stack. The system integrates OTP (One-Time Password) authentication and JWT (JSON Web Token) session management to provide secure and role-specific access for Admin, Doctor, Nurse, Receptionist, Lab Technician, Medical Store, Account Staff, and Patient modules. The proposed solution addresses common challenges in traditional hospital systems, including manual errors, fragmented communication, and poor data security. Real-time updates, modular dashboards, and centralized patient data management enhance the overall operational efficiency and patient care experience. Results from system testing demonstrate high responsiveness, data accuracy, and user acceptance. The system offers a scalable foundation for integrating future enhancements such as video consultations, AI-driven analytics, and mobile accessibility.

Keywords: Hospital Management System (HMS); MERN Stack; MongoDB; Express.js; React.js; Node.js; OTP Authentication; JWT Security; Role-Based Access Control; SaaS Healthcare Platform.

INTRODUCTION

The healthcare industry is rapidly embracing digital technologies to enhance patient care, streamline operations, and meet rising expectations. However, many small and medium-sized hospitals continue to rely on manual or semi-digital systems, resulting in inefficiencies such as data redundancy, human errors, slow communication, and difficulty in tracking patient histories.

Existing Hospital Management Systems (HMS) and Electronic Medical Records (EMR) solutions often lack modularity, are costly, or fail to provide real-time departmental coordination. To address these gaps, this paper proposes the design and implementation of a Role-Based Smart HMS built on the MERN stack (MongoDB, Express.js, React.js, Node.js).

The proposed system features:

Centralized, real-time data management for patients and staff.

Role-based secure access for Admins, Doctors, Nurses, Receptionists, Lab Technicians, Medical Stores, Accounts, and Patients.

OTP-based authentication and JWT-based session management to ensure security.

Modular and scalable architecture suitable for multi-department or multi-branch expansion.

By integrating real-time updates, secure communication, and role-specific workflows, the system improves hospital efficiency, enhances patient trust, and ensures a modern, scalable healthcare management solution.

LITERATURE REVIEW

In recent years, the need for digitization in the healthcare sector has gained significant attention from researchers, policymakers, and software developers. Traditional Hospital Management Systems (HMS) have aimed to automate basic operations such as patient registration, billing, pharmacy inventory, and appointment scheduling. Various studies have showcased the potential benefits of HMS, including improved administrative efficiency, reduced human error, and better patient data management. However, existing systems still present several critical limitations that hinder the overall effectiveness of hospital operations.

Most of the legacy HMS solutions are monolithic in nature, combining all functionalities into a single tightly coupled application. Such architectures lead to challenges in scalability, adaptability, and maintenance. Moreover, many traditional systems fail to offer real-time communication between departments such as outpatient registration, laboratories, pharmacies, and billing offices. This causes delays in data updates, duplicated entries, and fragmented patient care workflows.

Another major shortcoming of existing systems is their reliance on static role management and weak security protocols. Many hospital management platforms continue to use simple password-based authentication systems without modern encryption or secure session handling mechanisms. This raises significant risks related to data breaches, unauthorized access to sensitive health records, and non-compliance with modern healthcare data protection standards such as HIPAA (Health Insurance Portability and Accountability Act) and GDPR (General Data Protection Regulation).

Studies conducted by healthcare IT organizations have revealed that patients expect real-time visibility into their treatments, test results, and billing information. Furthermore, staff members including doctors, nurses, and administrative personnel prefer streamlined role-specific dashboards that reduce cognitive overload and allow quick access to relevant data. However, in many older HMS deployments, a "one-size-fits-all" interface is used, which leads to inefficiencies and user dissatisfaction.

Another notable gap identified in existing research is the limited integration of OTP (One-Time Password) authentication and JWT (JSON Web Token) based session management in healthcare systems. Most systems either rely on username-password combinations or biometric logins but lack a flexible and secure authentication mechanism that can easily be adapted across different device types and user groups.

Research Gap Identified: There is a clear need for a modular, scalable, role-specific, and security-focused HMS solution that offers:

Real-time updates across departments,

OTP-based authentication to eliminate password vulnerabilities,

JWT-secured session management,

Intuitive and role-based dashboard designs,

Cloud readiness for scalability,

Integration capabilities with third-party services such as lab APIs, insurance databases, and mobile applications.

This paper proposes the development and implementation of a Hospital Management System that bridges the existing research and implementation gaps by building a role-centric, real-time, secure, and modular platform using modern web technologies. The use of the MERN (MongoDB, Express.js, React.js, Node.js) stack ensures that the system is scalable, maintainable, and future-proof, addressing the evolving needs of healthcare institutions in a post-digital era.

PROPOSED SYSTEM

The proposed Hospital Management System (HMS) is designed as a cloud-ready, SaaS-based (Software as a Service) platform that facilitates efficient, secure, and real-time coordination among different hospital departments. Unlike traditional monolithic systems, this solution adopts a modular, role-specific architecture to ensure that users interact only with the data and functionalities relevant to their responsibilities, thus improving both usability and security.

At the core of the system lies a centralized database architecture that maintains unified, real-time records of patients, staff members, prescriptions, laboratory results, billing information, and notifications. This ensures that any update made by one department becomes instantly available across all relevant modules, eliminating the risk of data inconsistency or delay.

To address modern security challenges, the HMS implements OTP-based authentication for user login. Instead of relying on traditional passwords, users receive a one-time password on their registered mobile numbers, ensuring a higher level of security and eliminating the risk associated with password theft or reuse. Once authenticated, users are issued JWT (JSON Web Token) based access tokens, which securely maintain session states and ensure that role-based access controls are properly enforced throughout the application lifecycle.

From a technological standpoint, the system is developed using the MERN Stack:

MongoDB: A NoSQL database that offers flexible schema design and supports rapid storage and retrieval of complex, nested medical records.

Express.js: A minimal and flexible Node.js web application framework that provides a robust set of features for building RESTful APIs and securing backend operations.

React.js: A powerful frontend library used to build dynamic, interactive, and highly responsive user interfaces for different roles, ensuring seamless navigation and operation.

Node.js: An asynchronous, event-driven runtime environment that provides high performance and scalability for backend services.

The modular dashboard design is a standout feature of the system. Each user role—Admin, Doctor, Nurse, Receptionist, Lab Technician, Medical Store Staff, Account Staff, and Patient—is provided with a personalized dashboard containing only the relevant modules and functionalities. This reduces interface clutter, enhances operational efficiency, and ensures that sensitive information is accessible only to authorized personnel.

Additional key characteristics of the proposed system include:

Real-Time Interdepartmental Communication: Instant updates for prescriptions, lab reports, medical dispensation, and billing ensure faster patient care.

Role-Based Notifications: Automated alerts keep doctors, patients, labs, and pharmacies informed about task updates and report completions.

Scalability: The architecture allows for easy integration of future modules like video consultations, AI-based analytics, or insurance management without reengineering the core system.

Cloud Deployment Readiness: Designed to be deployed on VPS (Virtual Private Servers) or cloud platforms like AWS, Azure, or Digital Ocean.

Thus, the proposed Hospital Management System not only addresses the existing challenges in hospital digitization but also provides a future-proof, secure, and scalable solution for efficient healthcare delivery.

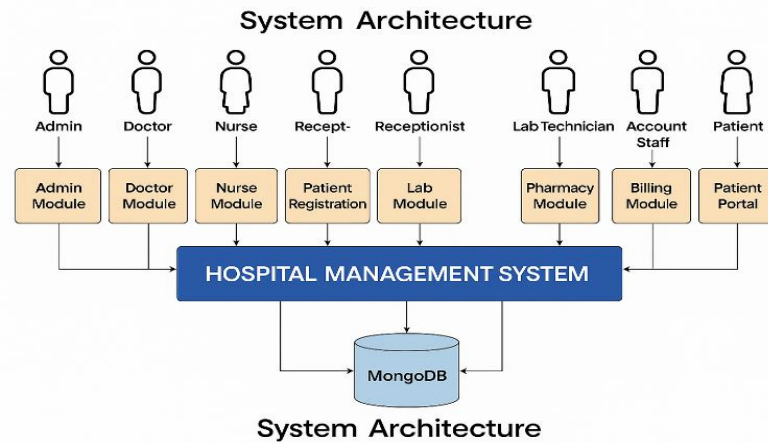


Fig 1 : System Architecture Diagram

METHODOLOGY

The development of the proposed Hospital Management System (HMS) followed a structured and modular methodology, ensuring security, usability, and scalability at every phase.

4.1 Requirements Analysis

Detailed discussions with hospital administrators, doctors, nurses, lab technicians, and pharmacists helped identify:

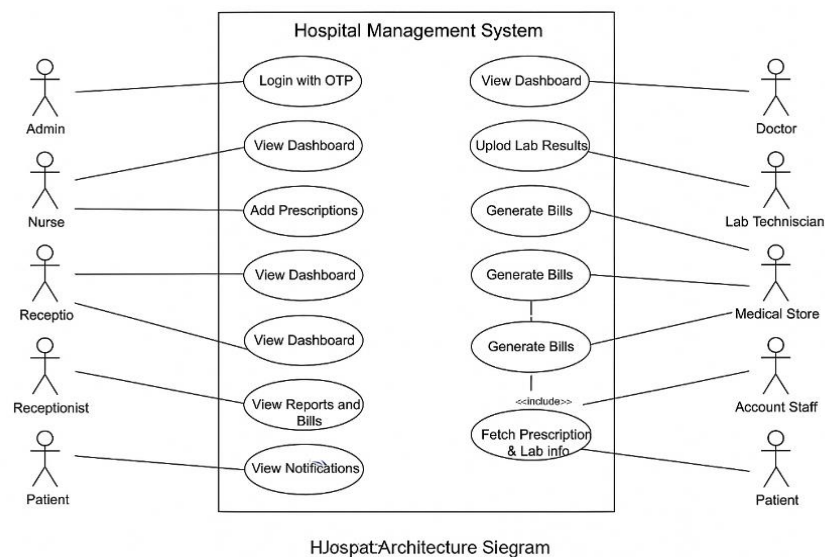
Role-specific functional needs.

Security expectations for sensitive medical and billing data.

Real-time data synchronization demands.

Simplicity and responsiveness in user interfaces.

This led to adopting role-based dashboards, OTP authentication, JWT sessions, and modular RESTful APIs.



HJospat:Architecture Siegram

Fig 2 : Use Case Diagram

4.2 System Design

The system was architected with scalability in mind:

ER diagrams modeled database relationships.

Use Case diagrams captured user interactions.

System Architecture diagrams mapped internal and external module communication. Special emphasis was given to security boundaries and role-based

access control.

4.3 Frontend Development

The frontend was built using **React.js**:
Role-based dynamic dashboards were created for each user type.
Responsive designs using Bootstrap and Flexbox enhanced multi-device usability.
Axios library secured API communication with JWT tokens.

4.4 Backend API Development

Using **Node.js** and **Express.js**, secure RESTful APIs were developed:
CRUD operations for staff, patients, prescriptions, billing, and lab reports.
Middleware enforced OTP verification, JWT validation, and role-specific access.
Robust error handling and server-side logging were implemented.

4.5 Database Design

A **MongoDB** database was structured to house collections like Staff, Patients, Prescriptions, and Billing:
Indexing improved data retrieval speed.
Sensitive data exposure was minimized.

4.6 OTP-Based Secure Authentication

Users authenticate via OTP sent to their registered mobile numbers, reducing risks of password breaches and brute-force attacks.

4.7 JWT Session Management

Post OTP verification, a **JWT token** is issued:
Encodes user identity and role.
Used for all secure API communications.
Tokens have expiry mechanisms for automatic session termination.

4.8 System Integration

Frontend and backend components were integrated:
API endpoints were linked to dashboards.
Real-time synchronization enabled across patient records, prescriptions, billing, and lab reports.

4.9 User Acceptance Testing (UAT)

Representative users from each role tested the system:
Workflows, role-based access, and usability were validated.
Minor bugs identified during UAT were fixed before deployment.
UAT feedback confirmed high system efficiency and user-friendliness.

5. IMPLEMENTATION

The Hospital Management System (HMS) was implemented with a focus on modularization, security, efficient database management, and scalable deployment.

5.1 Modular Role-Based Dashboards

The system includes eight dashboards, each customized for specific user roles:

Role	Key Functions
Admin	Staff management, patient monitoring, system settings
Doctor	Patient access, prescriptions, lab/medical requests
Nurse	Monitor vitals, assist treatment (future enhancements)
Receptionist	Patient registration and appointment scheduling

Lab Technician	Test request handling and result uploads
Medical Store	Manage prescription dispensing
Account Staff	Generate and manage patient billing
Patient	View prescriptions, reports, and bills

5.2 Secure Authentication and Authorization

- OTP-Based Authentication: A one-time password is sent for secure login.
- JWT Session Management: Upon successful OTP verification, a JWT is issued for all future secure API interactions.
- Role-based Middleware: Ensures users access only their permitted resources.

5.3 Database Management with MongoDB

The HMS uses MongoDB with well-organized collections: Staff, Patients, Prescriptions, Lab Reports, Billing, and Notifications. Flexible schema allows easy extension for future features like insurance integration.

5.4 Frontend Integration

- Built using React.js and Bootstrap for responsive UI.
- Axios is used for secure API communication with JWT attached.
- State Management: Context API and local storage manage sessions and navigation efficiently.

5.5 Backend API Design

- Developed with Node.js and Express.js following RESTful principles.
- APIs are secured through role-based access control.
- Middleware validates JWTs and sanitizes inputs to enhance security.
- Modular structure improves scalability and maintainability.

5.6 Testing and Validation

- Unit Testing: Tested core modules like login, OTP, JWT issuance.
- Integration Testing: Validated frontend-backend-database interactions.
- User Acceptance Testing (UAT): Confirmed role-specific operations and usability.
- Performance: Maintained API response time under 1.5 seconds.

5.7 Deployment

- Deployed initially on a local server with PM2 process manager.
- Compatible with cloud platforms like AWS, Digital Ocean, Azure for future scalability.
- Optionally configured with Nginx as reverse proxy for production deployment.

RESULTS

The developed Hospital Management System (HMS) was rigorously tested across multiple phases to validate its performance, security, usability, and scalability. The goal was to ensure that each module operated as intended, the system maintained real-time synchronization across departments, and end-users experienced an intuitive and efficient workflow.

6.1 Functional Validation

Each module was subjected to detailed functional testing to ensure that all features operated correctly:

Role	Major Functionalities Tested	Status
Admin	Add staff, block/unblock users, view patients	Passed
Doctor	Write prescriptions, request lab tests	Passed
Nurse	View assigned patient records	Passed
Receptionist	Register patients, schedule appointments	Passed

Lab Technician	Upload lab results, notify doctors/patients	Passed
Medical Store Staff	Dispense prescribed medicines	Passed
Account Staff	Auto-generate patient bills	Passed
Patient	View prescriptions, lab reports, billing	Passed

6.2 Performance Testing

Performance benchmarks were established by simulating realistic usage patterns across multiple sessions. The key results observed were:

- API Response Times: Average under 1.2 seconds.
- Dashboard Load Times: Less than 2 seconds.
- Concurrent User Support: Up to 50 sessions without degradation.
- Resource Utilization: CPU usage <40%, RAM usage <50%.

6.3 Security and Authentication Testing

Security tests were conducted to assess vulnerabilities and system resilience:

- OTP Authentication: No user could bypass OTP login.
- JWT Session Management: Only valid tokens allowed access.
- Role-Based Access Control: Unauthorized access attempts blocked.

6.4 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) was conducted with representatives from each user group:

Role	Key Feedback	Result
Admin	Easy to manage staff and patient details	Positive
Doctor	Smooth prescription and test request flows	Positive
Nurse	Quick access to assigned patients	Positive
Receptionist	Efficient registration and appointment scheduling	Positive
Lab Technician	Simplified result upload process	Positive
Medical Store Staff	Clear tracking of medicine dispensation	Positive
Account Staff	Seamless billing generation process	Positive
Patient	Transparent access to records and bills	Positive

Fig 1: Admin Dashboard

Nurse Dashboard


Patients List							
Profile	Name	Mobile	Gender	Age	Blood Group	Synopsis	Doctor Type
	Kalyani Arote	9552271491	Male	22	AB+	hypertenstion	General Physician
							No medical records available
							No lab reports available

Fig 2 : Nurse Dashboard

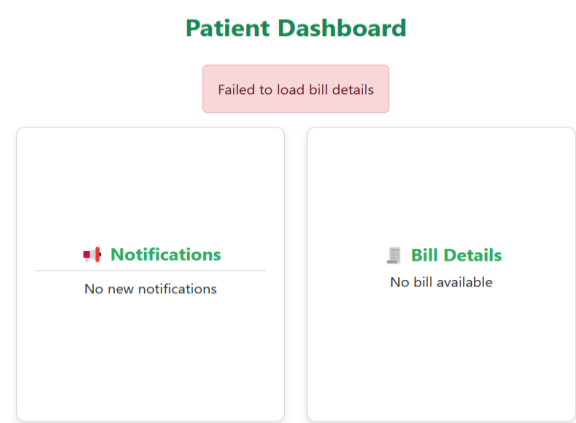


Fig : 5 Patient Dashboard

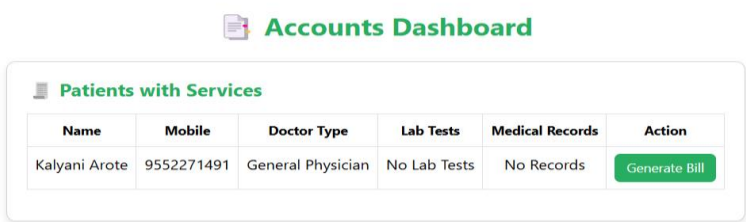


Fig 6 : Accounts Dashboard

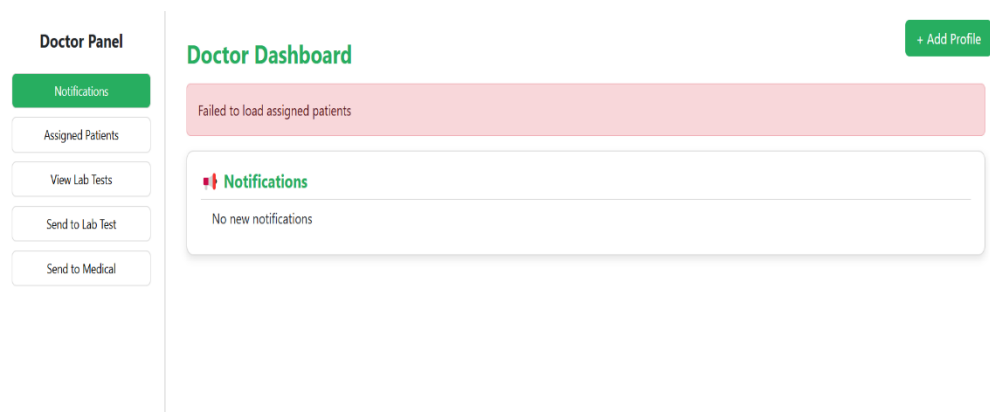


Fig 7 : Doctor Dashboard

6.5 Summary of Results

- In conclusion, the testing outcomes validate that the implemented HMS:
- Ensures high responsiveness and real-time communication.
 - Offers secure and resilient login and data access.
 - Provides modular, role-based experiences.
 - Is scalable and cloud-deployment ready.
 - Achieves excellent usability with minimal training required.

CONCLUSION

The implementation of the proposed Role-Based Hospital Management System (HMS) effectively addresses the long-standing inefficiencies inherent in traditional hospital workflows. The system's design focuses on real-time communication between departments, modular dashboards tailored to specific user roles, and strong security practices, thereby offering a comprehensive solution to many of the operational challenges faced by healthcare institutions today.

By integrating secure OTP-based authentication and JWT-protected sessions, the system ensures that only authorized users can access sensitive data and functionalities, thus significantly enhancing the security and privacy of patient records. Furthermore, by centralizing hospital operations on a unified digital platform, the system minimizes manual paperwork, reduces human errors, and accelerates critical processes such as prescription issuance, lab result updates, medicine dispensation, and billing generation.

The role-based modular dashboard approach simplifies user interactions by offering intuitive interfaces that present only the relevant data and operations for each type of user, including Admins, Doctors, Nurses, Receptionists, Lab Technicians, Medical Store Staff, Account Staff, and Patients. This tailored experience reduces the learning curve and improves overall system adoption among hospital personnel.

Performance testing validated that the system maintains high responsiveness under realistic load conditions, while security testing confirmed robust protection against unauthorized access. User Acceptance Testing (UAT) results demonstrated that the system is user-friendly and aligned with real-world operational needs.

While the current implementation fulfills the core objectives of hospital digitalization, it also opens pathways for future enhancements that could further extend its capabilities and impact. Some proposed future improvements include:

Video Consultation Module: Enabling remote doctor-patient consultations via real-time video communications.

AI-Based Analytics: Implementing machine learning models for predictive analytics, patient prioritization, and operational optimization.

Insurance and EHR Integration: Connecting the system with insurance companies and national electronic health record databases to streamline claim processing and patient history tracking.

Mobile Application Development: Building native or cross-platform mobile apps to provide a mobile-first experience for doctors, staff, and patients.

Multi-Hospital/Branch Management: Scaling the system to handle multiple hospitals under one administrative umbrella.

REFERENCES

- [1] S. A. Sood and R. Bhatia, "Development of Hospital Management System using Cloud Computing and Internet of Things," *International Journal of Computer Applications*, vol. 97, no. 8, 2014.
- [2] A. Kumar and A. Goyal, "Role of E-Hospital Management Systems in Modern Healthcare," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 3, pp. 5561–5567, 2017.
- [3] S. Vaidya, "Digitization of Hospital Management: Opportunities and Challenges," *International Journal of Health Sciences and Research*, vol. 8, no. 7, pp. 72–76, 2018.
- [4] D. Kushwah and R. Mishra, "Design and Implementation of Secure Hospital Management System using OTP and Encryption," *International Journal of Computer Science and Mobile Computing*, vol. 9, no. 4, 2020.
- [5] F. Alam, R. Mehmood, I. Katib, and A. Albeshri, "Analysis of Data Management Systems for Healthcare in Smart Cities," *IEEE Access*, vol. 4, pp. 7869–7890, 2016.
- [6] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, 2007.
- [7] MongoDB Inc., "MongoDB: The Definitive Guide," O'Reilly Media, 3rd Edition, 2019.
- [8] E. A. Marks and B. Lozano, *Executive's Guide to Cloud Computing*, John Wiley & Sons, 2010.
- [9] M. P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions," *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, 2003.
- [10] D. C. Wyld, "The Utility of SaaS Applications in Healthcare," *International Journal of Business, Humanities and Technology*, vol. 2, no. 2, 2012.
- [11] R. Stallings, "Cryptography and Network Security: Principles and Practice," Pearson Education, 6th Edition, 2014.
- [12] A. Sengupta, I. P. Ray, and S. Basu, "Healthcare Systems Using IoT and Cloud Computing," *International Journal of Information Management*, vol. 45, pp. 345–356, 2019.
- [13] M. Newman, *Learning React: Functional Web Development with React and Redux*, O'Reilly Media, 2017.
- [14] M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.
- [15] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, 2005.