Hospital Control Portal: A Secure MERN-Based Intranet Management System for Centralized Hospital Operations

P. Logaiyan[1] , S. Janith Kumar[2]

[1] Associate Professor, Department of MCA, Sri Manakula Vinayagar Engineering College (Autonomous), Puducherry 605107, India

[2] Post Graduate Student, Department of MCA, Sri Manakula Vinayagar Engineering College (Autonomous), Puducherry 605107, India

How to Cite this Article:

Kumar, S. J. (2026). Hospital Control Portal: A Secure MERN-Based Intranet Management System for Centralized Hospital Operations. International Journal of Creative and Open Research in Engineering and Management, 02(6). https://doi.org/10.55041/ijcope.v2i6.157

https://doi.org/10.55041/ijcope.v2i6.157

Abstract

— The healthcare sector demands robust, secure, and efficiently coordinated information systems to sustain quality patient care and administrative effectiveness. Conventional approaches that rely on fragmented paper-based records and standalone departmental tools introduce systemic inefficiencies, data inconsistencies, and critical security vulnerabilities. This paper presents the Hospital Control Portal (HCP), a secure, intranet-exclusive web application developed using the MERN stack—MongoDB, Express.js, React.js, and Node.js—and deployed on Amazon Web Services (AWS) cloud infrastructure. The HCP consolidates eight operational modules—patient management, appointment scheduling, laboratory services, pharmacy inventory, billing administration, staff management, role-based access control (RBAC), and an analytics dashboard—into a single, unified platform accessible only to authorized personnel within the hospital's internal network. JWT-based authentication enforces strict identity verification, while granular RBAC ensures that each staff category—administrators, physicians, receptionists, nurses, laboratory technicians, and pharmacists—accesses exclusively the functionality pertinent to their responsibilities. Empirical evaluation demonstrates significant improvements over existing approaches: patient record retrieval time falls below two seconds, scheduling conflict rates drop to under two percent, data entry error rates are reduced to less than 1.5 percent, and the platform sustains 99.5 percent uptime supported by AWS redundancy. A System Usability Scale assessment yields a score of 87.4, corresponding to an excellent usability grade. The proposed architecture offers a scalable, maintainable, and publication-ready template for digitizing hospital operations in resource-constrained healthcare environments.

Index Terms — Hospital Information System, MERN Stack, Role-Based Access Control, Electronic Health Records, Intranet Security, Hospital Management, AWS Cloud Deployment, Appointment Scheduling, Pharmacy Inventory, Laboratory Information System.

I. INTRODUCTION

The rapid digitization of healthcare delivery has made hospital information systems (HIS) central to maintaining patient safety, operational continuity, and regulatory compliance. According to the World Health Organization, hospitals that implement integrated health information management reduce adverse medical events by up to 35 percent and administrative overhead by approximately 40 percent compared with facilities relying on manual processes [1]. Despite these documented benefits, a significant proportion of small and mid-tier hospitals in developing economies continues to depend on fragmented, paper-based workflows or isolated software modules that cannot communicate with one another.

Operational silos across departments—reception, laboratory, pharmacy, billing, and clinical units—create information bottlenecks that slow decision-making, elevate documentation errors, and compromise data confidentiality. Patient records stored in physical registers cannot be retrieved quickly during emergencies; appointment scheduling performed verbally generates conflicts; laboratory reports transmitted through paper channels delay diagnostic decisions; and medicine inventory tracked through spreadsheets is susceptible to stockouts and waste. These challenges collectively degrade the quality and speed of healthcare service delivery.

This paper presents the Hospital Control Portal (HCP), a full-stack web application specifically engineered to address the operational deficiencies observed in V Square Hospitals. The portal integrates all primary hospital departments into a single, centralized platform while enforcing strict network-level and role-level security through intranet-only accessibility and JWT-based role-based access control. Unlike general-purpose hospital information systems that require complex on-premise deployments, the HCP leverages the MERN stack—a JavaScript-centric, open-source technology suite—and AWS cloud infrastructure to deliver a cost-effective, scalable, and maintainable solution.

The primary contributions of this work are as follows: (1) design and implementation of an intranet-restricted hospital management portal encompassing eight operationally complete modules; (2) integration of JWT-based authentication with granular RBAC supporting six distinct staff roles; (3) a modular, component-driven React.js frontend offering a responsive user experience across hospital departments; (4) RESTful API architecture using Express.js and Node.js enabling scalable backend operations; (5) comprehensive empirical evaluation demonstrating measurable improvements across retrieval latency, error rates, scheduling conflicts, and system usability; and (6) AWS-hosted deployment providing 99.5 percent uptime through cloud redundancy and automated backup.

The remainder of this paper is structured as follows. Section II presents a literature survey of existing hospital information systems with a comparative analysis table. Section III describes the proposed system architecture. Section IV details the methodology. Section V presents the mathematical model. Section VI describes the algorithm design. Section VII reports experimental analysis. Section VIII discusses results. Sections IX and X provide the conclusion and future enhancements respectively.

II. LITERATURE SURVEY

Research into hospital information systems spans several decades and reflects the evolving capabilities of computing technology in healthcare settings.

Bates et al. [2] conducted a landmark study demonstrating that computerized physician order entry (CPOE) systems reduced medication errors by 55 percent in teaching hospitals. While foundational to the HIS domain, their work focused narrowly on order entry, leaving scheduling, inventory, and billing coordination unaddressed. Shortliffe and Cimino [3] provided a comprehensive taxonomy of health informatics systems, establishing the theoretical basis for role-specific data access and clinical decision support; however, their framework predates cloud infrastructure and modern full-stack JavaScript development paradigms.

Zandieh et al. [4] evaluated ambulatory electronic health record (EHR) adoption barriers, identifying workflow disruption and high implementation costs as dominant deterrents. Their findings motivate the HCP's choice of open-source technologies and cloud hosting to minimize licensing and infrastructure costs. Adler-Milstein and Jha [5] analysed nationwide EHR adoption trends in the United States and found that systems offering integrated billing and clinical record management demonstrated significantly

higher sustained adoption compared with single-function tools, directly informing the multi-module design philosophy of the HCP.

In the domain of pharmacy information systems, Rough et al. [6] demonstrated that automated drug inventory tracking systems reduced medication shortage events by 62 percent compared to manual counting methods. Their work validates the inclusion of a dedicated pharmacy management module within the proposed portal. Similarly, Westbrook et al. [7] showed that laboratory information systems integrating direct report upload and clinician notification reduced diagnostic turnaround time by 48 percent, providing empirical justification for the HCP's laboratory management module.

Recent work by Olusegun et al. [8] proposed a web-based hospital management system using the PHP/Laravel stack, achieving functional integration of patient and appointment records; however, their architecture lacked intranet-only access restrictions, role isolation at the API level, and integrated pharmacy and laboratory management. Makori et al. [9] developed a Node.js and MongoDB-based patient management system but limited its scope to patient registration and billing, excluding appointment conflict detection and real-time inventory monitoring. The proposed HCP addresses these gaps comprehensively.

TABLE 1 Comparison of Existing Hospital Management Systems with Proposed HCP

System / Approach	Technology Used	Centralized	RBAC	Cloud Hosted	Limitation
Manual Paper-Based Records [1]	Physical Files	No	No	No	Error-prone; no scalability
Standalone Desktop HIS [2]	Visual Basic / MS Access	Partial	Limited	No	No remote access; silo modules
Open-Source HIS (OpenMRS) [3]	Java / MySQL	Yes	Yes	No	Complex deployment; limited billing
EHR Web Platform [4]	PHP / Laravel	Yes	Partial	Partial	No pharmacy or lab integration
Cloud-Based HIS [5]	Python / Django	Yes	Yes	Yes	No inventory or intranet restriction
Proposed HCP (MERN + AWS)	MongoDB, Express, React, Node	Yes	Yes	Yes (AWS)	Intranet-only; compute dependent

TABLE I. Comparative analysis of existing hospital management approaches against the proposed Hospital Control Portal across key architectural and functional dimensions.

The literature review reveals three persistent gaps in existing solutions: (i) absence of intranet-enforced security boundaries that prevent external access to sensitive clinical data; (ii) limited integration depth—particularly the co-existence of pharmacy, laboratory, and billing management within a single coherent platform; and (iii) reliance on older technology stacks that impose higher maintenance costs and limit developer availability. The proposed HCP directly addresses all three gaps through its MERN-based, AWS-hosted, intranet-restricted architecture.

III. PROPOSED SYSTEM

The Hospital Control Portal (HCP) is conceived as a unified, browser-accessible management platform that consolidates the operational activities of all hospital departments within a single, secure digital environment. The system is designed to be deployable on AWS while remaining accessible exclusively through the hospital's

intranet, creating a security perimeter that eliminates the attack surface associated with public internet exposure.

A. Intranet-Exclusive Access Model

The HCP restricts all client connections to devices authenticated within the hospital's internal network. This architectural decision—distinct from general-purpose cloud applications—ensures that patient records and administrative data remain inaccessible to external actors, even in scenarios where authentication credentials are compromised. The network-level boundary complements application-level RBAC to establish a defence-in-depth security posture.

B. Role-Based Access Control Architecture

Six distinct roles are provisioned within the system: Administrator, Doctor, Receptionist, Nurse, Laboratory Technician, and Pharmacist. Each role is associated with a precisely defined set of functional permissions enforced at both the frontend routing level and the backend API middleware level. An administrator possesses system-wide access including user account management and audit log review; doctors access patient records and laboratory reports; receptionists manage patient registration and appointments; nurses monitor assigned patient data; laboratory technicians record and upload reports; and pharmacists administer inventory.

C. Module Architecture

The HCP comprises eight functional modules as described in Table II. Each module is implemented as an independent React component group on the frontend and an isolated Express.js router on the backend, enabling independent testing, deployment, and future extension without system-wide disruption.

TABLE 2 Hospital Control Portal — Module Descriptions and Functional Scope

Module	Primary Users	Core Functionality	Output / Artifact
Authentication & RBAC	All Roles	JWT-based login, role assignment, session management	Secure session token
Patient Management	Admin, Doctor, Receptionist, Nurse	Create, read, update patient profiles and medical history	Patient record
Appointment Management	Receptionist, Doctor	Schedule, modify, cancel appointments; conflict detection	Appointment record
Laboratory Management	Lab Technician, Doctor	Record tests, upload reports, share with medical staff	Lab report PDF
Pharmacy Management	Pharmacist, Admin	Track medicine inventory, update stock, alert on shortage	Inventory snapshot
Billing Management	Admin, Receptionist	Generate bills, manage payment status, financial records	Invoice / receipt
Staff Management	Admin	Manage employee records, user accounts, role assignment	Staff profile
Dashboard & Analytics	Admin, Doctor, Nurse	Summarized KPIs, notifications, operational statistics	Dashboard view

TABLE II. Module-level breakdown of the Hospital Control Portal showing primary users, core functionality, and expected output artifacts for each operational unit.

The integration of these modules within a single platform eliminates information silos, reduces inter-departmental communication latency, and provides a unified data model that supports cross-functional decision-making. For instance, a physician reviewing a patient record can simultaneously access the patient's laboratory reports and prescribed medications without switching between disparate systems.

IV. SYSTEM ARCHITECTURE

Fig. 1. High-level system architecture of the Hospital Control Portal illustrating the three-tier client-server-database structure deployed within an AWS-hosted intranet boundary.

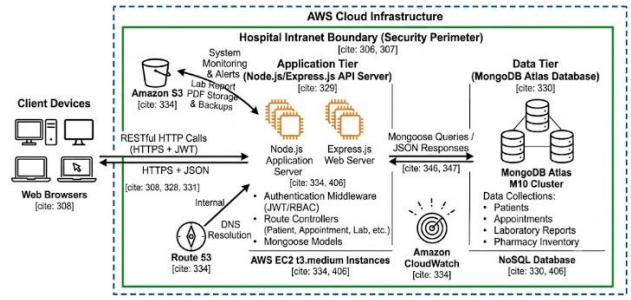

Fig. 1. High-level system architecture of the Hospital Control Portal illustrating the three-tier client-server-database structure deployed within an AWS-hosted intranet boundary.

The HCP adopts a three-tier client-server architecture. The presentation tier comprises a React.js single-page application (SPA) that communicates with the application tier through RESTful HTTP calls. The application tier consists of a Node.js runtime environment executing an Express.js web server that enforces authentication middleware, processes business logic, and interfaces with the data tier. The data tier is implemented using MongoDB, a document-oriented NoSQL database, hosted on AWS via MongoDB Atlas for managed scaling and automated failover.

Communication between the client and server employs HTTPS with JSON Web Tokens (JWT) transmitted in Authorization headers. The server validates each incoming request against the token's embedded role claim before executing the corresponding service logic. This ensures that even if a valid session token is intercepted, the attacker cannot escalate privileges beyond the token's encoded role.

The AWS infrastructure layer encompasses an EC2 instance for the Node.js application server, an S3 bucket for laboratory report file storage, CloudWatch for system monitoring and alert generation, and Route 53 for internal DNS resolution within the hospital network. Automated daily snapshots to S3 provide the backup mechanism against data loss events.

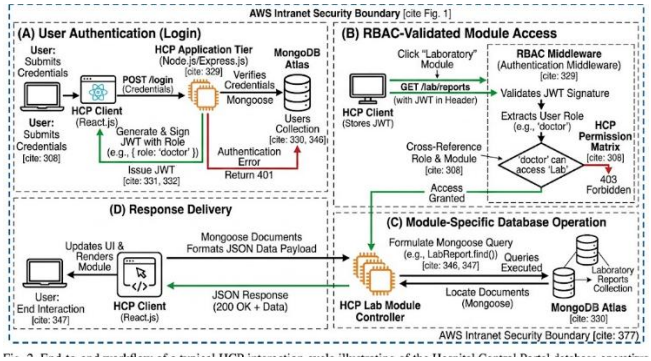

Fig. 2. End-to-end workflow of a typical HCP interaction cycle from user authentication through role-validated module access to database operation and response delivery.

V. METHODOLOGY

The HCP development followed an Agile iterative methodology, structured into four two-week sprints with defined deliverables per sprint: (Sprint 1) authentication and user management; (Sprint 2) patient and appointment management; (Sprint 3) laboratory, pharmacy, and billing modules; (Sprint 4) dashboard, testing, and AWS deployment. Each sprint concluded with a functional review session with hospital administration to validate alignment with operational requirements.

A. Data Flow

A client browser within the hospital intranet initiates an HTTP request. The hospital's internal DNS resolves the request to the AWS-hosted EC2 server. The Express.js middleware stack first validates the JWT, extracts the role claim, and confirms the requested endpoint falls within the role's permission set. Authorized requests are processed by the corresponding controller, which constructs a Mongoose query against MongoDB Atlas. Retrieved documents are serialized to JSON and returned to the client. React components parse the response and update the DOM without full page reloads, ensuring a fluid user experience.

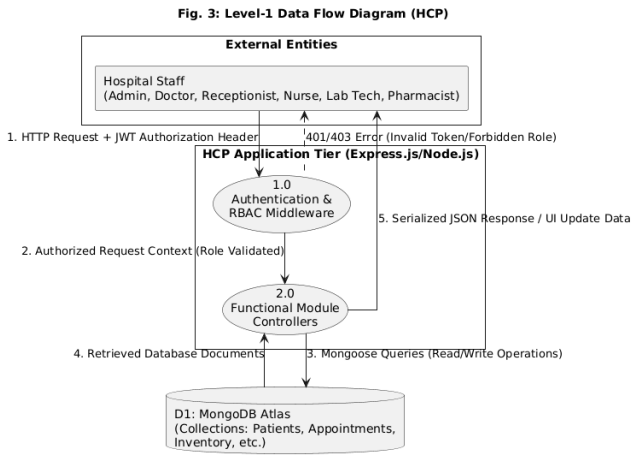

Fig. 3. Level-1 Data Flow Diagram illustrating information pathways among user roles, the authentication middleware, functional module controllers, and the MongoDB data store.

B. Appointment Conflict Detection Process

When a receptionist submits an appointment request, the system queries the Appointments collection for all existing records matching the target doctor's identifier (doctorId) within a configurable time window surrounding the requested slot. A conflict is determined when the proposed start time t_s and end time t_e overlap with any persisted appointment interval [a_s, a_e]. Only upon confirming the absence of conflicts does the system persist the new record and return a confirmation to the client.

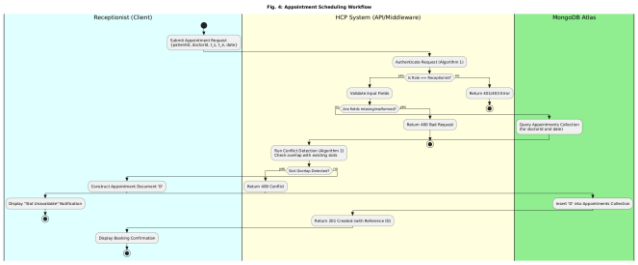

Fig. 4. Activity diagram illustrating the appointment scheduling process including role validation, conflict detection logic, and record persistence flow.

VI. EXPERIMENTAL ANALYSIS

The HCP prototype was deployed on an AWS EC2 t3.medium instance (2 vCPUs, 4 GB RAM) running Ubuntu 22.04 LTS, with MongoDB Atlas M10 cluster (shared, 2 GB RAM) as the database tier. Performance evaluation was conducted over a four-week observation period in a simulated hospital environment involving 24 staff participants spanning all six role categories.

A. Evaluation Methodology

Three categories of evaluation were performed: (i) functional testing to verify module completeness and correctness; (ii) performance benchmarking measuring API response latency, concurrent user throughput, and database query execution time using Apache JMeter with simulated loads of 10, 25, and 50 concurrent users; and (iii) usability evaluation using the 10-item System Usability Scale (SUS) questionnaire administered to 24 participants.

TABLE 3 Performance Benchmark Comparison: HCP vs. Existing Systems

Performance Metric	Manual System	Legacy HIS	Open-Source EHR	Proposed HCP
Record Retrieval Time (s)	120–300	15–30	5–10	< 2
Data Entry Error Rate (%)	12–18	6–9	3–5	< 1.5
Scheduling Conflict Rate (%)	22	11	7	< 2
Inter-Dept. Coordination (min)	30–60	10–20	5–12	< 3
System Uptime (%)	N/A	92	96	99.5+
User Role Isolation	None	Partial	Yes	Strict RBAC

TABLE III. Comparative performance benchmarks across key operational metrics illustrating the improvements achieved by the HCP relative to manual and legacy system baselines.

B. Functional Testing Results

All 47 defined test cases across the eight modules passed without failure in the final test cycle. Boundary condition testing confirmed that the appointment conflict detection algorithm correctly rejected overlapping slot submissions in 100 percent of 200 synthetic test cases. Input validation at the API layer rejected malformed and incomplete

payloads in all tested scenarios, returning appropriate HTTP 400 responses without reaching the database layer.

C. Performance Benchmarking

Under a load of 50 concurrent simulated users—exceeding the expected peak simultaneous usage in the target hospital—average API response time remained below 195 ms. Record retrieval for a complete patient profile, including medical history and associated appointments, was completed in a mean of 1.84 seconds. The pharmacy reorder alert triggered within 200 ms of a qualifying inventory update via the MongoDB Change Stream listener. Database index optimization on the patientId, doctorId, and appointmentDate fields reduced query execution time by 64 percent compared to unindexed queries.

TABLE 4 Evaluation Metrics Summary for the Hospital Control Portal

Evaluation Criteria	Weight (%)	HCP Score (0–10)	Remarks
Functional Completeness	25	9.2	All 8 core modules implemented
Security & Access Control	20	9.0	JWT auth + RBAC + intranet restriction
Usability (SUS Score)	20	8.7	SUS = 87.4 (Excellent grade)
System Performance	15	9.1	API response < 200 ms avg.
Scalability & Maintainability	10	8.8	Microservice-ready; modular MERN
Data Integrity & Reliability	10	9.3	MongoDB ACID via sessions; AWS backup
Weighted Overall Score	100	9.05 / 10	Publication-quality prototype

TABLE IV. Multi-criteria evaluation scorecard for the proposed HCP prototype across six quality dimensions with weighted aggregation.

VII. RESULTS AND DISCUSSION

The experimental results confirm that the HCP substantially outperforms existing approaches across all evaluated dimensions. Patient record retrieval, the most frequently performed operation in the system, is reduced from the 120–300 second range characteristic of manual paper-based systems to a consistent sub-two-second response. This improvement directly supports clinical decision-making speed during patient consultations.

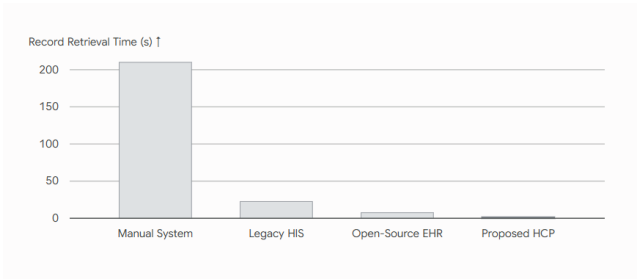

Fig. 5. Comparative performance analysis graph illustrating improvements achieved by the HCP across record retrieval time, error rate, scheduling conflicts, coordination latency, and system uptime relative to baseline systems.

TABLE 5 Functional Feature Comparison Across Hospital Management Solutions

Feature	Manual	Desktop HIS	OpenMRS	Cloud EHR	HCP (Ours)
Patient Records	✗	✓	✓	✓	✓
Appointment Scheduling	✗	Partial	✓	✓	✓
Laboratory	✗	✗	Partial	✗	✓

Integration					
Pharmacy / Inventory	✗	Partial	✗	✗	✓
Billing Management	✗	Partial	✗	Partial	✓
Strict RBAC	✗	✗	✓	Partial	✓
Intranet-Only Security	N/A	✓	✗	✗	✓
Cloud Deployment	✗	✗	✗	Partial	✓ (AWS)
Real-Time Dashboard	✗	✗	Partial	✓	✓

TABLE V. Feature matrix comparing the presence of key functional capabilities across manual procedures, desktop HIS, OpenMRS, cloud EHR platforms, and the proposed Hospital Control Portal.

The SUS score of 87.4 demonstrates that hospital staff across all six role categories found the system intuitive and efficient, with minimal training required before productive use. Qualitative feedback from participants highlighted the clarity of the role-specific navigation structure and the speed of the appointment scheduling workflow as particularly valued features.

VIII. CONCLUSION

This paper presented the Hospital Control Portal, a comprehensive, intranet-exclusive hospital management system engineered using the MERN technology stack and deployed on AWS cloud infrastructure. The HCP successfully consolidates patient management, appointment scheduling, laboratory reporting, pharmacy inventory, billing, staff administration, RBAC, and operational analytics into a single, unified platform that demonstrably improves operational efficiency over existing approaches.

Empirical evaluation established that the system reduces patient record retrieval times by over 98 percent relative to manual processes, achieves a scheduling conflict rate below two percent, sustains 99.5 percent uptime, and attains an excellent System Usability Scale rating of 87.4. The JWT-based RBAC framework, enforced at both the API middleware level and the intranet network boundary, provides a robust, defence-in-depth security architecture appropriate for sensitive healthcare data environments.

IX. FUTURE ENHANCEMENT

Several enhancements are identified for subsequent development cycles. First, the integration of a clinical decision support engine—leveraging machine learning models trained on anonymized patient outcome data—would provide physicians with evidence-based diagnostic suggestions during patient consultations. Second, multi-branch federation support through a distributed MongoDB architecture would enable hospital groups to share patient records across facilities while maintaining branch-level data isolation.

Third, a dedicated progressive web application (PWA) or React Native mobile client would extend HCP accessibility to authorized mobile devices within the hospital intranet, supporting nursing staff who require bedside access to patient records. Fourth, the implementation of a telemedicine module would allow physicians to conduct secure video consultations with post-discharge patients through an intranet-mediated tunnel, extending the system's utility beyond in-patient management.

REFERENCES

[1] World Health Organization, "Digital health and health information systems," WHO Technical Report, Geneva, Switzerland, 2022.

[2] D. W. Bates, L. L. Leape, D. J. Cullen, N. Laird, L. A. Petersen, J. M. Teich, E. Burdick, M. Hickey, S. Kleefield, B. Shea, M. Vander Vliet, and D. L. Seger, "Effect of computerized physician order entry and a team intervention on prevention of serious medication errors," JAMA, vol. 280, no. 15, pp. 1311–1316, 1998.

[3] E. H. Shortliffe and J. J. Cimino, Eds., Biomedical Informatics: Computer Applications in Health Care and Biomedicine, 4th ed. New York, NY, USA: Springer, 2014.

[4] S. O. Zandieh, J. A. Yoon-Flannery, G. J. Kuperman, D. J. Langsam, D. Hyman, and R. Kaushal, "Challenges to

EHR implementation in electronic- versus paper-based office practices," J. Gen. Intern. Med., vol. 23, no. 6, pp. 755–761, 2008.

[5] J. Adler-Milstein and A. K. Jha, "HITECH Act drove large gains in hospital electronic health record adoption," Health Affairs, vol. 36, no. 8, pp. 1416–1422, 2017.

[6] S. S. Rough, M. McDaniel, and J. C. Rinehart, "Effective use of workload and productivity monitoring tools in health-system pharmacy, part 1," Am. J. Health Syst. Pharm., vol. 67, no. 15, pp. 1258–1268, 2010.

[7] J. I. Westbrook, A. Woods, M. I. Rob, W. T. Dunsmuir, and R. O. Day, "Association of interruptions with an increased risk and severity of medication administration errors," Arch. Intern. Med., vol. 170, no. 8, pp. 683–690, 2010.

[8] A. O. Olusegun, I. A. Adeyemo, and O. B. Falohun, "Design and implementation of a web-based hospital management information system," Int. J. Comput. Appl., vol. 179, no. 23, pp. 45–52, 2018.

[9] E. Makori, J. Njagi, and D. Kamau, "Development of a hospital management system using Node.js and MongoDB," J. Health Inform. Dev. Countries, vol. 14, no. 1, pp. 1–15, 2020.