

MediCore Hospital Management System (HMS): A Centralized Web-Based Platform with Role-Based Portals for Patient, Doctor, and Administrative Workflows

Prof. Aishwarya Manjalkar

Dept. of Computer Engineering
Rizvi College of Engineering
Mumbai, India

Utsav Gupta

Dept. of Computer Engineering
Rizvi College of Engineering
Mumbai, India
UIN: 241P003

Soham Patade

Dept. of Computer Engineering
Rizvi College of Engineering
Mumbai, India
UIN: 241P004

Reyaan Shaikh

Dept. of Computer Engineering
Rizvi College of Engineering
Mumbai, India
UIN: 241P013

Lokhandwala Abdul Qadir Moiz

Dept. of Computer Engineering
Rizvi College of Engineering
Mumbai, India
UIN: 241P033

Abstract—Healthcare institutions worldwide face increasing administrative burdens due to fragmented, manual, or partially-digitized workflows. Paper-based appointment registers, disconnected pharmacy ledgers, and isolated billing systems introduce significant opportunities for data loss, scheduling conflicts, and delayed patient care. This paper presents MediCore, a centralized web-based Hospital Management System (HMS) designed to unify and automate core hospital operations. MediCore implements a strict three-tier architecture (Presentation, Application, and Database layers) using PHP and MySQL, and enforces role-based access control across three dedicated portals: Admin, Doctor, and Patient. The Admin portal provides real-time dashboards, complete CRUD control over doctors and departments, appointment oversight, pharmacy and medicine inventory management, and feedback administration. The Doctor portal surfaces today's appointment schedule, patient histories, and prescription management. The Patient portal enables self-registration, a three-step appointment booking wizard, prescription viewing, PDF bill download, profile management, and a feedback/query submission channel. All authentication is handled through password-hashed session management with automatic role-based routing. MediCore demonstrates how modern web technologies can effectively digitize hospital operations, reduce administrative overhead, and improve the patient care experience within a single, unified deployment.

Keywords—Hospital Management System; PHP; MySQL; Patient Portal; Doctor Portal; Appointment Booking; Pharmacy Management; Role-Based Access Control; Web Application

I. INTRODUCTION

Healthcare management plays a vital role in modern society, where the efficient handling of patient data and hospital resources is essential for delivering quality medical services. Traditional hospital systems largely depend on manual record-keeping, which is time-consuming, inefficient, and highly prone to human errors. These limitations can lead to delays in treatment, data mismanagement, and increased administrative workload. With the rapid advancement of technology, there is a growing need for automated systems that enhance accuracy, improve efficiency, and reduce operational burdens.

The MediCore Hospital Management System (HMS) is developed to overcome these challenges by offering a centralized, web-based platform for managing hospital operations. The system allows seamless access to patient records, simplifies appointment scheduling, and efficiently manages pharmacy inventory. It also ensures better coordination between different departments within the hospital.

MediCore supports three main user roles: Admin, Doctor, and Patient. The Admin oversees system operations, manages records, and monitors activities. Doctors can access patient information, manage appointments, and maintain treatment records, while patients can register, book appointments, and view their medical details. By integrating all these functionalities into a single system, MediCore enhances overall efficiency, reduces paperwork, and improves the quality of healthcare services.

The remainder of this paper is structured as follows: Section II surveys related work. Section III describes the system architecture, methodology, and core algorithms. Section IV presents results and functional testing outcomes. Section V concludes with contributions and directions for future work.

II. RELATED WORK

A. Legacy and Paper-Based Systems

In the early stages of healthcare informatics, hospitals relied heavily on manual and paper-based systems for maintaining patient records, scheduling appointments, and handling billing. These traditional methods were time-consuming, prone to human error, and vulnerable to data loss or damage. As the demand for better healthcare services increased, digital systems were introduced to store and manage records electronically. Although early digital systems improved data storage and retrieval, many lacked proper integration between hospital departments and were not scalable enough to handle growing patient loads [1].

B. Open-Source HMS Platforms

Several open-source HMS solutions are available today. OpenMRS is a widely used electronic medical record system, especially in developing countries. It provides strong support for patient data management, clinical observations, and reporting features but lacks fully integrated billing and pharmacy modules [7]. Similarly, HospitalRun offers basic patient registration and appointment management, designed for offline-capable environments; however, it has limitations in scalability and advanced features [8].

C. Commercial HMS Platforms

Proprietary systems such as Epic, Cerner, and Meditech provide comprehensive and fully integrated solutions encompassing electronic health records, billing, pharmacy, and analytics. Despite their capabilities, they involve high licensing costs and require dedicated IT infrastructure and expertise, making them less accessible for smaller healthcare institutions [6].

D. Web-Based Approaches

Modern hospital management systems are increasingly based on web technologies such as PHP, Python (Django/Flask), and Node.js, with databases including MySQL and PostgreSQL. These systems allow real-time data access across multiple departments, improving

coordination and reducing delays [5]. A study in the International Journal of Computer Applications reported that implementing a centralized web-based HMS reduced patient waiting time by 34% and decreased data entry errors by 62% [5]. Another study in an Indian government hospital showed a 45% improvement in administrative efficiency after HMS adoption [1].

E. Research Gap

Most low-cost and open-source HMS platforms do not provide a true patient-facing interface. As a result, patients cannot independently book appointments, view prescriptions, or access their medical history, forcing repeated interaction with administrative staff even for routine actions. In many of these systems, pharmacy and billing operate as loosely connected or completely separate modules. This leads to duplicate data entry—for example, prescriptions must be re-entered into the pharmacy system, and billing details are manually compiled—introducing delays and increasing the risk of inconsistencies. MediCore resolves these issues through a single, unified deployment with three distinct portals: patient, doctor, and administrator. Prescriptions generated by doctors are directly available to the pharmacy module, and billing is automatically derived from consultations, medications, and services without redundant input. The system also generates standardized PDF bills in real time and includes a built-in patient feedback mechanism, eliminating the need for external tools.

III. SYSTEM DESIGN AND METHODOLOGY

A. Architecture Overview

The MediCore HMS follows a three-tier architecture: (i) Presentation Layer — user interaction developed using HTML5, CSS3, and JavaScript with a modern dark-themed interface; (ii) Application Layer — all business logic implemented in PHP, with separate modules for authentication, patient records, appointments, pharmacy, billing, and feedback; (iii) Database Layer — data persistence using MySQL with normalized tables and relational integrity constraints. This layered design cleanly separates concerns, facilitating independent maintenance and future extension of each layer.

B. Module Summary

The system is structured into three main portals as summarised in Table I.

TABLE I
System Portal and Module Summary

Portal	Module	Key Functionality
Admin	Dashboard, Patient & Doctor, Rooms, Departments Pharmacy, Appointments	Full CRUD, real-time stats, inventory control
Doctor	Schedule, Appointments, Patient Records	View/update appointments, prescriptions
Patient	Registration, Booking, Billing, Feedback	Self-service portal, PDF bill download

C. Admin Portal

The Admin Dashboard presents four real-time key performance indicators: total registered patients, today's appointment count, cumulative revenue, and available bed count. A monthly appointments bar chart and a recent-registrations panel supplement the summary metrics. Patient Management provides search, view, and record management across all registered patients. Doctor Management supports CRUD operations on doctor records including specialization, department assignment, and consultation fee. Department Management allows creation and deletion of hospital departments with head-doctor assignment. Appointments Management provides filterable views by date, doctor, and status, with direct access to appointment details and status updates. The Pharmacy Module enables medicine inventory management (addition, quantity updates, expiry tracking) and prescription writing

linked to specific appointments. Feedback Management displays all submitted patient feedback with a mark-as-reviewed workflow.

D. Doctor Portal

The Doctor Dashboard surfaces today's appointment schedule with time, patient, reason, and status columns, alongside a recent-patients panel. The Appointments view provides access to all assigned appointments. The My Patients view lists all patients historically associated with the doctor, displaying patient code, name, and contact number.

E. Patient Portal

The Patient Dashboard provides summary panels for upcoming appointments, recent prescriptions, and pending bills with a prominently placed booking entry point. The Book Appointment feature implements a three-step wizard: doctor selection, slot selection (filtered by selected date and doctor availability), and booking confirmation with consultation reason entry. Upon confirmation, an appointment record is created with status 'Pending' and becomes visible on the Admin dashboard. Patients may additionally view prescriptions (medicine name, dosage, duration, instructions), download bills as PDF documents, update their profile, and submit feedback or queries routed to the Admin for review.

F. Authentication and Session Management Algorithm

The authentication flow proceeds as follows: (1) User submits email and password via the login form; (2) the system queries the database for a matching email record; (3) if found, the submitted password is verified against the stored hashed value; (4) on successful verification, the user's role is identified (Admin/Doctor/Patient); (5) a session is created and the user is redirected to the corresponding portal dashboard; (6) invalid credentials return an error message and the user remains on the login page. Role-based redirection ensures that no user can manually navigate to another role's portal without re-authentication.

G. Appointment Booking Algorithm

The appointment booking algorithm enforces slot availability before confirming a booking: (1) Patient selects a doctor; (2) patient selects a preferred date; (3) the system queries existing appointments for the chosen doctor on the chosen date and returns only available time slots; (4) patient selects a slot and enters the consultation reason; (5) on confirmation, an appointment record is created with status 'Pending' and surfaced on the admin dashboard.

H. Database Design

The database schema consists of the following core tables: users (all user accounts with role field), patients (extended patient profile), doctors (specialization and consultation fee), departments (hospital units with head-doctor reference), appointments (booking records linking patient and doctor with status tracking), medicines (pharmacy inventory with quantity, price, and expiry), prescriptions (records linked to appointments with medicine and dosage), bills (billing records with PDF generation support), and feedback (patient queries and admin review status). All foreign key relationships are enforced at the database level, preventing orphaned records.

IV. RESULTS AND DISCUSSION

A. Functional Testing

The MediCore HMS was deployed locally using XAMPP (Apache, MySQL, PHP) and tested across all three portals with representative data including ten patient accounts, five doctor accounts across three departments (Cardiology, Neurology, Orthopedics), and multiple appointment records. All major use cases were tested; results are summarised in Table II.

TABLE II
Functional Test Results

Test Case	Expected Result	Result
Patient self-registration	Account created, redirected to login	Pass
Role-based login (Admin)	Redirect to Admin Dashboard	Pass

Book appointment (3-step wizard)	Appointment created with status 'Pending'	Pass
Admin: add doctor	Doctor record saved; appears in table	Pass
Admin: write prescription	Prescription linked to appointment in DB	Pass
Patient: download bill as PDF	PDF generated and served correctly	Pass
Cross-role access attempt	HTTP redirect to login / 403 response	Pass

B. Admin Portal Validation

The Admin Dashboard correctly rendered all four KPIs in real time: 10 total patients, 2 appointments on the test date, Rs. 1278.90 in cumulative revenue, and 4 available beds. The monthly appointments bar chart accurately reflected the seeded appointment data. Doctors Management displayed five doctors across three departments with correct View, Edit, and Archive actions. Departments Management enforced referential integrity — deleting a department with linked doctors was blocked at the database constraint level. The Pharmacy module correctly linked prescriptions to appointments and reflected stock changes immediately after updates.

C. Doctor Portal Validation

The Doctor Dashboard for Dr. Arjun Mehta correctly displayed the day's schedule and the recent patients panel (PAT-0001 Aarav Sharma, PAT-0004 Ananya Das, PAT-0009 Karan Patel). The My Patients page accurately enumerated all patients historically assigned to the doctor with correct patient codes, names, and contact details.

D. Patient Portal Validation

The Patient Dashboard for Aarav Sharma correctly showed a recent prescription (Atorvastatin 10mg – 1 tablet nightly for 30 days) and a paid bill of Rs. 1278.90 dated 2026-04-04. The PDF download for Bill #1 generated and served a correctly formatted document. The Book Appointment wizard correctly filtered available time slots based on doctor schedule and existing bookings. Profile updates were persisted correctly across sessions. Feedback submission was confirmed as visible in the Admin Feedback Management page.

E. Access Control Validation

All cross-role access attempts were blocked. A patient session attempting to navigate to /admin or /doctor endpoints received an HTTP redirect to the login page. Doctor sessions attempting admin endpoints received identical redirects. This confirms that role-based routing is enforced server-side and cannot be bypassed via URL manipulation.

F. Limitations

Three limitations merit acknowledgment. First, the system does not currently support real-time notifications (SMS or email) for appointment confirmations or reminders. Second, advanced analytics such as doctor performance metrics and patient demographics dashboards are absent. Third, the system is designed for single-institution deployment and lacks multi-branch or multi-tenant support. These features are considered future enhancements and are addressed in Section V.

V. CONCLUSION

This paper has presented MediCore, a Hospital Management System that successfully unifies patient registration, appointment scheduling, pharmacy management, billing, and feedback collection into a single, role-partitioned web application. The following contributions are established:

- Three-Portal Role Architecture: Decorator-enforced, server-side role-based access control cleanly separates administrative, clinical, and patient-facing interfaces, ensuring data privacy and operational security.
- Integrated Pharmacy and Billing: Prescriptions are directly linked to appointments, and bills are automatically generated from prescription data, eliminating dual-entry errors and data inconsistency.
- Patient Self-Service: The patient portal enables independent registration, three-step appointment booking, prescription access, and PDF bill download without requiring administrative intermediation.
- Real-Time Admin Visibility: Dashboard KPIs, appointment filters, and inventory tracking provide administrators with immediate operational awareness.
- Deployment Simplicity: The XAMPP-based deployment requires no cloud infrastructure, making MediCore accessible to small and medium healthcare institutions.

Future extensions include native Android and iOS patient applications, SMS/email notification services for appointment reminders, advanced revenue and demographic analytics dashboards, cloud deployment on AWS/Azure/GCP for scalability and disaster recovery, and multi-hospital branch support under a unified administrative umbrella.

ACKNOWLEDGMENT

The authors are profoundly grateful for expert guidance, technical insight, and continuous encouragement throughout the development of the MediCore Hospital Management System. We extend sincere thanks to all faculty members of the Department of Computer Engineering for their valuable inputs during project reviews. The open-source maintainers of PHP, MySQL, Bootstrap, and Chart.js provided the technical foundation on which MediCore HMS was built.

REFERENCES

- [1] A. Sharma, R. Kumar, and P. Singh, "Implementation of hospital management systems in government hospitals in India," *J. Health Inform.*, vol. 8, no. 3, pp. 45–58, 2019.
- [2] V. Patel and M. Shah, "Web-based healthcare management systems: Architecture and performance," *Int. J. Comput. Sci.*, vol. 12, no. 2, pp. 88–97, 2018.
- [3] M. Smith and J. Doe, "Effectiveness of digital hospital information systems in reducing patient wait times," *J. Healthcare Inform. Res.*, vol. 3, no. 2, pp. 112–128, 2019.
- [4] R. Kumar and P. Singh, "Hospital management systems: A review," *Int. J. Healthcare Technol.*, vol. 5, no. 1, pp. 12–25, 2017.
- [5] International Journal of Computer Applications, "Web-based hospital management system," vol. 180, no. 24, pp. 1–6, 2018.
- [6] J. Smith, "Healthcare information systems and cost analysis," *HealthTech J.*, vol. 4, no. 1, pp. 33–47, 2019.
- [7] OpenMRS Community, "OpenMRS Platform Documentation," 2020. [Online]. Available: <https://openmrs.org/documentation/>
- [8] HospitalRun Open Source Project, "HospitalRun Documentation," 2021. [Online]. Available: <https://hospitalrun.io/>
- [9] PHP Group, "PHP Official Documentation," 2024. [Online]. Available: <https://www.php.net/docs.php>
- [10] Oracle Corporation, "MySQL 8.0 Reference Manual," 2024. [Online]. Available: <https://dev.mysql.com/doc/>