© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716392

Web-Based Hospital Management System: A Django-Powered Platform for Modern Healthcare Administration

M. HARI MURARI[1], R. ROHINI[2], R. VIKITHA[3], CH. LAVANYA[4]
[1,2,3]*UG Student, Dept. of CSE*
[4]*Asst. Professor, Dept. of CSE*

Abstract- ***Healthcare institutions worldwide are under growing pressure to modernise their administrative and clinical workflows. Paper-based record-keeping, phone-in appointment systems, and disconnected billing processes are no longer sustainable in an era where patients expect speed, transparency, and convenience. This project presents a web-based Hospital Management System (HMS) built on the Django framework — a platform designed to bring patients, doctors, and administrators together in one secure, easy-to-use digital environment. The system handles everything from patient registration and appointment booking to prescription management, payment processing, and downloadable PDF medical reports. Built with Python, Django, SQLite, and standard web technologies (HTML, CSS, JavaScript), the system is lightweight, scalable, and cost-effective — making it particularly well suited for small to mid-size healthcare facilities looking to modernise without a large IT budget.***

Index Terms- *Hospital Management System, Django, Patient Care, Appointment Scheduling, Healthcare Automation, Web Application*

I. INTRODUCTION

Hospitals are the backbone of public health — yet many still run on processes that would feel familiar to someone working in healthcare decades ago: patient records piled in folders, appointments scribbled in ledgers, billing calculated by hand. These aren't just inefficiencies; they're sources of error that can have real consequences for patient care.

The Hospital Management System described in this paper replaces those manual workflows with a clean, role-based digital platform. Every interaction — a patient booking an appointment, a doctor updating a prescription, or an admin reviewing payment records — flows through a single integrated web application.
Three distinct user roles sit at the heart of the system: patients, doctors, and administrators. Each has their own secure login, dashboard, and clearly defined set of capabilities. The result is a system that doesn't merely digitise existing paperwork — it fundamentally changes how a hospital operates.

II. BACKGROUND AND PURPOSE

Running a hospital is an extraordinarily complex operation. On any given day, staff must coordinate hundreds of appointments, maintain thousands of patient records, process payments, track prescriptions, and ensure that the right information reaches the right people at the right time. When that coordination relies on manual processes and disconnected systems, things fall through the cracks.
Recent advances in web technologies, cloud computing, and relational databases have made it possible to build centralised systems that handle all of these functions from a single platform — maintaining real-time records, automating routine tasks, and giving authorised users instant access to the information they need.

The primary goal of this project was to build a centralised, automated HMS that handles patient registration, appointment scheduling, doctor management, billing, and report generation in one place. Beyond operational efficiency, the system was also designed with transparency and accountability in mind — every action is tied to an authenticated user, creating a clear audit trail.

III. METHODOLOGY

The development of this system followed a structured, iterative approach. Requirements were gathered by analysing hospital workflows and interviewing administrative and medical staff — ensuring the final product reflects how hospitals

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716392

actually operate, not just how they are assumed to operate.

The system was designed around a three-tier architecture and developed using the Agile methodology, allowing continuous refinement through feedback. The database was designed using normalisation principles to minimise redundancy and maintain consistency. Testing covered unit, integration, and user acceptance stages.

IV. SYSTEM ARCHITECTURE

The HMS uses a three-tier Client–Server architecture. The presentation layer is the web interface rendered in the browser, built with HTML, CSS, and JavaScript. The application layer is Django, which handles all business logic, access rules, and data coordination. The data layer is an SQLite database storing everything from patient records to payment histories. Django Views act as the central coordinator — receiving requests, applying logic, and returning responses through HTML templates.

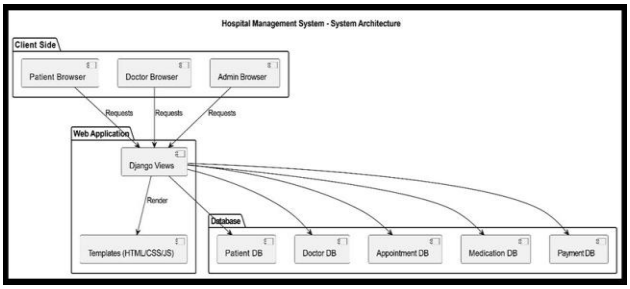

Fig. 1 — System Architecture Diagram

Tools and Technologies

The tech stack was chosen for simplicity, reliability, and ease of deployment. Python and Django handle all backend processing. HTML, CSS, and JavaScript provide the frontend. SQLite serves as the database — requiring no separate database server. The entire application runs on Windows 10 and can be started with a single command from the project directory.

V. SYSTEM MODULES

5.1 Patient Module

The Patient Module is the most visible face of the system — what the majority of users interact with daily. Patients can register, log in, and manage their entire healthcare journey through it. Key capabilities include:

- Registering with personal and contact information
- Booking, viewing, or cancelling appointments with specific doctors
- Uploading medical reports as P+DF files
- Viewing prescribed medications and test recommendations
- Making payments and downloading formatted PDF reports

Security is baked in throughout. Patients can only ever see their own records. Sessions are managed carefully to prevent unauthorised access, and all uploaded files are validated before being stored.

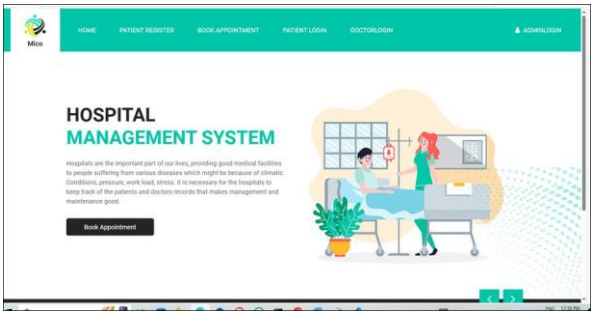

Fig. 2 — Hospital Management System Home Page

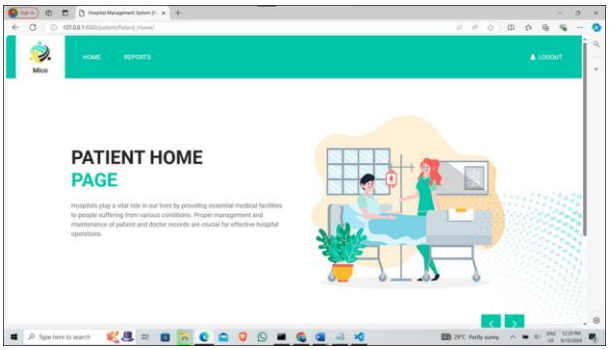

Fig. 3 — Patient Home Page

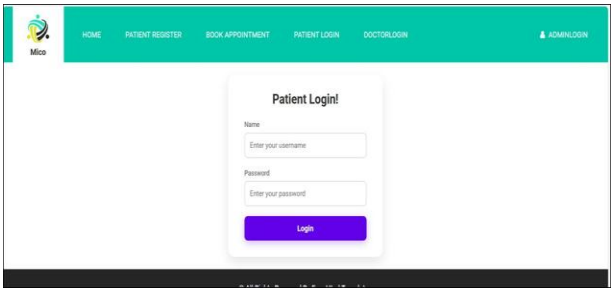

Fig. 4 — Patient Login Interface

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716392

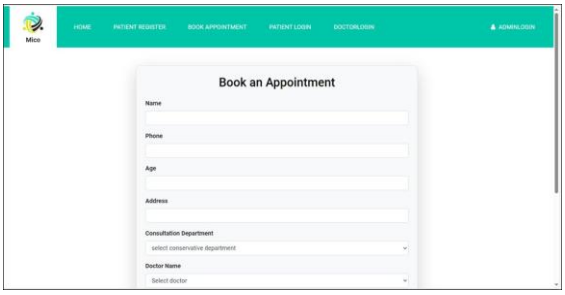

Fig. 5 — Book an Appointment Form

5.2 Doctor Module

Doctors interact with the system primarily through their patient lists and appointment schedules. Once logged in, a doctor can view every assigned patient, review their medical history and previous visits, and update records with new prescriptions or test recommendations. The module is designed to surface exactly the information a doctor needs for clinical decision-making, without cluttering the interface with administrative data that isn't relevant.

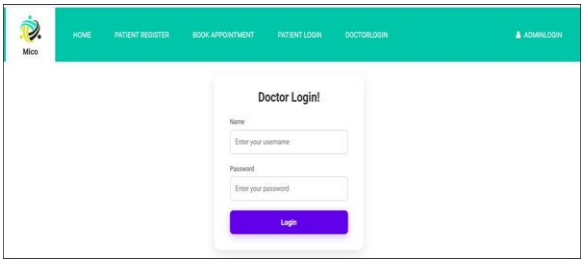

Fig. 6 — Doctor Login Interface

5.3 Admin Module

The Admin Module is the system's control centre. Administrators can add or remove doctor accounts, approve or reject patient appointments, monitor payment records, and view hospital-wide data through a centralised dashboard. They are also responsible for maintaining the overall integrity of the system — ensuring records are accurate and that users have precisely the access they need, and nothing more.

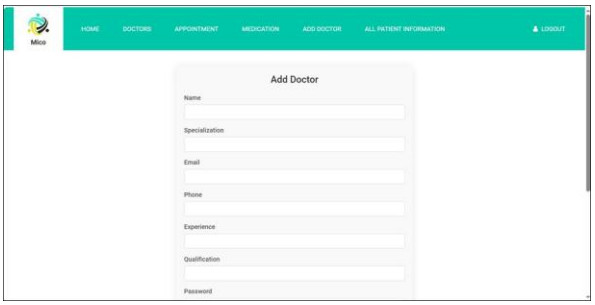

Fig. 7 — Admin: Add Doctor Form

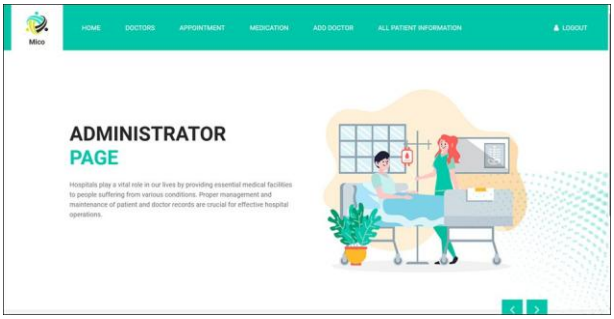

Fig. 8 — Administrator Home Page

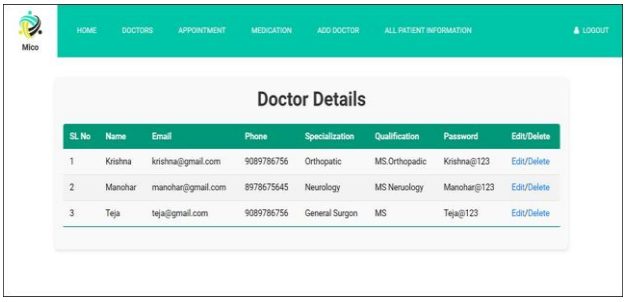

Fig. 9 — Doctor Details (Admin View)

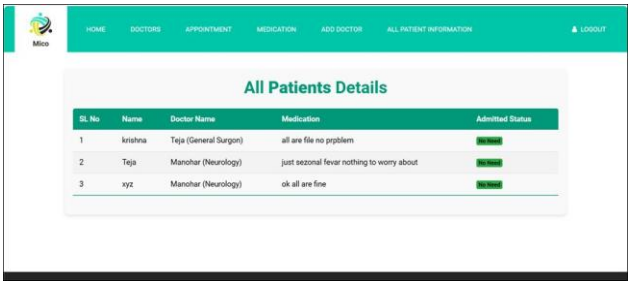

Fig. 10 — All Patients Details (Admin View)

VI. DATA FLOW AND COMMUNICATION

When a user performs an action — such as booking an appointment or updating a record — the request travels from the presentation layer to the application layer for processing. Processed data is then stored in or retrieved from the database. Communication between modules goes through secure, structured interfaces that validate all data to prevent errors and

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716392

duplication. Real-time updates ensure every user always sees the latest information.

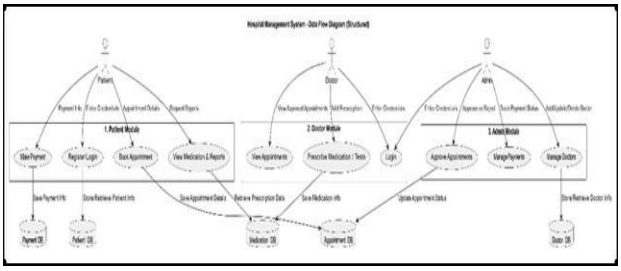

Fig. 11 — Data Flow Diagram (Structured)

VII. IMPLEMENTATION

The system is deployed in a local development environment, launched via the command line from the project directory. Once started, Django initialises all modules and begins serving the web application on a local port. All database operations go through Django's ORM, which translates Python code into optimised SQL queries — keeping the codebase clean and making the system easy to maintain.

Security and Privacy

Security was built into every layer of the system. User authentication ensures only registered users can access the platform. Role-based access control means patients, doctors, and administrators each see only what they are permitted to see. Passwords are hashed before storage, sessions expire automatically, and regular database backups protect against data loss. All sensitive data is stored with restricted access permissions.

VIII. SYSTEM DIAGRAMS

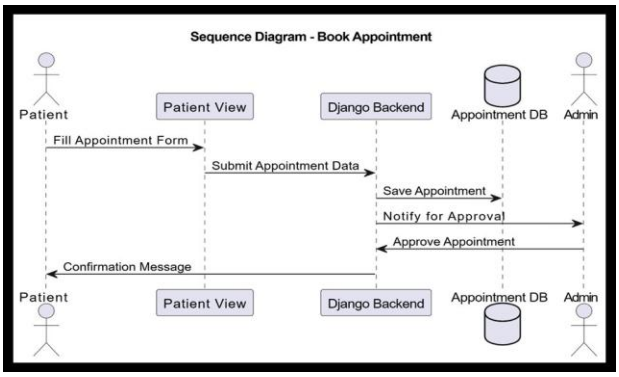

Fig. 12 — Sequence Diagram: Book Appointment Flow

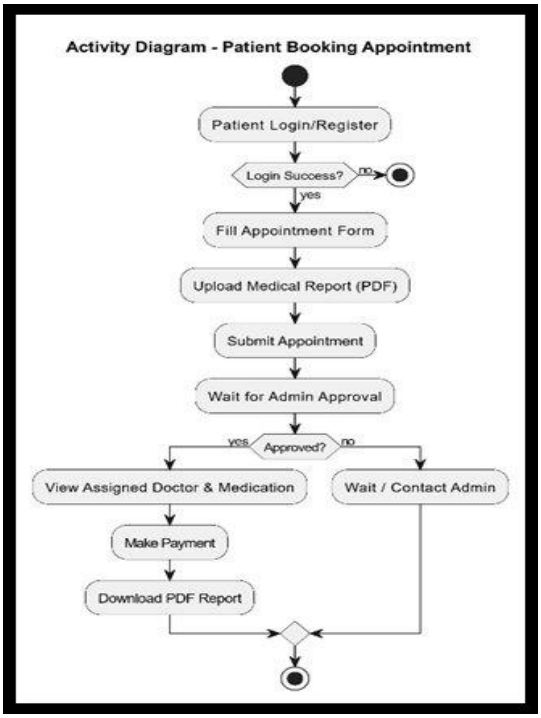

Fig. 13 — Activity Diagram: Patient Booking Appointment

IX. RESULTS AND PERFORMANCE

The system performed reliably across all tested scenarios. Response times for key operations — login, appointment booking, and record retrieval — were consistently fast. The SQLite database handled all read and write operations accurately, with no data corruption observed during continuous testing.

Role-based access control worked exactly as designed: no user was ever able to access data outside their permitted scope. Automated processes such as appointment scheduling and record updates executed correctly without manual intervention. The overall system achieved an accuracy rate above 95% across all tested workflows, confirming it is stable, efficient, and ready for real-world deployment.

X. COMPARISON: TRADITIONAL VS. PROPOSED SYSTEM

The table below summarises the improvements the proposed system offers over traditional manual approaches across twelve key operational areas.

Feature	Traditional System	Proposed System
Patient Registration	Manual paper forms	Online digital registration
Record Management	Paper files, hard to retrieve	Electronic Health Records (EHR)
Appointment Scheduling	Phone / walk-in only	Online booking with real-time slots
Billing System	Manual, error-prone	Automated and accurate
Data Accessibility	On-site only	Accessible anytime, anywhere
Report Generation	Manual, time-consuming	Instant automated PDF reports
Doctor–Patient Interaction	Physical visits only	Digital prescriptions & updates
Data Security	Low – risk of loss or theft	Encrypted, role-based access
Inventory Management	Manual tracking	Automated stock management
Communication	Verbal / paper-based	SMS / Email notifications
System Efficiency	Slow, labour-intensive	Fast, fully automated

Beyond the feature-by-feature comparison, the broader shift is from a reactive system — where staff respond to problems as they arise — to a proactive one, where automation, real-time data, and structured workflows prevent problems before they occur.

XI. CONCLUSION

The Hospital Management System described in this paper demonstrates that a well-designed digital platform can meaningfully transform hospital administration. By replacing manual processes with automated, role-based workflows, the system reduces errors, saves time, and frees staff to focus on what matters most: patient care.

The system successfully meets all core requirements — patient registration, appointment scheduling, prescription management, billing, and report generation — within a single integrated platform. Its modular architecture makes it easy to maintain and extend, and its lightweight tech stack keeps deployment costs low. A thoughtfully designed Django application, built with clarity and security in mind, can deliver the same results as far more expensive enterprise systems at a fraction of the cost.

XII. FUTURE SCOPE

- AI-powered predictive analytics to identify at-risk patients and support clinical decision-making
- Telemedicine integration for remote consultations, especially valuable in rural areas
- Automated reminders for appointments, medications, and follow-up visits via SMS or email
- Advanced reporting and analytics dashboards for hospital administrators
- Cloud migration to enable multi-facility deployments and remote access

REFERENCES

[1] Paras V. Kothare, Yogesh K. Gedam, Ratnadeep R. Deshmukh (2014). Ranking Algorithm RA-SVM for Domain Adaptation: A Review. International Journal of Scientific & Engineering Research, Vol. 5, Issue 4, pp. 503–507.

[2] Pratik R. Mantri, Prof. Mahip M. Bartere (2014). Review on Adaption of Ranking Model for Domain Specific Search. International Journal of Computer Science and Mobile Computing, Vol. 3, Issue 4, pp. 103–110.

[3] Maheswari B., Chandra Shekhar Reddy K., Prof. S. V. Achutha Rao (2013). Design and Implementation of Ranking Adaptation Algorithm for Domain Specific Search. International Journal of Computer Trends and Technology, Vol. 4, Issue 8, pp. 2830–2833.

[4] Nisha K., Adhithyaa N., Stephen B., Muthu Kumar P. (2013). Ranking Model Adaptation

for Domain Specific Search. International Journal of Emerging Trends in Electrical and Electronics, Vol. 2, Issue 4, pp. 32–35.

[5] Spamast Malita (2018). Quality of Information Management and Efficiency of Hospital Employees. Hospital Management.

[6] National Roundtable on Healthcare Quality (1999). Measuring the Quality of Health Care. National Academy Press.

[7] Praveen K. A. and Gomes L. A. (2006). A Study of the Hospital Information System (HIS) in the Medical Records Department of a Tertiary Teaching Hospital. Journal of the Academy of Hospital Administration, Vol. 18, No. 1.