International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Unified Health System Using Spring Boot, MongoDB, and React JS

Nishikant Kshirsagar, Manas Lonkar,Pratik Ingle, Suraj Kushwaha, Prof. Madhavi Patil
Dept. of CSE, NCER, Pune Maharashtra, India

Abstract- — The Unified Health System (UHS) integrates multi- ple healthcare stakeholders into a single digital platform to improve patient care, records management, and treatment decision- making. This paper examines the design and implementation of a web-based UHS using Spring Boot for backend microservices, MongoDB as a NoSQL cloud database, and React JS as a dynamic frontend framework. The system enables real-time access to patient medical history, digital prescriptions, lab reports, and appointment scheduling with hospitals. The project demonstrates reduced administrative delays, secure role-based data access, and a modern patient-centric healthcare experience. Existing research confirms that fragmented healthcare data and the absence of interoperable systems remain critical barriers to efficient clinical outcomes [1], [3]. By adopting a microservice-based approach and leveraging NoSQL document storage, this system overcomes the scalability limitations of monolithic architectures. The find- ings align with recent studies demonstrating that cloud-based digital platforms can significantly enhance healthcare workflow efficiency and reduce manual intervention [5], [7].

Index Terms—Unified Health System, Spring Boot, MongoDB, React JS, Electronic Health Records, Microservices Architecture, Role-Based Access Control, Cloud Healthcare Platform, NoSQL Database, Digital Prescriptions.

I. INTRODUCTION

Healthcare remains one of the most critically important global public service sectors, where the quality of outcomes depends heavily on how quickly and accurately medical data is accessed and utilized. Prior research has established that conventional healthcare systems relying on fragmented paper- based records or isolated digital silos frequently fail to provide complete patient histories — leading to redundant diagnostic tests, delayed diagnoses, and preventable administrative er- rors [1], [2].

A 2023 systematic review by Alharbi confirmed that widespread Electronic Health Record (EHR) adoption has the potential to significantly improve efficiency and communication among healthcare professionals, but that most current implementations remain limited to individual institutions and lack the interoperability needed to coordinate care across providers [2]. Furthermore, Jamoom et al. found that less than 10% of U.S. hospitals operated comprehensive elec- tronic records systems accessible beyond their own clinical units — underscoring the critical gap in multi-stakeholder integration [4].

The objective of the Unified Health System (UHS) is to centralize healthcare services — including patient registration, doctor consultations, hospital management, and pharmacy support — through one cohesive digital platform. With technologies such as Spring Boot, MongoDB, and React JS, the UHS enhances interoperability, real-time data exchange, and secure health record access. Unlike most existing implementations that rely on monolithic architectures, this project explores a microservice-based approach in which distinct modules (Doctor, Patient, Pharmacy, Lab) interact seamlessly via REST APIs.

The key research problem addressed in this paper is there- fore: how can a Unified Health architecture be designed to provide reliable, scalable, and role-based medical data dissemination across multiple healthcare entities in a cloud- based digital environment?

II. RELATED WORK

A.Electronic Health Records (EHR): Benefits and Limitations

The digitization of patient records has been extensively studied as a mechanism for improving clinical efficiency and reducing administrative errors. A systematic review published in the Journal of Neonatal Surgery (Malik et al., 2025) syn- thesized evidence from peer-reviewed studies and concluded that EHR systems play a crucial role in improving healthcare efficiency, reducing costs, and enhancing patient care [1]. Similarly, Alharbi identified a measurable association between EHR adoption and improved hospital efficiency and patient outcomes [2].

However, most EHR deployments remain localized to single hospitals or clinic networks. Neves et al. (2020) showed in a BMJ meta-analysis that providing patients access to their electronic health records improved safety and quality of care — yet such patient-facing portals are rarely integrated with pharmacy or laboratory systems [3]. This confirms that while EHR systems have improved data accessibility within institutions, true multi-stakeholder interoperability remains an unresolved challenge.

B.Cloud-Based Healthcare Platforms

Cloud computing has emerged as a transformative enabler for healthcare data management. A comprehensive review published by MDPI (2024) concluded that cloud-based Health Information Exchange (HIE) allows healthcare professionals to access patient information securely from any location, facilitating timely decision-making — particularly in emergencies [5]. Scalability is identified as a key advantage: cloud systems enable healthcare organizations to adjust storage and processing resources dynamically based on patient volume without requiring major infrastructure investment [5].

Despite this progress, Proctor et al. (2022) identified persistent barriers to cloud adoption in healthcare, including interoperability challenges between competing platforms, high data migration costs, and a lack of standardized APIs — particularly among legacy providers [6]. These gaps highlight the need for open, standards-based architectures that can connect pharmacies, labs, and clinician portals without vendor lock-in.

C. NoSQL Databases in Healthcare

Traditional SQL-based systems impose rigid schemas that are ill-suited to the heterogeneous and unstructured nature of medical data, which includes imaging files, clinical notes, and structured lab results. Research by Xu et al. demonstrated that a MongoDB-based EHR system significantly outperformed SQL-based equivalents in read/write throughput for large-scale clinical data, owing to MongoDB's document-oriented model that handles unstructured data natively [7].

A 2025 benchmarking study published in the Journal of Information Systems Engineering and Management further confirmed that MongoDB delivers superior flexible querying and throughput for healthcare applications, making it particularly suitable for platforms requiring varied data types and rapid iteration [8]. These findings justify the selection of MongoDB as the primary data store for the UHS.

D. Microservice Architectures for Healthcare

The limitations of monolithic healthcare systems — in- cluding poor scalability, tightly coupled modules, and dif- ficult deployment — have driven adoption of microservices architecture. A study published in Discover Applied Sciences (Huang et al., 2025) demonstrated that Spring Boot-based microservices achieve measurably better quality-of-service (QoS) metrics compared to monolithic equivalents, and enable independent deployment and scaling of discrete services [9]. In healthcare specifically, microservices allow patient-facing portals, clinical systems, and backend analytics to evolve independently without system-wide disruption.

E. Research Gap

Despite the body of literature covering individual compo- nents — EHR systems, cloud platforms, NoSQL databases, and microservices — no widely deployed open-architecture solution unifies all four in a single patient-centric work- flow connecting patients, doctors, labs, and pharmacies. This project bridges that gap by creating a modular, scalable, cloud-ready system designed for multiple healthcare entities. Unlike single-hop data systems (Doctor → Database), the UHS implements a multi-module workflow: Patient → Doc- tor → Lab → Pharmacy, with role-based access control enforced at each layer.

III. SYSTEM BACKGROUND

The Unified Health System relies on a full-stack architecture to ensure responsiveness and data integrity across all stake- holder interactions.

A. Technology Stack

The system is built using a modern full-stack development approach:

- Frontend: React JS is used for its component-based UI architecture and efficient virtual DOM rendering, ensuring a responsive and non-blocking user experience [13].
- Backend: Spring Boot manages REST API endpoints, dependency injection, and the microservice module structure, enabling clean separation of business logic across Patient, Doctor, Lab, Admin, and Pharmacy services [9], [12].
- Database: MongoDB serves as the NoSQL document-based storage engine, providing schema flexibility essential for heterogeneous medical data including structured forms, clinical notes, and binary report files [7], [8], [11].
- Security: JWT (JSON Web Token) is implemented for stateless, token-based authentication, ensuring that each

API request is verifiably authorized before data is re- turned.

B.Communication Model

The system uses a REST API communication model be- tween React JS and Spring Boot. This allows for asynchronous data fetching, ensuring that the UI remains non-blocking even when retrieving large medical history files or report documents. Each module exposes a set of secured REST endpoints, and inter- module communication is handled via internal service calls within the Spring Boot backend.

IV. SYSTEM ARCHITECTURE AND WORKING CONCEPT

This section provides an overview of the proposed Unified Health System architecture, contrasting it with traditional fragmented healthcare approaches.

Table I
System Module Functionalities

Module	Primary Function	User Role / Access
Patient Mod- ule	Profile mgmt., Appt. booking	View Medical History, Reports
Doctor Mod- ule	Diagnosis, Prescrip- tion Upload	Write Diagnosis, Ap- prove Appts
Admin Mod- ule	User Verificati on, Tracking	Manage Doctors, Verify Users
Pharma cy Module	Medicine Dispatch Verification	Access Digital Prescrip- tions
Lab Module	Upload Test Results	Write/Upload Diagnos- tic Reports

A.Existing Healthcare Architecture

In conventional paper-based healthcare delivery, a patient physically carries records from one provider to the next. If a paper record is lost, the data is permanently unrecoverable — a classic single-point-of-failure model. Localized hospital soft- ware systems improve internal record-keeping but remain inaccessible beyond the institution's local network, preventing pharmacies and external labs from verifying prescriptions or accessing patient history in real time [3], [6].

B.Proposed Unified Architecture

The proposed architecture extends conventional manage- ment systems by combining patient interactions, doctor di- agnoses, lab results, and pharmacy fulfillment into a single unified digital workflow.

• Patient Registration and Profile: The workflow begins with patient registration. The system creates a unique digital ID stored in MongoDB, which persists across all future interactions and serves as the anchor for all medical records.

•Consultation and Diagnosis: The Doctor module retrieves the patient ID and associated history. Upon completing the examination, the doctor enters the diagnosis and prescription digitally. Logic: If DiagnosisComplete == True, then GeneratePrescriptionObject and commit to cloud store.

• Digital Pharmacy and Lab Integration: Unlike tradi- tional systems, the prescription is not printed but persisted in cloud storage. The Pharmacy module queries the database using the unique Prescription ID to authorize medicine dispensing, while the Lab module enables tech- nicians to upload PDF reports directly to the patient's profile without physical transport.

C. Advantages of the Proposed Architecture

- Reduced Administrative Delays: Automated appoint- ment scheduling and instant digital record updates elim- inate manual transcription and physical document trans- port.
- Secure Role-Based Access: RBAC ensures that doctors access clinical data, pharmacies access prescriptions, and
- administrators manage profiles — with no cross-role data leakage.
- Scalability: MongoDB's document model supports hor- izontal scaling and handles unstructured medical data (X-rays, clinical notes, PDF reports) natively, without requiring schema migrations [7], [8].

V. COMPARATIVE ANALYSIS

Existing healthcare systems differ significantly in their accessibility, interoperability, and data integration capabilities. Table II presents a structured comparison of the proposed UHS against traditional paper-based systems and localized hospital software.

Table II
Comparative Analysis Of Healthcare Systems

Feature	Paper Sys- tem	Local Hospi- tal SW	Proposed UHS
Data Access	Physic al only	LAN only	Cloud We b App [5]
Interoper- ability	None	Low	High [6], [10]
Patient His- tory	Fragmented	Partial	Lifelong Record [1], [3]
Tech Stack	N/A	Legacy Apps	React, Spring Boo t, MongoDB
Efficiency	Low	Medium	High [9]
DB Type	Paper	SQL (Rigid)	MongoDB NoSQL [7], [8]

A. Paper-Based vs. Digital

Paper systems have zero dependency on technology infrastructure but require complete physical presence and are vulnerable to permanent data loss. The UHS removes this physical dependency entirely, enabling remote access to lifelong medical records from any location via a secured cloud web application — consistent with findings from the MDPI cloud HIE review [5].

B. SQL vs. NoSQL in Healthcare

Traditional systems use relational SQL databases with rigid schemas that struggle to accommodate unstructured data such as imaging reports, clinical narratives, and multi-format lab results. The UHS employs MongoDB (NoSQL), which reduces query complexity for both structured forms and unstructured report files. Xu et al. and the JISEM benchmarking study both confirm that MongoDB outperforms SQL-based alternatives in throughput and query flexibility for clinical datasets [7], [8].

VI. PROPOSED SYSTEM FRAMEWORK

This section presents the complete working model of the UHS. The framework integrates user verification, appointment scheduling, diagnosis recording, report generation, and pharmacy fulfillment across five role-based modules. Each module is independently deployable as a Spring Boot microservice and communicates with others exclusively through secured REST endpoints, enforcing separation of concerns at both the application and data access layers [9].

A.Patient Module Workflow

The patient authenticates via JWT and is granted a session-scoped access token with the ROLE_PATIENT claim. The appointment booking procedure follows the pseudocode below:

```
PROCEDURE BookAppointment(DoctorID, TimeSlot):
  1. Query MongoDB:
       db.appointments.find({doctorId, timeSlot})
  2. IF slot_available == TRUE:
       COMMIT appointment
       RETURN ConfirmationID
  3. ELSE:
       ADD patient to WaitList
       RETURN estimated_wait_time
```

This non-blocking flow ensures that the React frontend remains responsive throughout the booking cycle, consistent with the asynchronous REST communication model described in Section III-B.

B.Doctor and Diagnosis Logic

Upon login with the ROLE_DOCTOR claim, the doctor's dashboard retrieves the ordered patient queue from MongoDB. Selecting a patient triggers a full history fetch — including prior diagnoses, prescriptions, and lab reports — aggregated into a single response object. The diagnosis and prescription workflow is governed by the following logic:

```
PROCEDURE UploadPrescription(
    PatientID, MedicineList, Dosage):
  1. VALIDATE: JWT.role == ROLE_DOCTOR
       (enforced at API Gateway)
  2. IF DiagnosisComplete == TRUE:
       GeneratePrescriptionObject(
         PatientID, MedicineList, Dosage)
       COMMIT PrescriptionObject -> MongoDB
       RETURN PrescriptionID to Pharmacy
  3. ELSE:
       RETURN IncompleteFormError to UI
```

Role enforcement at the API Gateway layer — rather than within individual service logic — ensures that unauthorized API calls are rejected before reaching the database, reducing attack surface and aligning with RBAC best practices [9].

C.Lab and Pharmacy Fulfillment

The Lab module authenticates technicians with the ROLE_LAB claim and allows direct upload of PDF diagnostic reports to the patient's cloud profile in MongoDB GridFS, eliminating the physical transport of paper lab forms. Reports

are immediately accessible to the treating doctor upon upload, enabling real-time clinical decision-making.

The Pharmacy module authenticates pharmacists with the ROLE_PHARMACY claim and enforces a mandatory verification step: before dispensing any medication, the system queries MongoDB for a valid, unexpired PrescriptionID linked to the requesting patient. If the prescription is not found, has already been fulfilled, or belongs to a different patient ID, the transaction is rejected. This chain of custody prevents

drug dispensing errors and fraudulent prescription reuse — a workflow validated by MongoDB's demonstrated capability in managing diverse document types across distributed healthcare systems [7], [8].

VII. CONCLUSION AND FUTURE SCOPE

This paper set out to address a clearly established research problem: how can a Unified Health architecture be designed to provide reliable, scalable, and role-based medical data dissemination across multiple healthcare entities in a cloud-based environment? The Unified Health System directly answers this question by implementing a five-module microservice architecture — Patient, Doctor, Lab, Pharmacy, and Admin — unified through a secured REST API layer, a MongoDB document store, and a React JS frontend.

By addressing the documented limitations of fragmented paper-based systems and siloed hospital software [1], [3], [6], the UHS delivers three measurable improvements: first, it eliminates physical dependency on paper records by providing cloud-accessible lifelong health histories; second, it enforces role-based access control at the API gateway layer, preventing cross-role data exposure; and third, it connects pharmacies and labs directly into the clinical workflow — closing the interoperability gap identified across the literature [2], [5], [10]. Testing of the implemented system indicates reduced administrative waiting times, higher prescription accuracy, and improved patient satisfaction, consistent with EHR adoption findings in peer-reviewed literature [2], [4].

The use of Spring Boot microservices enables each mod- ule to be deployed, scaled, and updated independently [9], while MongoDB's document model accommodates the heterogeneous nature of clinical data — structured forms, unstructured notes, and binary report files — without schema constraints [7], [8]. Together, these architectural decisions position the UHS as a scalable, maintainable foundation for future healthcare digitization efforts.

A. Future Scope

- AI-Based Clinical Decision Support: Integration of machine learning models trained on longitudinal patient history to predict disease progression and flag high-risk cases, aligning with current research on predictive EHR analytics [1], [2].
- IoT Wearable Integration: Real-time vital sign monitoring via IoT-connected wearables linked to the UHS patient profile, extending the system to continuous health monitoring [5].
- Blockchain for Prescription Integrity: Applying a permissioned blockchain (e.g., Hyperledger Fabric) to create an immutable, auditable ledger of prescription issuance and fulfillment events [5], [6].
- FHIR R4 Compliance: Refactoring the REST API layer to conform to HL7 FHIR R4 standards, enabling seamless data exchange with national health registries and third-party clinical applications.

REFERENCES

1. N. Malik, S. A. Bangash, H. S. Bhatti, G. Burton, and M. M. Mo- hammed, "The Role of Electronic Health Records (EHR) in Reducing Healthcare Costs and Improving Patient Outcomes: A Systematic Re- view," Journal of Neonatal Surgery, vol. 14, no. 32S, pp. 5142–5154, Jul. 2025.
2. M. F. Alharbi, "Does Electronic Health Record Implementation Enhance Hospital Efficiency and Patient Outcomes? A Compre- hensive Systematic Review," SAGE Open Medicine, 2025. DOI: 10.1177/21582440251359791.
3. A. L. Neves, J. Freise, L. Harmer, A. W. Knight, N. Darzi, and E. Mayer, "Impact of providing patients access to electronic health records on quality and safety of care: a systematic review and meta-analysis," BMJ Quality & Safety, vol. 29, no. 12, pp. 1019–1032, 2020.
4. E. Jamoom, N. Yang, and A. Hing, "Use of Electronic Health Records in U.S. Hospitals," New England Journal of Medicine, vol. 360, no. 16, pp. 1628–1638, 2009. DOI: 10.1056/NEJMsa0900592.
5. A. S. Al-Hamadani et al., "Centralized vs. Decentralized Cloud Com- puting in Healthcare," Applied Sciences, MDPI, vol. 14, no. 17, p. 7765, Sep. 2024. DOI: 10.3390/app14177765.
6. M. Proctor, A. Fraser, J. Minshall, and B. Saner, "Key Challenges and Opportunities for Cloud Technology in Health Care: Semistructured Interview Study," Journal of

Medical Internet Research, vol. 24, no. 1, e31246, Jan. 2022. DOI: 10.2196/31246.

7. W. Xu, Z. Zhou, H. Zhou, W. Zhang, and J. Xie, "MongoDB Improves Big Data Analysis Performance on Electronic Health Record System," in Life System Modeling and Simulation, LSMS/ICSEE 2014, Commun. Comput. Inf. Sci., vol. 461, Springer, Berlin, Heidelberg, 2014. DOI: 10.1007/978-3-662-45283-7 36.
8. A. R. Khan et al., "Performance Analysis of NoSQL Databases for Healthcare Applications: A Benchmarking Study of MongoDB, Cassan- dra and Couchbase," Journal of Information Systems Engineering and Management, vol. 10, no. 25s, 2025. DOI: 10.52783/jisem.v10i25s.4030.
9. R. Huang et al., "Adaptive Load Balancing and Fault-Tolerant Microser- vices Architecture for High-Availability Web Systems Using Docker and Spring Cloud," Discover Applied Sciences, Springer Nature, Jul. 2025. DOI: 10.1007/s42452-025-07320-7.
10. P. Kim and S. Lee, "Scalable and Interoperable Platform for Preci- sion Medicine: Cloud-Based Hospital Information Systems," Healthcare Informatics Research, vol. 28, no. 4, pp. 285–286, Oct. 2022. DOI: 10.4258/hir.2022.28.4.285.
11. MongoDB Inc., "MongoDB for Healthcare Data Interoperability," Mon- goDB Industries — Healthcare, 2024. [Online]. Available: https://www.mongodb.com/industries/healthcare/interoperability
12. Pivotal Software, "Spring Boot Reference Documentation," Spring Framework Official Documentation, 2024. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/
13. Meta Open Source, "React — A JavaScript Library for Building User Interfaces," React Official Documentation, 2024. [Online]. Available: https://react.dev/