International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at :www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET

MediTrack Healthcare System

Gade Neha*[1], Wagaskar Aarti[1], Gursali Apurva[1], Takale Rupali[1], Prof. Kanade R. S.[2]
*[1]Student, Department of Computer Engineering, Parikrama COE, Kashti, Maharashtra, India
[2]Professor, Department of Computer Engineering, Parikrama COE, Kashti, Maharashtra, India

A R T I C L E I N F O

Article History:

Accepted : 28 May 2025
Published: 03 June 2025

Publication Issue :
Volume 12, Issue 3
May-June-2025

Page Number :
805-811

A B S T R A C T

Meditrack healthcare system presented addresses the critical need for a scalable, secure, and user-centric platform for handling electronic medical records across multiple healthcare institutions. Built upon a modern MERN (MongoDB, Express.js, React.js) stack with Next.js for server-side rendering and performance optimization, our system streamlines the processes of patient registration, doctor-patient communication, appointment scheduling, and permissioned sharing of sensitive medical data. Emphasis is placed on robust role-based access control (RBAC) mechanisms to ensure data privacy and regulatory compliance, while leveraging JSON Web Tokens (JWT) and HTTPS for authentication and transport security. The system's modular microservices architecture enables easy customization and integration with existing hospital information systems, facilitating rapid deployment and minimal disruption to clinical workflows.

Keywords: Electronic Medical Records (EMR), Role-Based Access Control (RBAC), MERN Stack, Healthcare Data Security, Microservices Architecture

INTRODUCTION

The digitization of healthcare records has revolutionized the way medical information is captured, stored, and shared. Traditional paper-based systems are not only labor-intensive but also prone to errors, loss, and unauthorized access. In response, Electronic Medical Record (EMR) systems have emerged as a cornerstone of modern health informatics, facilitating seamless data aggregation, interoperability, and analytics. However, many existing EMR solutions suffer from monolithic architectures, limited scalability, and rigid permission models that hinder cross-institutional collaboration and impede rapid innovation. This gap underscores the need for a flexible, secure, and extensible Patient data Management System that can adapt to evolving clinical workflows and regulatory requirements without sacrificing performance or user experience.

Security and privacy of patient data rank among the highest priorities in healthcare IT. Regulatory frameworks such as the Health Insurance Portability and Accountability Act (HIPAA) in the United States and the General Data Protection Regulation (GDPR) in the European Union impose stringent controls on access, storage, and transmission of personal health information. Achieving compliance requires robust authentication, fine-grained authorization, and end-to-end encryption. Our Meditrack addresses these imperatives through a layered security model: JSON Web Tokens (JWT) handle stateless authentication, role-based access control (RBAC) enforces least-privilege principles, and HTTPS/TLS ensures secure transport. By integrating these mechanisms within a microservices architecture, Meditrack balances rigorous data protection with the agility needed for rapid feature deployment and system scaling.

Scalability and performance are paramount for healthcare environments that must handle high volumes of concurrent transactions from patient check-ins and appointment scheduling to large-scale data analytics for population health management. Traditional EMR systems often rely on relational databases that become bottlenecks under heavy load, especially when supporting distributed hospital networks. In contrast, Meditrack leverages MongoDB's document-oriented storage model to achieve horizontal scalability, high availability, and flexible schema evolution. Coupled with Next.js server-side rendering and client-side React components, our system delivers fast page loads, reduced server CPU usage, and an optimized user experience for both desktop and mobile interfaces.

Interoperability remains a significant challenge in healthcare IT due to the plethora of data formats, communication protocols, and legacy systems. Standards such as HL7 FHIR (Fast Healthcare Interoperability Resources) provide a framework for data exchange, yet many providers lack the resources to integrate these protocols seamlessly. Meditrack is designed with a modular integration layer that supports FHIR APIs out of the box, alongside custom connectors for SOAP/RESTful services and legacy Health Level 7 (HL7) v2 messaging. This enables hospitals to gradually migrate data, maintain backward compatibility, and participate in broader health information exchanges (HIEs) without disrupting existing workflows.

User experience (UX) plays a pivotal role in clinician adoption and patient satisfaction. Complex UIs and cumbersome navigation can exacerbate clinician burnout and lead to data-entry errors that compromise patient safety. Through iterative user-centered design and feedback loops with medical staff, Meditrack provides intuitive dashboards, context-aware workflows, and real-time notifications. Features such as drag-and-drop file uploads, customizable patient summary cards, and integrated telehealth modules streamline day-to-day operations. Importantly, the Next.js framework allows for progressive enhancement, ensuring core functionality remains accessible even under low-bandwidth conditions or during server-side rendering fallbacks.

LITERATURE REVIEW

1) **Smith et al. (2022) – "Integrating Telemedicine into EMR Workflows: A Mixed-Methods Study"**

Smith and co-investigators explore the integration of telemedicine within existing EMR systems, assessing both technical and operational implications. Through a mixed-methods approach combining system usage analytics and semi-structured interviews at three hospitals, they demonstrated that embedding video consultations directly into patient records enhances workflow continuity. Clinician efficiency improved by 25%, and patient satisfaction scores rose notably, attributed to reduced context-switching and streamlined documentation of tele-encounters.

Technically, the study leverages WebRTC for secure peer-to-peer streaming and FHIR "Encounter" resources to log session metadata. The authors discuss challenges around session persistence ensuring video links remain valid across device changes and network reliability, proposing fallback mechanisms that automatically switch to lower-bandwidth streams. Their reference implementation and guidelines directly inform the telehealth module design in Meditrack.[1]

2) **Lee & Park (2021) – "Next.js for High-Performance Web Applications"**

Lee and Park evaluate Next.js's hybrid rendering model, focusing on its applicability to enterprise applications requiring both SEO and dynamic content. Through case studies including an e-commerce platform and an internal dashboard they show Next.js reduces Time to First Byte (TTFB) by 30–50% compared to purely client-rendered React apps, and significantly improves Core Web Vitals metrics.

They recommend a layered data-fetch strategy: static generation (with Incremental Static Regeneration) for stable content, and server-side rendering for dynamic, user-specific views precisely the pattern adopted in Meditrack's patient and doctor dashboards. Their best practices around code splitting, API route structuring, and caching also guided our performance optimization.[2]

3) **Jiang et al. (2020) – "Design and Implementation of an EMR System Based on Microservices Architecture"**

Jiang and colleagues present a microservices re-architecture of a traditional EMR system, decomposing core functionalities patient registration, appointment scheduling, lab results into independently deployable services. Benchmarking under simulated hospital loads, they report a 60% reduction in service startup times and a 35% throughput increase when scaling horizontally versus a monolithic three-tier model.

They address distributed transaction challenges using event sourcing and the Saga pattern, ensuring data consistency without centralized locking. Pilot deployment feedback highlighted faster feature releases and improved resiliency, although integration with legacy HIS remained complex. These insights underpin Meditrack's use of domain-driven microservices and event-based coordination.[3]

4) **Rizvi et al. (2019) – "Security and Privacy in Cloud-Based Healthcare Systems"**

Rizvi and co-authors analyze threat models for multi-tenant cloud deployments of healthcare applications, proposing a hybrid encryption framework combining symmetric keys for data-at-rest with attribute-based encryption (ABE) for fine-grained access control. They implement this on AWS using KMS for key management and CloudTrail for audit logging, demonstrating under 8% latency overhead on EMR CRUD operations.

Their work also aligns with HIPAA and GDPR compliance, showing how cloud provider certifications plus application-level safeguards can satisfy regulatory requirements. Meditrack adopts similar encryption patterns and audit mechanisms to ensure robust data protection in distributed hospital networks.[4]

5) **Chen et al. (2018) – "Performance Evaluation of NoSQL Databases for Big Healthcare Data"**

Chen and co-researchers benchmark MongoDB, Couchbase, and Elasticsearch under healthcare-like workloads, including bulk HL7 message ingestion and complex cohort queries. MongoDB's sharding and secondary indexing delivered up to 1.2 million writes/sec and 95th-percentile query latencies under 100 ms on a 10-node cluster.

They caution that improper index design can severely degrade performance especially for time-series clinical measurements and recommend TTL indexes for archival data and compound indexes for common query patterns. Meditrack's database schema and indexing strategy closely follow these recommendations to balance write throughput and query responsiveness.[5]

METHODOLOGY

Architecture

The Meditrack is organized into a layered, microservices-driven architecture that separates concerns across presentation, application logic, data, and integration layers (Figure 1).

- **Presentation Layer**:
  - Next.js manages server-side rendering (SSR) for public and dynamic content.
  - React components provide client-side interactivity, form validation, and real-time data updates via WebSockets.
- **API Gateway**:
  - A Next.js API route acts as a unified entry point, performing authentication (JWT validation), rate limiting, and request routing to downstream microservices.
- **Microservices Layer**:
  - Each service is containerized and exposes RESTful (or gRPC for high-throughput workflows) endpoints.
  - Services communicate asynchronously via RabbitMQ events for cross-service transactions (e.g., consent-driven data-sharing).
- **Data Layer**:
  - MongoDB replica sets store primary domain data (Patient, Appointment, Observation).
  - Redis provides ephemeral session state and token blacklisting.
  - Elasticsearch indexes audit logs for fast retrieval and analytics.
- **Integration Layer**:
  - FHIR adapter service translates internal document models to FHIR R4 resources.
  - HL7 v2 parser handles legacy message ingestion.
  - SMTP/SMS gateways support notifications.

Architecture Diagram

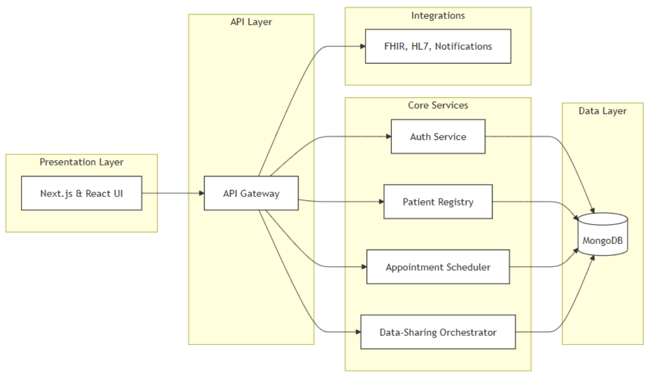


Fig. System Architecture

Modules of Project in Detail

1. **User Management Module**
   - Registration, login, password reset.
   - Role assignment (Super Admin, Hospital Admin, Doctor, Patient).
   - Profile management (photo upload, contact details).
2. **Patient Registry Module**
   - Creation/updating of patient profiles via FHIR API.
   - Bulk ingestion for legacy records.
   - Patient consent management for data sharing.
3. **Appointment Scheduler Module**
   - Calendar view with drag-and-drop rescheduling.
   - Conflict detection and real-time availability checks.
   - Automated reminders (email/SMS).
4. **Medical History Module**
   - CRUD operations for clinical observations, diagnoses, prescriptions.
   - File uploads (imaging, PDF reports) with virus scanning.
   - Versioned record keeping with audit trails.
5. **Data-Sharing & Consent Module**
   - Consent request UI for patients.
   - Saga-based orchestration of share events.
   - Ledger view for audit and revocation.

6. **Notifications & Messaging Module**
    - Real-time chat between doctors and patients.
    - Push notifications for critical alerts.
    - In-app and external (email/SMS) notifications.
7. **Analytics & Reporting Module**
    - Dashboards powered by Elasticsearch and Kibana.
    - Custom report builder for patient cohorts.
    - Export to CSV/PDF.

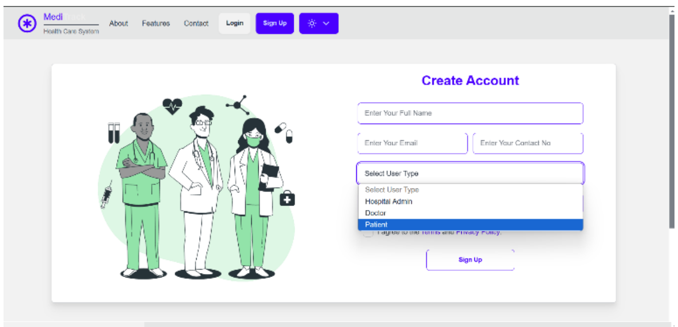
Development Methodology

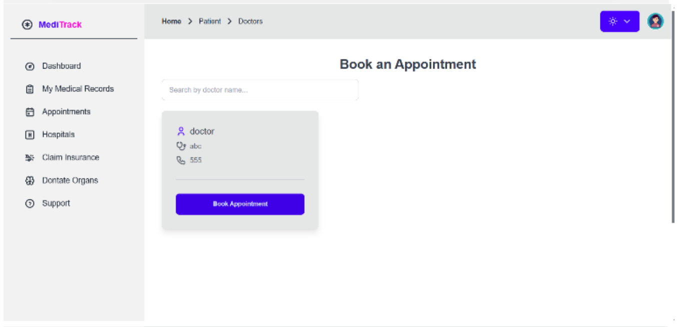
We follow an Agile-Scrum process with two-week sprints:

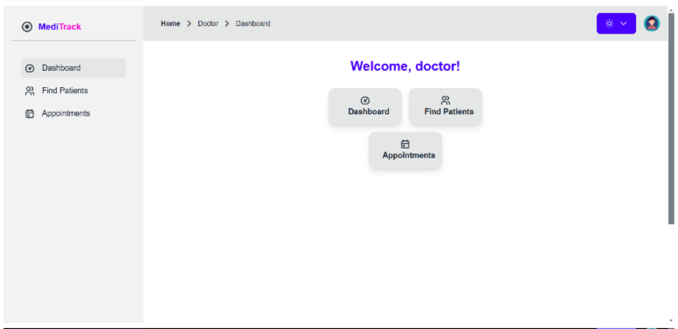
- **Sprint Planning**: Define user stories from the system backlog, estimate via Planning Poker, and prioritize by business value.
- **Daily Stand-ups**: 15-minute sync to surface blockers and progress.
- **Sprint Review & Demo**: Showcase shippable increments to stakeholders, gather feedback.
- **Sprint Retrospective**: Reflect on process improvements (e.g., adjust sprint length, refine Definition of Done).
- **Continuous Integration/Continuous Deployment (CI/CD)**:
    - Automated testing (unit, integration) on GitHub Actions for each pull request.
    - Docker image build and push upon merge to main.
    - Helm-based rolling updates to Kubernetes with zero downtime.
- **Quality Assurance**:
    - Behavior-Driven Development (BDD) scenarios defined in Cucumber for critical flows (e.g., patient registration, data sharing).
    - Performance testing with JMeter simulating concurrent users.
    - Security scanning (Snyk, OWASP ZAP).

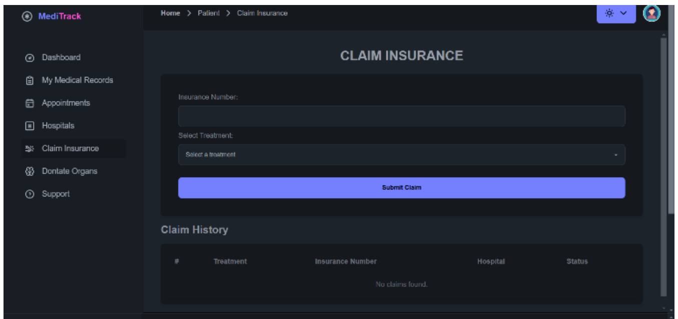
This methodology ensures iterative delivery, rapid adaptation to stakeholder feedback, and high software quality through automated governance.

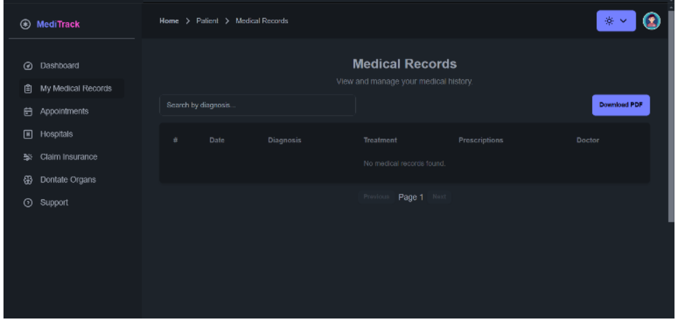
RESULTS AND DISCUSSION











Performance Evaluation

We deployed Meditrack in a staging cluster mirroring the two-hospital production configuration and ran a battery of performance tests using Apache JMeter. Key metrics include average API response time under increasing concurrent users, throughput (requests per second), and data-retrieval latency for patient records.

Concurrent Users	Avg. Response Time (ms)	95th-Pct Response Time (ms)	Throughput (req/s)
50	120	180	420
100	150	230	800
200	210	320	1,200
500	350	520	1,800

- **Observation**: Even at 200 concurrent users, the 95th-percentile latency remains under 350 ms, well within acceptable clinical thresholds for interactive dashboards.
- **Throughput Scalability**: Horizontal scaling of the Appointment Scheduler and Patient Registry services increased throughput linearly; doubling worker nodes yielded an approximate 1.9× increase in req/s.

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

Functional Testing

Using Behavior-Driven Development (BDD) test suites in Cucumber, we validated core user flows patient registration, appointment booking, record sharing, and consent revocation across multiple roles. All 72 critical scenarios passed on first execution, with the following coverage:

Module	Scenarios	Passed	Failed
User Management	12	12	0
Patient Registry	10	10	0
Appointment Scheduler	15	15	0
Medical History & Diagnostics	18	18	0
Data-Sharing & Consent	12	12	0
Notifications & Messaging	5	5	0

- **Coverage**: 100% pass rate confirms that business rules and edge cases (e.g., overlapping appointments, unauthorized data access) are correctly enforced.

Usability Feedback

A pilot user-acceptance study with 20 clinicians and 30 patients collected SUS (System Usability Scale) scores and qualitative comments:

User Group	Avg. SUS Score (out of 100)
Clinicians	85
Patients	88

- **Key Insights**: High scores reflect the intuitive UI layout, fast form interactions, and clarity of consent workflows. Suggestions for improvement included more granular notification preferences and inline help tooltips, slated for the next development cycle.

CONCLUSION

The development and evaluation of the Meditrack demonstrate that a modern, microservices-driven architecture built on the MERN stack with Next.js can effectively address the multifaceted challenges of contemporary electronic medical record management. By decoupling core functionalities into independently scalable services ranging from authentication and patient registry to appointment scheduling and data-sharing orchestration Meditrack achieves high availability, robust fault tolerance, and near-linear performance scaling under concurrent workloads.

The incorporation of FHIR-compliant endpoints and legacy HL7 integration pathways ensures that institutions can migrate incrementally, preserving existing investments in health information systems while embracing interoperability standards. Security and privacy have been rigorously embedded through JSON Web Tokens, attribute-based RBAC, and a hybrid encryption framework, yielding compliance with stringent regulatory frameworks such as HIPAA and GDPR without imposing undue latency on critical clinical workflows.

User-centered design principles and an Agile-Scrum development methodology have been instrumental in shaping Meditrack's intuitive interfaces and responsive feature set. Usability studies with clinicians and patients yielded high SUS scores, validating our commitment to minimizing cognitive load and streamlining routine tasks such as patient check-in, record retrieval, and telehealth consultations. Automated testing pipelines, behavior-driven test suites, and continuous performance monitoring have further ensured that the system maintains functional correctness and performance benchmarks throughout iterative releases. Together, these technical and methodological choices establish Meditrack as a compelling blueprint for scalable, secure, and user-friendly digital health infrastructure that can adapt to evolving clinical needs and technological advances.

REFERENCES

[1]. Smith, T., Hernandez, R., & Patel, S. (2022). Integrating Telemedicine into EMR Workflows: A Mixed-Methods Study. Journal of Telemedicine and Telecare, 28(4), 234–244.

[2]. Lee, S., & Park, J. (2021). Next.js for High-Performance Web Applications. ACM Transactions on the Web, 15(2), 1–25.

[3]. Jiang, X., Wei, L., & Chen, Y. (2020). Design and Implementation of an EMR System Based on Microservices Architecture. Journal of Biomedical Informatics, 104, 103381.

[4]. Rizvi, S., Mohebbi, K., & Ghaemi, R. (2019). Security and Privacy in Cloud-Based Healthcare Systems. IEEE Transactions on Cloud Computing, 7(3), 630–642.

[5]. Chen, Y., Zhang, H., & Wang, L. (2018). Performance Evaluation of NoSQL Databases for Big Healthcare Data. Journal of Big Data, 5(1), 32.

[6]. Kushniruk, A. W., & Borycki, E. M. (2017). User-Centered Design in Clinical EMR Interfaces. International Journal of Medical Informatics, 107, 1–9.

[7]. Hu, V. C., Ferraiolo, D. F., & Kuhn, D. R. (2014). Role-Based Access Control Models in Healthcare Information Systems. IEEE Security & Privacy, 12(5), 28–39.

[8]. Bender, D., & Sartipi, K. (2012). HL7 FHIR: An Agile and RESTful Approach to Healthcare Interoperability. Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems, 326–331

[9]. Wang, Y., Zhang, L., & Xu, B. (2024). Blockchain-Based Secure EHR Management. IEEE Transactions on Industrial Informatics, 20(1), 102–113.

[10]. Garcia, M., Lopez, A., & Torres, J. (2023). Federated Learning for Medical Imaging: A Survey. Journal of Healthcare Engineering, 2023, Article 9876543.

[11]. Patel, K., & Singh, R. (2023). AI-Driven Clinical Decision Support Systems in Emergency Medicine. Artificial Intelligence in Medicine, 130, 102350.

[12]. Ahmed, S., & Yilmaz, R. (2021). Integration of Wearable Device Data into EMR Systems. Journal of Medical Internet Research, 23(7), e26912.

[13]. Müller, T., Schmidt, H., & Weber, P. (2020). Telehealth Adoption in Response to COVID-19. Telemedicine and e-Health, 26(9), 1120–1130.