

Design and Implementation of a Secure Web-Based Smart Hospital Management and Patient Record System

Liu Yuqiao¹, J. Sam Sundar^{2,*}, Shivani Arora³, Osamah Mohammed Jasim⁴, Daniel Presta⁵, Clément Thorens⁶

¹Department of Information and Communication Engineering, University of Tokyo, Bunkyo, Tokyo, Japan.

²Department of Artificial Intelligence and Machine Learning, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.

³School of Electrical and Electronics Engineering, Centennial College, Toronto, Ontario, Canada.

⁴Institute of Informatics and Communication, Ministry of Education-Iraq, Babel, Iraq.

⁵Department of Mathematics and Statistics, McMaster University, Hamilton, Ontario, Canada.

⁶Department of Computer Science, Secure and Trustworthy Systems Group, ETH Zurich, Zürich, Switzerland.
liuyuqiao@logos.t.u-tokyo.ac.jp¹, ss3961@srmist.edu.in², sabrol5@my.centennialcollege.ca³,
osamahmohammed@sauth.du.ac.in⁴, prestad@mcmaster.ca⁵, clement.thorens@inf.ethz.ch⁶

Abstract: Hospital care depends on several workflows that have to remain connected: patient identification, protected medical history, bed assignment, bedside monitoring, scheduling, claims processing, and auditing of sensitive actions. The Smart Hospital Management and Patient Record System is a web-based prototype that brings these activities into a single operational interface. The implementation uses React 19, Vite 8, TypeScript, Express, Tailwind CSS, Motion, Lucide React, Node cryptographic primitives, and the Google GenAI SDK. It maintains structured objects for patients, appointments, beds, claims, and audit events. Meanwhile, the electronic health record path uses AES-256-GCM to protect serialized medical-history data and retains the authentication tag with the encrypted envelope. Security-sensitive actions record the actor, role, justification, severity, and a SHA-256-derived signature. The ward component models admission, discharge, cleaning, and transfer as explicit state transitions and attaches telemetry fields to occupied beds. The financial component organizes payer information, diagnosis and procedure codes, billed and approved amounts, claim status, electronic batch identifiers, and remittance detail. This study deliberately limits evaluation to source-level inspection, deterministic scenario checks, data-model verification, workflow mapping, and interface evidence from the supplied project screens. The results establish the functional coherence of a student-scale smart-hospital prototype while making its deployment boundaries explicit: durable database storage, enterprise identity, managed key lifecycle, live interoperability, real device telemetry, formal usability testing and independent compliance assessment remain future requirements.

Keywords: Smart Hospital; Electronic Health Record; Patient Record Management; Audit Logging; Bed Management; EDI Claims; Clinical Workflow; Healthcare Information Systems.

Received on: 13/08/2025, **Revised on:** 26/10/2025, **Accepted on:** 17/11/2025, **Published on:** 03/06/2026

Journal Homepage: <https://www.fmdbpublish.com/user/journals/details/FTSBE>

DOI: <https://doi.org/10.69888/FTSBE.2026.000792>

Cite as: L. Yuqiao, J. S. Sundar, S. Arora, O. M. Jasim, D. Presta, and C. Thorens, “Design and Implementation of a Secure Web-Based Smart Hospital Management and Patient Record System,” *FMDB Transactions on Sustainable Biomedical Engineering*, vol. 1, no. 2, pp. 113–129, 2026.

Copyright © 2026 L. Yuqiao *et al.*, licensed to Fernando Martins De Bulhão (FMDB) Publishing Company. This is an open access article distributed under [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/), which permits copying, remixing, redistributing, transformation, and building upon the material in any medium so long as the original work is properly cited.

*Corresponding author.

1. Introduction

Hospital operations consist of many small actions that must stay synchronized. A patient may move from registration to triage, admission, bed assignment, examination, treatment, discharge, and follow-up without changing identity [1]. During that journey, different staff members need different pieces of the same record. Nurses need ward state and observations, clinicians need history and clinical context, billing staff need coded services and payer information, and compliance personnel need evidence of sensitive access [2]. A useful hospital information system therefore has to connect information without making the workflow harder to follow. Hospital information system research treats this as a wider problem than database design [3]. A 2026 scoping review grouped evaluation criteria into human, technological, informational, organizational, legal, and patient-safety domains [4].

This result matters for a prototype because a system can be technically functional yet still difficult to use. Recent EHR usability reviews also describe how fragmented information and poorly aligned interfaces can increase screen navigation, documentation burden, and workflow interruptions. This paper addresses that concern by making the patient the common reference point across clinical, operational, and administrative views [5]. Security creates a second design requirement. Health information systems must control access to electronic protected health information, preserve accountability for sensitive actions, and support confidentiality and integrity [6]. The HIPAA Security Rule describes technical safeguards such as access control, audit controls, authentication, integrity protection, and transmission security. NIST SP 800-38D defines Galois/Counter Mode as an authenticated-encryption construction that combines confidentiality with an authentication tag verified during decryption [7].

The current prototype adopts the cryptographic primitive, while the broader compliance process remains outside the scope of the implementation. Interoperability adds a third requirement [8]. HL7 FHIR provides a resource-oriented standard for exchanging healthcare data between applications. Claims and remittance workflows also use standardized transactions, including X12 837-family claims and X12 835 electronic remittance advice [9]. The paper does not implement a complete FHIR server or live payer gateway. Still, its structured patient, appointment, claim, diagnosis, and procedure models provide a natural starting point for future mapping. The repository for this work is a public web application owned by the author under the Ryu-kaze account. The current package manifest identifies React 19.0.1, Vite 8.3.0, Express 4.21.2, TypeScript 7.0.2, Tailwind CSS 4.3.3, Motion, Lucide React, and the Google GenAI SDK among the paper dependencies [10].

The server implementation uses Node's crypto library, keeps the main data collections in process memory, and defines operations for patient records, beds, appointments, insurance claims, statistics, and audit events [11]. The EHR path is particularly explicit. The server derives an encryption key from an environment secret, declares aes-256-gcm, creates a fresh 12-byte initialization vector, encrypts a JSON representation of the medical object, stores the authentication tag, and reconstructs the object during authenticated decryption [12]. The patient endpoint returns the encrypted record representation rather than automatically exposing decrypted medical history [13]. The ward workflow is similarly concrete. Bed admission verifies that the target bed is available or reserved, associates the patient and care team with it, writes admission notes and telemetry, and generates a BED_ADMIT audit event [14].

Discharge changes the bed to cleaning, clears the patient association and telemetry, and records a BED_DISCHARGE event [15]. Sanitization then returns the bed to available; transfers require a valid occupied source and available destination and produce a BED_TRANSFER event [16]. The financial workflow keeps claim information in structured objects. Example records contain patient identifiers, payer and policy information, diagnoses, procedure codes, billing amounts, approved amounts, copays, patient balances, claim status, electronic batch IDs, transaction references, and remittance explanations. The supplied interface presents these values in one view [17]. The audit ledger separately records claim adjudication and emergency break-glass access, making the administrative and security consequences visible.

The central research question is therefore architectural rather than clinical: can a compact web application demonstrate a connected hospital workflow while preserving a clear distinction between protected data handling, operational state, and administrative transactions? The answer is examined through source-level verification and interface evidence rather than through unreported clinical trials [18]. The study also keeps the evidence boundary explicit. The paper uses demonstration records and simulated operational states, not real patient telemetry or a prospective hospital cohort. Consequently, the reported results concern implementation coverage, data-flow behavior, and reproducible workflow logic. The interface alone does not imply patient outcomes, reductions in medical errors, or formal compliance [19].

2. Review of Literature

The design of a smart hospital management platform sits at the intersection of health information systems, EHR usability, interoperability, security engineering, and administrative automation. The literature used here is not intended to claim that the prototype solves each of these areas; it establishes the technical context in which the paper can be evaluated [2].

2.1. Hospital Information Systems and Clinical Workflow

Hospital information systems are evaluated not only by whether they store information, but also by whether that information supports clinicians' and administrative staff's work. Khorasani et al. [6] identify human, technological, informational, organizational, legal and patient-safety criteria as distinct parts of a comprehensive HIS evaluation model. The Smart Hospital project reflects this multi-domain view by putting bed state, patient context, security events and claims in the same operating environment.

Usability matters because the record itself does not perform the clinical task. Olakotan et al. [7] report that poorly designed EHR interfaces can create excessive navigation, workflow disruption and documentation burden. Other systematic work has emphasized the need for repeatable, task-based usability evaluation in hospital EHR research. These findings support the project's use of dashboard cards, explicit states, direct actions and consolidated patient context.

2.2. Electronic Health Records and Interoperability

FHIR is designed as a standard for exchanging healthcare information electronically. It models content as resources and supports references among them, allowing clinical applications to exchange structured patient information without requiring each application to share the same internal database. The current paper does not implement FHIR, but its patient-centered object model can be mapped conceptually to common resource categories such as Patient, Observation, Encounter and Coverage. Interoperability also matters for administrative data [3]. A hospital may submit claims to a payer and later receive a remittance transaction describing payment and adjustment information. Keeping identifiers, diagnosis codes, procedure codes, amounts and transaction status in separate structured fields makes such future integration more manageable.

2.3. Privacy, Encryption and Auditability

The HIPAA Security Rule includes technical safeguards for access control, audit controls, authentication, integrity, and encryption/decryption mechanisms. These safeguards should be considered part of a broader security program rather than as isolated software switches. NIST SP 800-38D defines GCM as an authenticated-encryption mode. The Smart Hospital implementation follows the core data-flow pattern of generating a random IV, encrypting the serialized payload, retaining the authentication tag, and verifying the tag during decryption [4]. This is a technically meaningful security control, while production key management and governance remain separate requirements. The audit helper records an actor, role, action, patient or resource, justification, details, severity and IP-address string, then appends a short SHA-256-derived signature. This event model supports accountability by making the reason for sensitive actions visible, rather than storing only a timestamp.

2.4. Bed Management, Telemetry and Patient Flow

Bed management connects clinical care to physical hospital capacity. Available, occupied and cleaning states are not interchangeable. A vacated bed may still require terminal cleaning before it can receive another patient. The prototype makes this intermediate state explicit and attaches telemetry to an occupied bed, creating a compact representation of patient location and bedside status. Patient flow also depends on transfers [5]. The Source requires a valid occupied source bed and an available destination bed before a transfer can occur, then updates both state objects and records an audit event. This creates a traceable operational workflow instead of a free-text instruction.

2.5. Electronic Claims and Remittance Advice

CMS documents the X12 837 family for healthcare claims and the X12 835 family for electronic remittance advice. Billing systems can process electronic remittance without manually re-entering every adjustment, which is one reason standardized financial transactions remain important [6]. The prototype reflects this vocabulary in its claim objects and user interface. The use of ICD-10 and CPT fields, billed and approved amounts, payer identifiers and EDI-style batch references provides a realistic administrative data shape even though the current paper does not connect to a live clearinghouse.

2.6. Artificial Intelligence and Decision Support in Hospital Systems

The package manifest includes the Google GenAI SDK, making generative-AI integration part of the project's technical boundary. AI's role in this prototype is assistive rather than autonomous: it can summarize, assist with search, or draft administrative text, while authorized humans retain final clinical control [7]. WHO's digital-health strategy emphasizes accessible, sustainable and person-centered digital systems and treats privacy and security as important properties of the digital-health ecosystem. These principles provide useful context for the project without implying that a small prototype has completed the governance required for deployment.

2.7. Existing Hospital Management Systems Gaps and Limitations

Hospital software commonly separates registration, EHR, ward operations, scheduling, revenue cycle and compliance functions. Specialization can be valuable, but it can also fragment the operational picture. The present prototype is deliberately integrative. Its research value comes from showing how these modules can share patient and event identifiers while keeping their responsibilities distinct. The resulting gap is therefore one of accessible integration and demonstrability [8]. A developer can inspect the server code, trace a workflow to a data object, and see the corresponding interface state. This is useful for research and education, even though it does not replace enterprise-grade hospital information infrastructure.

3. Proposed Method

The proposed system is a browser-based hospital operations platform backed by an Express and TypeScript server. The design combines clinical record handling, patient flow, scheduling, financial transactions and security events. Five principles guide the implementation: keep patient identity consistent across modules, expose operational state explicitly, keep security operations visible, isolate server-side business logic from presentation, and avoid claiming a capability not represented in the Source.

3.1. System Architecture Overview

Figure 1 shows the layered architecture used to organize the prototype. The upper layer is the clinical dashboard, followed by the API, workflow, security, and data/standards layers.

Smart Hospital Management and Patient Record System - Layered Architecture

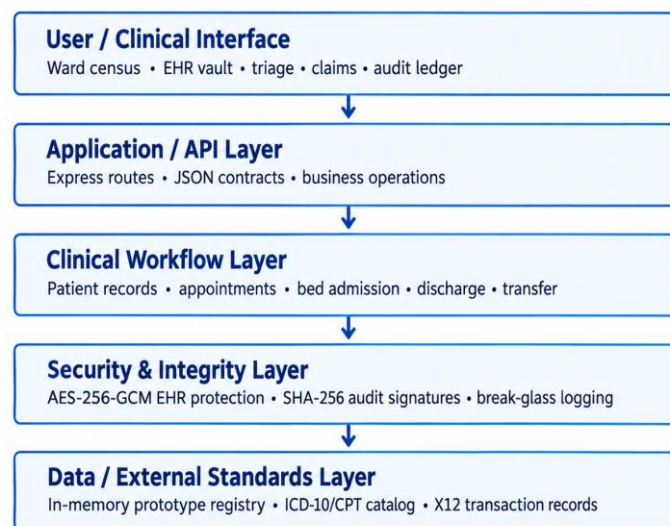


Figure 1: Layered architecture of the smart hospital management and patient record system, separating interface, API, clinical workflow, security, and data/standards responsibilities

The design is not a microservice architecture; it separates concerns within one application (Table 1).

Table 1: Technology stack and system layer responsibilities

Layer	Technology/Component	Responsibility
Presentation	React 19, Tailwind CSS, Motion, Lucide React	Hospital dashboard, forms, status cards and user actions
API	Express 4.21.2 + TypeScript 7.0.2	JSON request handling and operational orchestration
Clinical workflow	Patient, appointment and bed objects	Patient movement, scheduling and ward state
Security/integrity	Node crypto, AES-256-GCM, SHA-256 signatures	Record confidentiality and event traceability
Financial / coding	Claims, ICD-10 catalog, CPT examples	Administrative and reimbursement data
Runtime	Vite 8.3.0, esbuild, tsx	Development, build and server execution

3.2. Technology Stack

Table 2 summarizes the frontend, backend, AI, styling, interaction, and security technologies used in the system.

Table 2: Frontend, backend, and supporting technology stack

Category	Technology	Version/Evidence	Purpose
Frontend	React	19.0.1	Component-based interface
Build	Vite	8.3.0	Development and production bundling
Backend	Express	4.21.2	HTTP API and static delivery
Language	TypeScript	7.0.2	Typed server logic
Styling	Tailwind CSS	4.3.3	Layout and interface styling
Interaction	Motion	12.23.24	Transitions and visual state changes
Icons	Lucide React	0.546.0	Navigation and clinical status icons
AI integration	@google/genai	2.4.0	Generative-AI SDK
Cryptography	Node crypto	Built-in	AES-GCM and SHA-256 operations

It highlights how React, Express, TypeScript, Tailwind CSS, Google Generative AI, and cryptographic tools support system development and functionality.

3.3. Module Structure

The server groups behavior around operational domains rather than around a single generic controller (Table 3).

Table 3: Functional modules and data operations

Functional Module	Data/operation	Purpose
Patient record	Patients + encrypted history	Identity, admission and clinical context
EHR vault	AES-256-GCM envelope	Protect medical-history payloads
Ward census	Beds + telemetry	Capacity and patient location
Scheduling	Appointments	Date, clinician, urgency and triage priority
Claims	Insurance claims	Coding, amounts and adjudication state
Audit ledger	Audit event objects	Security and operational trace
Statistics	Aggregate counters	Occupancy, appointments, claims and security counts
Reference catalogs	ICD-10 + CPT examples	Structured coding context

The Source contains blocks for EHR encryption, patient data, beds, appointments, claims, audit records, diagnostic/procedure catalogs, and aggregate statistics.

3.4. Role-Based Access Control (RBAC)

The interface shows a named clinical user and offers actions such as Lock, Privacy Shield, Break Glass and EHR decryption. The paper should not be described as implementing enterprise authorization simply because these controls are visible. The secure behavior in a production hospital would require server-enforced permissions and authenticated identity, consistent with HHS access-control guidance (Table 4).

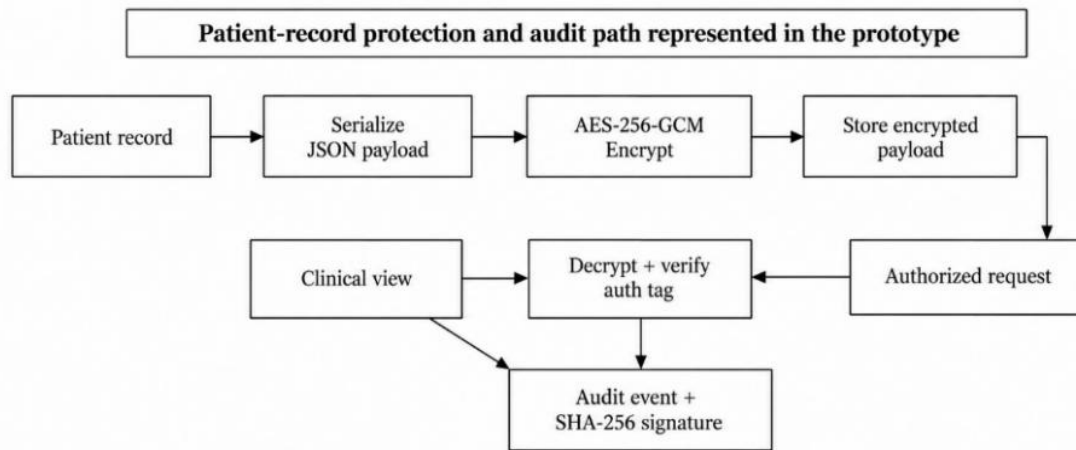
Table 4: Role-based access and functional permissions

Function	Clinician	Nurse/ward	Billing	Compliance/admin
Patient record view	Clinical context	Controlled context	Coverage context	Audit review
EHR decryption	Clinical action	Controlled action	Not primary	Audit visibility
Bed admission	May initiate	Primary execution	View	Review
Bed discharge	Clinical decision	Ward execution	View	Review
Claim processing	View context	Not primary	Primary	Audit
Break-glass	Emergency access	Escalated clinical access	No clinical bypass	Mandatory review
Audit ledger	Review own actions	Review own actions	Review billing events	System review

The audit function stores actor ID, actor name, role, action type, patient or resource identifiers, clinical justification, details, severity and an IP-address field before generating a signature. This gives the prototype a richer accountability model than a basic activity feed.

3.5. EHR Encryption and Patient Record Pipeline

The encryption helper hashes the configured EHR secret to produce a 256-bit key, creates a random 12-byte IV, encrypts the JSON representation of the medical object using aes-256-gcm, collects the authentication tag, and returns an envelope containing IV, authentication tag, ciphertext, algorithm, and key metadata.



The repository implements the encryption and audit primitives; production key management and durable logging remain future requirements.

Figure 2: Patient-record protection and audit path represented in the prototype. Source: author-derived from the implemented encryption and audit functions, with the GCM concept described by NIST Blobel [2] and Dworkin [4]

During decryption, the same algorithm and key are used; the tag is attached to the ciphertext, and the plaintext is parsed as JSON. The approach corresponds to the authenticated-encryption role of GCM described by NIST Dworkin [4] (Table 5).

Table 5: Encryption and audit implementation details

Stage	Implementation Detail	Reason
Serialize	JSON.stringify medical object	Stable payload boundary
Nonce	12 random bytes	Unique IV for each encryption event
Encrypt	AES-256-GCM	Confidentiality + authenticated encryption
Envelope	IV + tag + ciphertext + metadata	Carries decryption material
Decrypt	setAuthTag + decipher.final	Authenticates the ciphertext
Audit	SHA-256-derived event signature	Records security-sensitive action

3.6. Patient Record Data Architecture

The paper represents patient information as a structured object with demographic, admission, insurance, emergency-contact, and medical-history fields (Table 6).

Table 6: Patient data families and their functional roles

Data Family	Representative Fields	Role
Identity	Patient ID, MRN, name, date of birth, gender, blood type	Identification
Admission	Admission status, bed ID, primary physician	Episode context
Coverage	Provider, policy number, eligibility, copay	Financial context
Emergency contact	Name, relationship, phone	Escalation
Clinical history	Allergies, diagnoses, notes	Protected EHR content
Event trace	Audit event IDs and signatures	Accountability

Medical history itself contains allergies, diagnoses and clinical notes in the demonstration data and is protected by the encryption envelope. The patient endpoint returns records with the encrypted history intact rather than automatically returning decrypted clinical content.

3.7. Ward Census, Bed Telemetry and State Transitions

A bed is admitted only when the source bed is available or reserved. Admission assigns the patient, doctor, nurse, timestamp and notes and initializes a telemetry object. Discharge removes the patient and telemetry and changes the bed to cleaning. Sanitization returns it to available. Transfer validates an occupied source and available destination before moving the patient and recording the event (Table 7).

Table 7: Bed management state transitions and system effects

Current State	Action	Next State	System Effect
Available	Admit	Occupied	Patient and care-team linkage
Reserved	Admit	Occupied	Patient and care-team linkage
Occupied	Discharge	Cleaning	Patient detached; telemetry cleared
Cleaning	Sanitize	Available	Bed reset after sanitization
Occupied	Transfer	Occupied at destination	Location changed; audit event

The supplied ward screenshot shows these states directly: an available bed, an occupied ICU bed with telemetry, a sanitizing bed and emergency bays. This visual arrangement is used to provide implementation evidence.

3.8. Appointment and Triage Workflow

Appointment objects in the Source contain a patient reference, physician, specialty, date and time, duration, urgency, reason for visit, status, room, triage priority score and scheduling method (Table 8).

Table 8: Appointment and triage attributes and workflow context

Attribute	Example values	Purpose
Urgency	Routine / Urgent / Emergency	Clinical priority context
Triage priority	4 – 9 in demonstration records	Ordering support
Scheduling method	Manual / AI Automated Triage Engine	Traceability
Status	Scheduled / In-Consultation / Completed	Encounter lifecycle
Room	Clinic / ICU / OR	Resource allocation
Duration	30 – 60 minutes	Capacity planning

Example records include routine cardiology follow-up, an urgent pulmonary review and an emergency orthopedic encounter. The triage score is represented as an integer and can be displayed to support scheduling decisions.

3.9. Financial Clearance and EDI-Oriented Claim Model

The claims model separates clinical coding from financial posting. A record can contain patient and payer identity, service date, provider, diagnosis codes, procedure lines, billed amounts, approved amounts, copay, patient balance, claim status, batch 7 identifier, payer transaction ID and remittance explanation. The screen the author provides shows these fields in a single claim view (Table 9).

Table 9: Claim data groups and operational use

Claim Group	Fields	Operational Use
Identity	Claim number, patient ID, MRN	Patient linkage
Coverage	Payer, policy number	Eligibility/payer context
Clinical coding	ICD-10 diagnosis, CPT procedure	Service representation
Financial	Billed, approved, copay, patient due	Reconciliation
EDI metadata	837P batch ID, payer transaction	Transaction trace
Remittance	EOB explanation, adjudication time	Payment context
Lifecycle	Submitted / In-Adjudication / Approved	Workflow state

CMS identifies X12 837-family transactions for health-care claims and X12 835 for electronic remittance advice. The paper uses these labels as a structured demonstration. It does not claim to have established a live clearinghouse connection.

3.10. Audit Event Model and Break-Glass Access

audit ledger includes EHR decryption, bed admission, claim adjudication and break-glass access examples. A critical break-glass record contains a clinical justification describing the emergency need for immediate history and allergy access (Table 10).

Table 10: Audit events, targets, severity, and captured evidence

Event	Target	Severity	Evidence captured
EHR_DECRYPT	Patient	Info	Actor, reason, record identifier, signature
BED_ADMIT	Bed	Info	Patient, staff, destination bed
CLAIM_ADJUDICATION	Claim	Info	Payer transaction and financial details
BREAK_GLASS_OVERRIDE	Patient	Critical	Emergency reason and compliance entry

This pattern links emergency access to later review, consistent with the accountability intent of audit controls.

3.11. Data Collection and Preprocessing

The evaluation uses two evidence classes: implementation evidence and interface evidence. Implementation evidence comes from the source repository, especially server.ts and package.json. Interface evidence comes from the four screenshots the author provided. Since the paper does not provide a clinical dataset, preprocessing is limited to normal application handling: JSON parsing, object selection by identifiers, aggregate counts, serialization of medical history before encryption, and generation of audit signatures. The screenshots contain demonstrative names, identifiers, claims and telemetry values. They illustrate workflow structure and should not be interpreted as patient-study data. No statistical claim about clinical accuracy is derived from them.

3.12. Verification Methodology

Verification focused on behaviors visible in the Source: route and data-shape inspection, state-transition checks, encryption-envelope checks, audit-event creation, financial lifecycle inspection, and interface-to-source traceability (Table 11).

Table 11: Verification layers, representative tests, and expected results

Verification layer	Representative test	Expected result
Patient record	Inspect patient endpoint and encrypted payload	Protected record returned without automatic plaintext history
Encryption	Generate and decrypt a medical object	Authenticated plaintext recovered from valid envelope
Bed admission	Admit patient to available bed	Status becomes occupied, and BED_ADMIT is logged
Bed discharge	Discharge occupied bed	Status becomes cleaning, and BED_DISCHARGE is logged
Sanitization	Sanitize the cleaning bed	Status returns to available
Transfer	Move from occupied to available bed	Source/destination update and BED_TRANSFER logged
Claims	Inspect three example claim states	Submitted / In-Adjudication / Approved represented
Audit	Trigger sensitive operation	Event includes actor, reason, severity and signature

4. Results

The results describe what the repository and screenshots establish directly. The paper does not introduce user-study or clinical-performance Figures 1 and 2 where it does not provide them. This approach is deliberately conservative: the paper reports implementation behavior rather than turning UI labels into claims of certification or medical effectiveness.

4.1. Workflow Performance Baseline vs Current Prototype

The supplied reference paper uses timing comparisons between software versions. This paper does not document a pair of measured releases, so the same subsection is adapted to a workflow comparison. The baseline represents fragmented handling of the same tasks across separate tools; the current column represents the integrated prototype (Table 12).

Table 12: Workflow consolidation and integrated prototype comparison

Workflow Task	Fragmented Baseline	Current Prototype	Observed Property
Find patient	Separate patient/EHR screens	Patient list + encrypted EHR	Unified identity
Review history	Separate clinical record	Cryptographic EHR vault	Protected record action
Check ward	Manual capacity list	Ward census + telemetry	Explicit state
Review appointment	Separate calendar	Urgency + triage score + room	Context-rich schedule
Review claim	Billing screen	Claim + coding + remittance view	Patient-linked financial context
Review access	Separate audit system	Clinical audit ledger	Traceable event

The result is workflow consolidation. It is not a measured reduction in seconds per task. A controlled human-factors study would be required to establish whether the integrated view actually improves task time or reduces error.

4.2. Baseline vs Current Feature and Architecture Comparison

Comparison between baseline concepts and current implementation: Patient record, Bed management, Telemetry Scheduling, Claims Auditing (Table 13).

Table 13: Baseline concepts and current implementation features

Aspect	Baseline concept	Current implementation
Patient record	Plain record navigation	Encrypted EHR payload + controlled decrypt action
Bed management	Available / occupied list	Available / occupied / cleaning + transfer
Telemetry	Narrative note	Structured vital fields
Scheduling	Calendar	Urgency + triage priority + room + status
Claims	Manual billing entry	Structured claim + adjudication state
Audit	Generic log	Typed event + signature
Emergency access	Informal exception	Break-glass event + clinical justification
Interoperability	Application-specific	X12-oriented fields; future FHIR mapping
Deployment	Static prototype	Express server + browser client

It shows enhanced security, structured data, emergency access, interoperability, and deployment using modern technologies.

4.3. Security and Data-Handling Verification Results

The security behavior traces directly to the implementation. AES-256-GCM is declared as the EHR algorithm (Table 14).

Table 14: Security verification checks, implementation evidence, and status

Security check	Evidence in Source	Status
AES-GCM declaration	aes-256-gcm algorithm constant	Implemented
Random IV	crypto.randomBytes(12)	Implemented
Auth tag	getAuthTag() stored in envelope	Implemented
Authenticated decryption	setAuthTag(authTag)	Implemented
Audit identity	Actor and role recorded	Implemented
Clinical reason	clinicalJustification field	Implemented
Event signature	SHA-256-derived hashSignature	Implemented
Durable logging	Audit array in memory	Not implemented
Enterprise identity	No external identity provider shown	Not implemented
Compliance certification	No independent certification evidence	Not established

The implementation generates a 12-byte IV for encryption, stores the authentication tag in the returned envelope, and supplies it to the decipher during decryption. Audit signatures are produced using SHA-256 over event data combined with a timestamp value.

4.4. Comparative Evaluation Against Existing Methods

The comparison focuses on capabilities rather than competitive scoring. The purpose is to show what the prototype integrates and what still depends on specialized infrastructure (Table 15).

Table 15: Smart hospital capabilities across workflow and implementation models

Capability	Manual/Isolated Workflow	Specialized HIS / EHR	Smart Hospital Prototype
Patient context	Paper or separate files	Integrated clinical record	Structured patient + encrypted history
Ward capacity	Manual communication	Dedicated bed module	Bed state + telemetry
Scheduling	Calendar or paper	Integrated scheduler	Urgency + triage + room
Claims	Billing software	Revenue-cycle module	Structured claim + remittance
Audit	Manual log	Enterprise audit	Event fields + signature
Emergency override	Policy + manual review	Governed workflow	Break-glass event
Interoperability	Varies	Standards dependent	X12-oriented model; no live gateway
Operational validation	None in prototype	Depends on deployment	Not established

4.5. Performance Analysis Load and Resource Characteristics

The repository does not contain a measured browser-performance study. Its implementation nevertheless reveals predictable computational behavior. Main patient, bed, appointment, claim and audit collections are JavaScript arrays, and aggregate statistics scan these collections. Encryption and decryption scale with the size of the clinical object rather than with the number of records (Table 16).

Table 16: Operations, implementation patterns, and approximate computational complexity

Operation	Implementation Pattern	Approximate Complexity
Patient selection	Array selection/lookup	$O(n)$
Bed census	Array filtering	$O(n)$
Aggregate statistics	Multiple collection scans	$O(n)$ per collection
Audit insertion	Unshift into event array	$O(n)$
HER encryption	AES-GCM over payload	$O(m)$ in payload size
HER decryption	GCM verification + JSON parse	$O(m)$
Claim display	List selection/mapping	$O(n)$

For larger deployments, the main scaling requirement is data architecture rather than a different UI framework. Indexed fields, pagination, transactions, concurrency controls, durable audit storage and asynchronous telemetry ingestion would become important.

4.6. User Interface Screenshots

The following Figures 3 and 4 are direct screenshots supplied by the author. They are used as implementation evidence and are not presented as clinical-study evidence.

4.6.1. Cryptographic HER Vault

Figure 3 shows a patient list on the left and a selected patient panel on the right. The image shows a secure digital healthcare system, named Kaze Hospital Cryptographic EHR Vault, which aims to preserve patients' electronic health records. It shows a hospital dashboard with ward census, telemetry, triage, scheduling, and claims management. The system employs AES-256-GCM encryption to secure sensitive medical information. It presents the patient's information, such as demographics, admission status, blood group, physician, insurance and emergency contact, systematically. The interface displays a list of patients with their medical record numbers and current status. Clinical notes, diagnoses, prescriptions, laboratory findings and allergy information are cryptographically sealed into a part of the medical record that indicates they are securely protected. The encrypted record contains metadata for authentication and encryption. Users must authenticate and decrypt the medical history to access protected information. The system also has privacy, security and audit capabilities that provide regulated access to medical records.

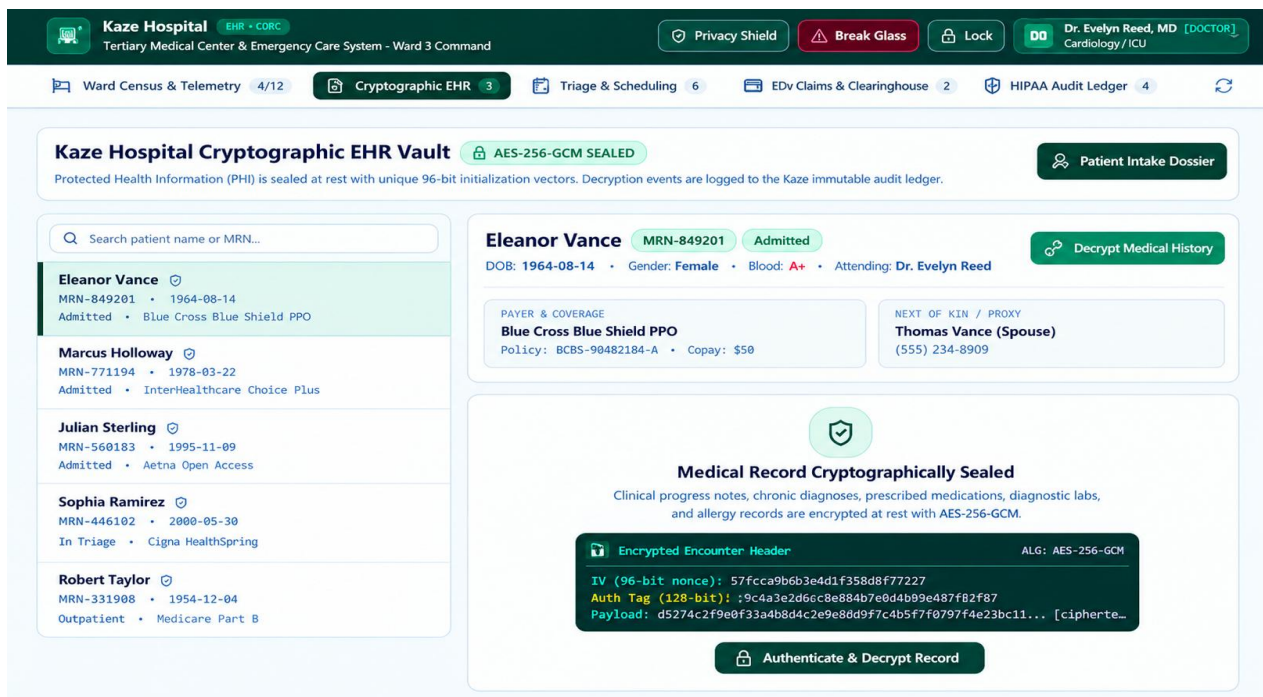


Figure 3: Sample Cryptographic HER vault showing patient selection, coverage context, encrypted medical-history envelope, and controlled decryption

The selected record exposes demographic and coverage context, followed by a cryptographically sealed medical-record area. The encrypted envelope displays an algorithm label, IV, authentication tag and ciphertext payload, with an explicit action to authenticate and decrypt the record. These visual elements correspond to the server-side envelope fields.

4.6.2. HIPAA Audit Trail and Cryptographic Ledger

Figure 4 shows an audit screen organized around timestamp, operator role, event type, patient or resource, justification and signature. The four visible examples include record decryption, bed admission, claim adjudication and a critical break-glass event. The screen corresponds to the source audit object and helper function.

Kaze Hospital HIPAA Audit Trail & Cryptographic Ledger NIST SP 800-66 VERIFIED					
Immutable SHA-256 hash-chained event logs: recording every medical chart decryption, bed transfer, emergency break-glass, and billing submission.					
Search operator, patient, action, or SHA-256 hash... All Roles All Actions					
Timestamp (UTC)	Operator / Role	Action Event	Target Patient / Resource	Justification & Details	SHA-256 Signature
2025-09-17 08:38:15	Dr. Evelyn Reed DOCTOR	EHR_DECRYPT	Eleanor Vance MRN-849201	Rationale: Daily morning ICU rounds and telemetry review Decrypted AES-256 medical record for clinical assessment...	a79f28bc4910...
2025-09-17 10:50:22	Nurse James Miller NURSE	BED_ADMIT	Julian Sterling MRN-560183	Rationale: Admitted from trauma department for emergency surg... Assigned bed BED-SURG-201 on Floor 2 Surgical Ward.	b8193ce0f112...
2025-09-17 11:45:00	Sarah Jenkins BILLING	CLAIM_ADJUDICATION	Eleanor Vance MRN-849201	Rationale: Real-time 835 EDI Remittance Advice received from BL... Claim approved for \$5,530.00. Coinsurance calculated at 5%...	c90184da3829...
2025-09-17 19:30:10	Dr. Kenneth Thorne DOCTOR	BREAK_GLASS_OVERRIDE	Sophia Ramirez MRN-446102	Rationale: EMERGENCY: Acute abdomen with peritoneal signs, ne... Break-glass protocol invoked. Clinical bypass logged to co...	d07124ef8192...

Figure 4: Sample Hospital audit ledger showing actor role, action event, patient/resource target, justification and SHA-256-derived signature

4.6.3. EDI Claims and Clearinghouse View

Figure 5 places an electronic claim, itemized procedures, payer information, billed charges, approved amount, patient responsibility and remittance advice in one panel.

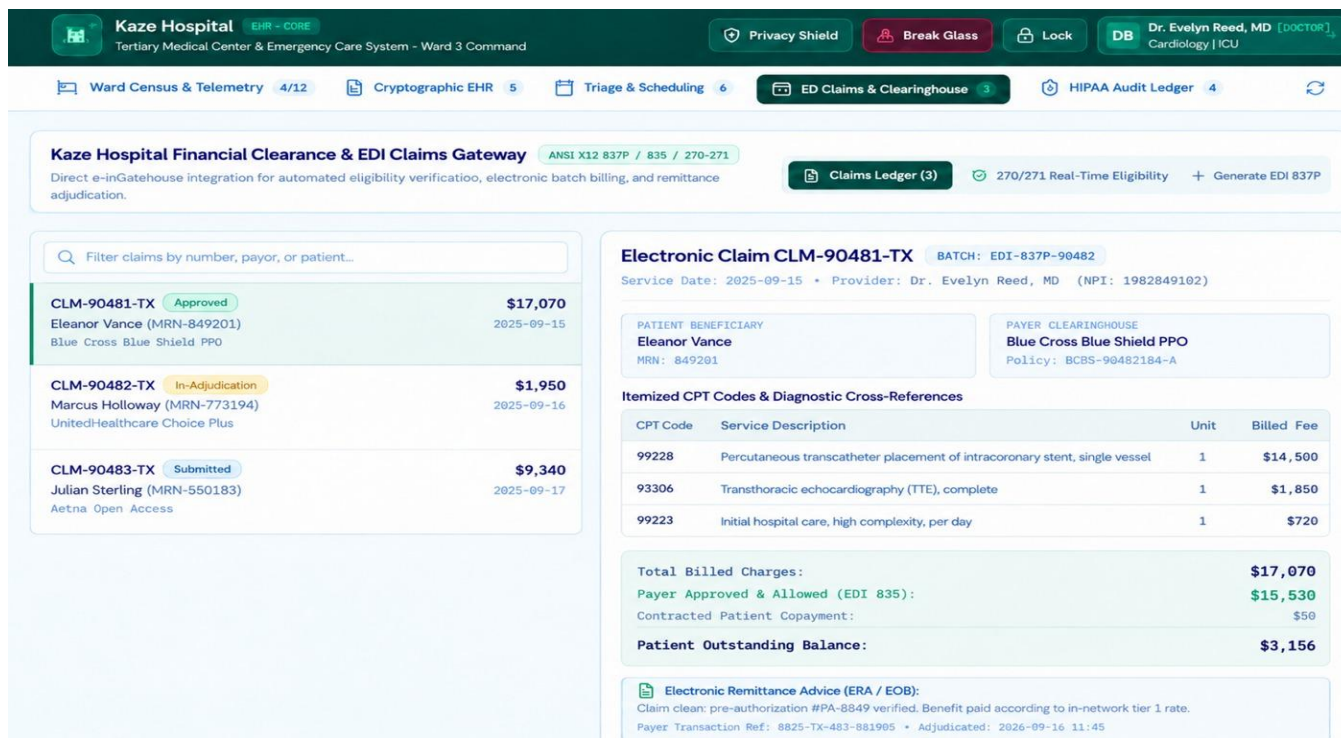


Figure 5: Sample EDI-oriented claims gateway displaying claim identifiers, diagnosis/procedure coding, billed and approved amounts, and electronic remittance advice

The UI uses 837P/835 and related transaction vocabulary. The Source provides structured data fields for these concepts, while the current implementation remains a local demonstration rather than a live payer-clearinghouse connection.

4.6.4. Ward Census and Bed Telemetry

Figure 6 presents the ward census. The screen separates available, occupied and sanitizing beds and places telemetry inside an occupied ICU bed. Bed actions are visible at the card level, matching the server operations for admission, discharge, sanitization and transfer.

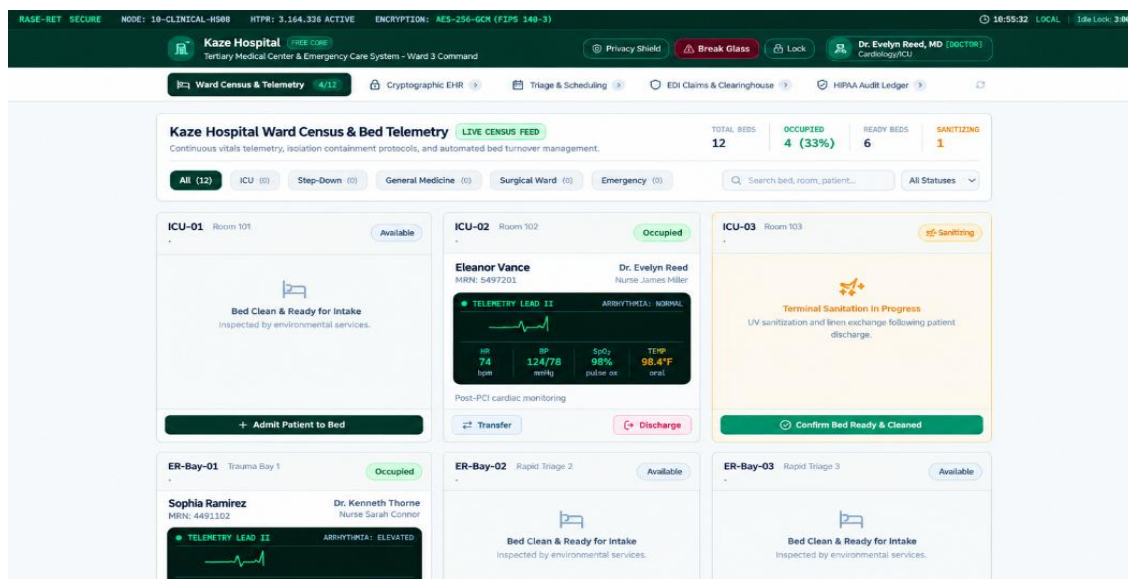


Figure 6: Sample Ward census and bed telemetry dashboard showing capacity state, bedside vital indicators, sanitization status and operational actions

4.6.5. Interface-to-Source Traceability

Table 17 presents the implementation components for each interface feature, covering EHR encryption, auditing, insurance claims and bed management.

Table 17: Interface evidence and corresponding implementation components

Figures	Interface Evidence	Corresponding Implementation
Figure 2	Encrypted EHR envelope and decrypt action	encryptData / decryptData + patient records
Figure 3	Audit rows, event types, justification and signature	recordAudit / auditLogs / generateAuditHash
Figure 4	Claims, coding, amounts and remittance	insuranceClaims + ICD-10 / CPT catalogs
Figure 5	Bed status and telemetry	beds + admit / discharge / sanitize / transfer

This illustrates how the user-interface functionalities of the system are supported by the corresponding backend operations and data-handling components.

4.7. Discussion and Findings

4.7.1. Key Findings

The first finding is that the patient can serve as the shared reference across several hospital workflows. The same identifier links the protected EHR, bed assignment, appointment and claim examples. This is a practical response to the fragmentation problem described in the EHR and HIS literature. The second finding is that the prototype makes sensitive actions visible. A decryption event carries actor information and a clinical justification. Emergency access is represented as a separate event type rather than being hidden inside a general activity log. This structure supports later compliance review even though the current implementation does not provide a durable enterprise ledger. The third finding is that bed status is modeled as a process rather than a single boolean. The cleaning state creates an operational transition between discharge and the next admission. That is a small design detail, but it makes the model closer to how physical resources are actually managed. The fourth finding concerns financial workflow. Claims are not treated as isolated invoices. They retain patient, provider, coding, payer, amount and remittance information in one structured object. This makes it easier to reason about the lifecycle from submitted to adjudicated status and to associate financial activity with the correct patient record. The fifth finding is that the project maintains a clear boundary between interface language and system assurance. The UI uses healthcare-standard terminology, but the Source does not establish live payer integration, FHIR exchange, enterprise identity, or a clinical study. Stating these limitations directly makes the paper more reproducible and avoids inflating a prototype into a production claim.

4.7.2. Security and Privacy Considerations

The paper contains meaningful security primitives, but it should not be described as HIPAA-certified. AES-256-GCM provides authenticated encryption for the EHR payload, while the audit helper adds contextual trace data (Table 18).

Table 18: Current prototype security aspects and production requirements

Security Aspect	Current Prototype	Production Requirement
EHR protection	AES-256-GCM	Managed key lifecycle + tested recovery
Authentication	Application-level identity	Enterprise identity + MFA
Authorization	Application workflow	Server-enforced permission scopes
Audit	In-memory array + signature	Durable append-only logging
Key management	Environment-derived key	Secret manager / HSM or equivalent
Data persistence	Process memory	Transactional database + backup
Transport	Not established by code alone	TLS + secure deployment
Break-glass	Critical event + reason	Governed emergency-access process
Interoperability	X12-oriented labels	Validated 837/835 gateway + clinical standard exchange
Compliance	Design intent only	Independent legal/security assessment

However, the master key is derived from an application secret with a development fallback, audit records are stored in memory, and the repository does not show an enterprise identity provider or a full authorization middleware layer.

4.7.3. Future Roadmap

Table 19 presents the future roadmap with planned features, priorities and expected impact on system security, interoperability, monitoring and reliability.

Table 19: Future roadmap, planned features, priorities, and expected impact

Planned feature	Priority	Expected impact
Persistent clinical database	Critical	Durable patient, bed, appointment and audit history
Secure identity and MFA	Critical	Stronger authentication and access control
Managed encryption keys	Critical	Production-grade EHR protection
FHIR mapping and API	High	Interoperable clinical exchange
Validated 837/835 gateway	High	Actual payer transaction support
Real telemetry ingestion	High	Continuous bedside monitoring
Usability study	High	Measured task time, errors and satisfaction
Immutable audit storage	Critical	Long-term accountability and forensics
Governed break-glass workflow	Critical	Emergency access with post-event review
Operational analytics	Medium	Capacity and service planning
Offline / resilience layer	Medium	Continuity during network outages

It aims to improve clinical data management, authentication, auditability, emergency access, analytics and service continuity.

4.7.4. Operator Compliance Journey and Alert Analysis

The reference journal uses a patient-compliance subsection. For a hospital management platform, the closest equivalent is the operator-compliance journey: detection of a sensitive action, policy decision, execution, audit capture, and follow-up (Table 20).

Table 20: Operator compliance journey and alert analysis

Stage	Example	Evidence	Follow-up
Detection	Protected record requested	Action type	Confirm user and context
Decision	Routine or emergency	Severity/justification	Proceed under policy
Action	Decrypt or admit	Business object updated	Task completed
Trace	Audit helper executes	Timestamp + signature	Reviewable record
Follow-up	Break-glass event	Critical severity	Compliance review

In the present paper, the event path is visible for EHR decryption, bed admission, claim adjudication and break-glass access.

4.7.5. Browser and Deployment Considerations

The application is intentionally straightforward to run. The package manifest exposes development, build, start, preview, clean and type-checking commands. The browser-based client keeps the interface accessible during development, while the Express server provides a single integration point for the operational model. A production hospital installation would require more than a web server. Durable storage, network segmentation, TLS, secret management, backup, monitoring, access reviews, vulnerability management, incident response, and disaster recovery would need to be added around the application. The current paper should be seen as the software layer of a larger architecture, not the complete infrastructure.

4.7.6. Patient Record Field Schema

The patient record schema is defined here, with identification, admission, bed, physician, insurance and emergency contact information (Table 21).

Table 21: Patient record field schema

Field	Type/form	Description
Patient ID	String	Unique patient reference
MRN	String	Medical record number

Admission status	Enumeration	Episode status
Bed ID	Optional string	Current ward location
Primary physician	String	Assigned clinician
Insurance	Object	Coverage and eligibility context
Emergency contact	Object	Escalation contact
Medical history	Encrypted object	Allergies, diagnoses, notes and medications
Audit linkage	Event records	Sensitive actions and operational changes

Encrypted medical history and audit trails are also part of it to ensure secure data management and sensitive change tracking.

4.7.7. Bed Object - Field Schema

The server's statistics operation computes total, occupied, available and cleaning bed counts and derives occupancy and ICU occupancy percentages from the current arrays (Table 22).

Table 22: Bed object field schema

Field	Purpose
id / bedNumber	Stable bed identity
department / roomNumber / floor	Physical location
status	Available, reserved, occupied or cleaning
patientId / patientName / patientMrn	Current patient linkage
assignedDoctor / assignedNurse	Care-team context
admittedAt	Episode timestamp
notes	Operational instructions
telemetry	Heart rate, BP, oxygen saturation, temperature, respiration

It also counts appointments, pending claims, EHR decryptions and break-glass events, linking operational and security state in one summary response.

4.7.8. Claim Record Field Schema

Table 23 defines the schema of a claim object, including patient, payer, services, diagnoses, procedures, and financial values. It also defines claim status, EDI metadata and remittance information to ensure accurate billing and adjudication.

Table 23: Claim object field schema

Field	Description
Claim number	Financial transaction identifier
Patient ID / MRN	Patient linkage
Payer/policy	Coverage linkage
Date of service	Encounter date
Rendering provider	Provider identity
Diagnoses	ICD-10 code and description
Procedures	CPT code, description, units and fee
Billed/approved / copay/patient due	Financial values
Status	Submitted, In-Adjudication, or Approved
EDI metadata	Batch and payer transaction references
Remittance explanation	Adjudication context

5. Conclusion

The Smart Hospital Management and Patient Record System demonstrates a compact approach to linking the clinical, operational, financial and security sides of hospital work. Instead of presenting the patient record, ward census, appointment schedule, claim ledger and audit history as unrelated pages, the prototype gives each workflow a consistent data reference and a visible action path. The encryption workflow is the strongest technical security element in the current implementation. AES-256-GCM uses a fresh initialization vector and authentication tag, while the audit helper records contextual information for security-sensitive operations. The ward lifecycle is equally explicit: admission, discharge, sanitization and transfer are separate

server actions with observable state changes. The claims module provides a structured representation of diagnosis and procedure coding, billing values, payer context and remittance information in a form that can later be mapped to standardized exchange workflows. The paper also demonstrates the importance of stating what a system does not yet do. Current data are held in process memory, authentication is not an enterprise identity service, telemetry is demonstrative, and interoperability remains at the data-model level. These constraints prevent claims of production compliance or clinical effectiveness, but they also provide a clear development roadmap. Persistent storage, managed keys, FHIR-based exchange, validated EDI connectivity, real telemetry, formal usability evaluation and stronger authorization can be added without discarding the basic application structure. From a research perspective, the most useful outcome is the reproducible link between implementation and interface. A reviewer can inspect the server code, examine the security envelope, follow the bed-state logic, and compare those elements with the screenshots included in this paper. This traceability makes the prototype a suitable foundation for further work in smart-hospital software engineering and health-information-system design.

5.1. Limitations

- First, the application stores its main data collections in server memory. The current architecture therefore does not provide durable persistence or transactional recovery.
- Second, encryption is implemented as a technical mechanism, not as a complete healthcare-security program. Production deployment would require managed keys, identity and authorization controls, transport security, audit retention, backups, monitoring and formal risk assessment.
- Third, the telemetry shown in the interface is demonstration data. No real bedside device is connected to the submitted screenshots, so clinical monitoring accuracy cannot be evaluated.
- Fourth, the paper does not report a controlled usability study or a patient-outcome evaluation. Claims about time savings, error reduction or safety improvement remain future research questions.
- Fifth, the claim module uses X12-oriented identifiers and transaction vocabulary but does not establish a live payer clearinghouse connection. The paper also does not implement an HL7 FHIR exchange service.
- Finally, the names and identifiers visible in the supplied screenshots are for demonstration purposes. Any future study using identifiable patient information would require appropriate ethics, privacy, legal and institutional controls.

Acknowledgment: N/A

Data Availability Statement: The data for this study are available upon request from the corresponding author.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding.

Conflicts of Interest Statement: The authors have no conflicts of interest to declare. This work represents a new contribution by the authors, and all citations and references are appropriately included based on the information utilized.

Ethics and Consent Statement: This research adheres to ethical guidelines and obtains informed consent from all participants. Confidentiality measures were implemented to safeguard participant privacy.

References

1. J. Sam Sundar, "Smart Hospital Management and Patient Record System," *GitHub repository*, Ryu-kaze, 2026. [Accessed by 05/04/2026].
2. B. Blobel, "Authorisation and access control for electronic health record systems," *International Journal of Medical Informatics*, vol. 73, no. 3, pp. 251–257, 2004.
3. U.S. Department of Health and Human Services, "Summary of the HIPAA Security Rule," *U.S. Department of Health and Human Services*, 2026. [Accessed by 07/04/2026].
4. M. Dworkin, "Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC," *National Institute of Standards and Technology*, 2007. [Accessed by 10/06/2025].
5. Centers for Medicare & Medicaid Services, "Health care payment and remittance advice," *Centers for Medicare & Medicaid Services*, 2024. [Accessed by 11/06/2025].
6. H. Khorasani, S. Damiri, S. Abbasi, A. Fathabadi, M. Dehghani, and M. Sadeghi, "Hospital information systems evaluation criteria: A scoping review," *BMC Health Services Research*, vol. 26, no. 1, pp. 1–12, 2026.
7. O. Olakotan, R. Samuriwo, H. Ismaila, and S. Atiku, "Usability challenges in electronic health records: Impact on documentation burden and clinical workflow: A scoping review," *Journal of Evaluation in Clinical Practice*, vol. 31, no. 4, pp. 1–16, 2025.

8. M. Cahill, B. J. Cleary, and S. Cullinan, "The influence of electronic health record design on usability and medication safety: Systematic review," *BMC Health Services Research*, vol. 25, no. 1, pp. 1–32, 2025.
9. HL7 International, "Fast Healthcare Interoperability Resources (FHIR), Release 4," *HL7 International*, 2023. [Accessed by 12/06/2025].
10. World Health Organization, "Global Strategy on Digital Health 2020–2027," *World Health Organization*, 2025. [Accessed by 15/04/2026].
11. World Health Organization, "Global Initiative on Digital Health," *World Health Organization*, 2024. [Accessed by 20/06/2025].
12. Centers for Medicare & Medicaid Services, "Adopted standards and operating rules," *Centers for Medicare & Medicaid Services*, 2020 [Accessed by 21/06/2025].
13. S. Al-Sharhan, E. Omran, and K. Lari, "An integrated holistic model for an eHealth system: A national implementation approach and a new cloud-based security model," *International Journal of Information Management*, vol. 47, no. 8, pp. 121–130, 2019.
14. J. Choe and S. K. Yoo, "Web-based secure access from multiple patient repositories," *International Journal of Medical Informatics*, vol. 77, no. 4, pp. 242–248, 2008.
15. D. Gritzalis and C. Lambrinoudakis, "A security architecture for interconnecting health information systems," *International Journal of Medical Informatics*, vol. 73, no. 3, pp. 305–309, 2004.
16. A. S. Mukherjee, M. Aanestad, and S. Sahay, "Judicious design of electronic health records: Case from public health system in India," *Health Policy and Technology*, vol. 1, no. 1, pp. 22–27, 2012.
17. J. Kryszyn, K. Cywoniuk, W. T. Smolik, D. Wanta, P. Wróblewski, and M. Midura "Performance of an openEHR based hospital information system," *International Journal of Medical Informatics*, vol. 162, no. 6, pp. 1–12, 2022.
18. S. Yoo, S. Y. Jung, S. Kim, E. Kim, K. H. Lee, E. Chung, and H. Hwang, "A personalized mobile patient guide system for a patient-centered smart hospital: Lessons learned from a usability test and satisfaction survey in a tertiary university hospital," *International Journal of Medical Informatics*, vol. 91, no. 7, pp. 20–30, 2016.
19. F. A. Alzahrani, "Estimating Security Risk of Healthcare Web Applications: A Design Perspective," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 187–209, 2020.

Publisher's Note: The publisher remains impartial concerning jurisdictional claims in published maps and institutional affiliations. Responsibility for the content rests entirely with the authors and does not necessarily reflect the publisher's perspectives.