Software Design Document

for

<HealthLink>

Version 1.0

Prepared by: Abhijith D.L. (01), Bala M.S. (29), Megha Shaji (47), Muhammed Aslam(48)

<TKM College of Engineering>

Date: 04/03/2025

Table of Contents

1. Introduction

1.1 Purpose

The Electronic Health Record (EHR) System is a permissioned blockchain-based solution designed to provide secure, auditable storage of patient records. The system leverages **Hyperledger Fabric** for blockchain functionalities, ensuring robust security and transparency. Additionally, **MongoDB** is used to store non-blockchain metadata efficiently.

This document provides an in-depth analysis of the EHR System's design, architecture, components, and interactions. It serves as a reference for developers, system architects, and stakeholders involved in its implementation and deployment.

1.2 Scope

The EHR System aims to enhance the integrity, privacy, and accessibility of electronic health records through blockchain technology. The key functionalities include:

- **Secure Access Control**: Role-based access management ensuring only authorized personnel can interact with patient records.
- **Immutable and Tamper-Proof Storage**: Health records are stored on **Hyperledger Fabric**, preventing unauthorized modifications.
- **Patient-Driven Access Management**: Patients have complete control over their records. Doctors must request access, which patients can approve or deny.
- **Auditability and Transparency**: Every interaction with the system is recorded as a blockchain transaction log, ensuring traceability and accountability.
- **Scalability and Integration**: The system integrates with **MongoDB** to store metadata and additional healthcare data that do not require blockchain-level security.

1.3 Definitions & Acronyms

To facilitate understanding, the following definitions and acronyms are used throughout this document:

- **EHR (Electronic Health Record)** – A digital version of a patient's paper chart, enabling real-time, patient-centered records.
- **Hyperledger Fabric** – A permissioned blockchain platform tailored for enterprise use, ensuring high security and privacy.
- **MSP (Membership Service Provider)** – A component in Hyperledger Fabric responsible for identity management and authentication.
- **MongoDB** – A NoSQL database used for efficient storage of metadata and auxiliary healthcare information.
- **CA (Certificate Authority)** – A trusted entity responsible for issuing and managing digital certificates to authenticate users.
- **Smart Contract (Chaincode)** – A self-executing contract with predefined rules deployed on the blockchain to manage transactions and enforce policies.

1.4 References

1. Hyperledger Fabric Documentation
2. MongoDB Documentation
3. Spring Boot Documentation

---

2. System Overview

2.1 High-Level Architecture

The system is designed as a secure electronic health record (EHR) management solution. It integrates multiple components to ensure efficiency, security, and scalability. The core components include:

i. Frontend

- Developed using **React** to provide a dynamic and interactive user interface.
- Ensures seamless navigation and a responsive experience for healthcare professionals and patients.
- Communicates with the backend via RESTful APIs for data exchange.

ii. Backend

- Built using **Spring Boot**, a robust Java-based framework for developing scalable and maintainable microservices.
- Handles business logic, user authentication, and API requests from the frontend.
- Implements security mechanisms such as JWT authentication.

iii. Blockchain Layer

- **Hyperledger Fabric** is utilized to store cryptographic hashes of EHR data for integrity verification.
- Ensures **immutability** and **tamper-proof** record validation without exposing sensitive patient data.
- Supports permissioned access control, restricting unauthorized modifications.

iv. Database

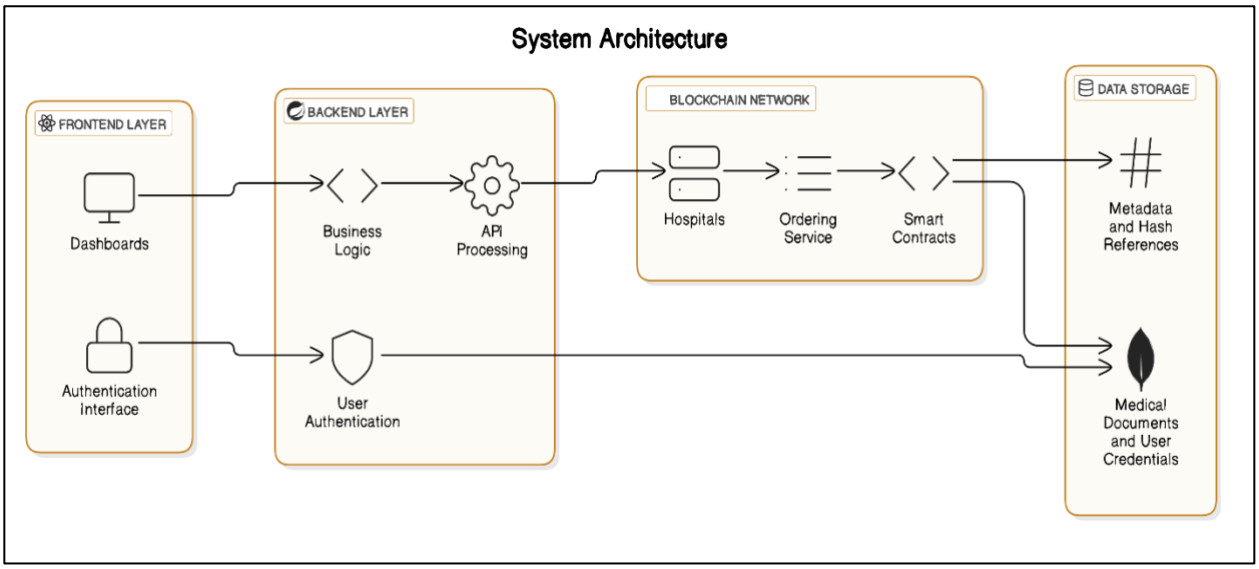
- **MongoDB** is used for storing metadata associated with EHRs, such as timestamps, user access logs, and audit trails.
- Provides high availability and scalability for handling large datasets efficiently.
- Works alongside the blockchain to ensure a hybrid approach to data management where raw data is stored in MongoDB while cryptographic proofs reside on Hyperledger Fabric.

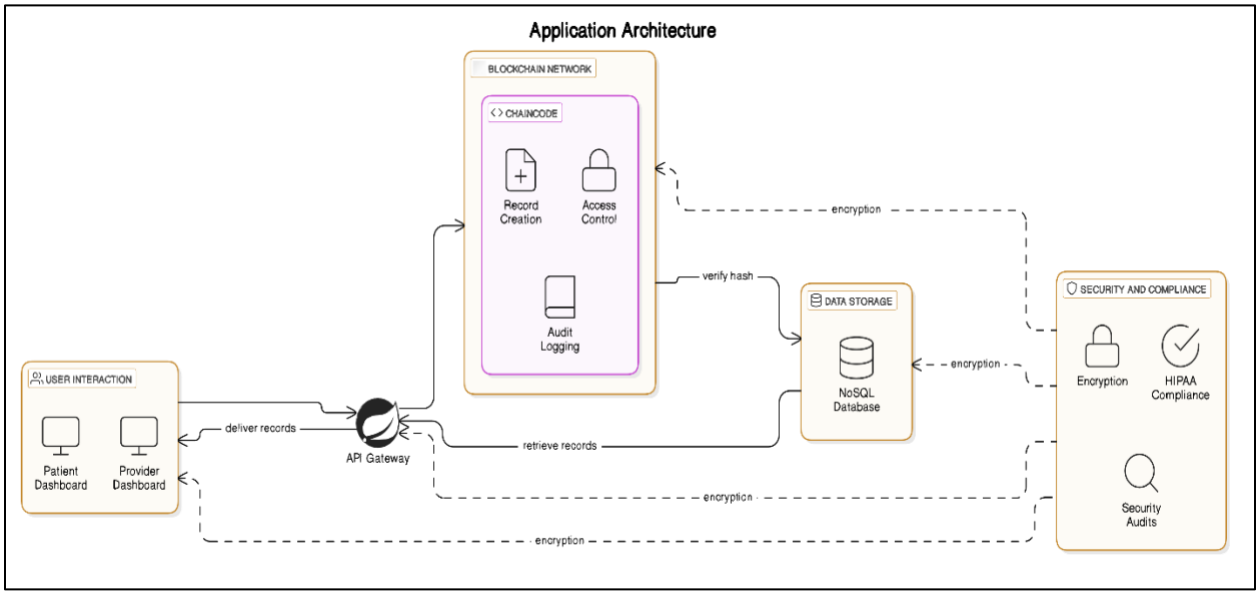
2.2 System Workflow

1. **User Authentication & Access Control**
    - Users (Doctors, Patients, Administrators) log in through the frontend.
    - The backend verifies credentials and grants appropriate access levels.
2. **EHR Data Storage & Hashing**
    - New EHRs are uploaded and stored in MongoDB.
    - A cryptographic hash of the record is generated and stored on Hyperledger Fabric.

- **EHR Retrieval & Validation**
    - When retrieving an EHR, the system verifies its hash against the blockchain to ensure integrity.
    - If the hashes match, the data is deemed **authentic and untampered**.
- **Audit & Compliance**
    - Every transaction (creation, modification, access) is logged.
    - Administrators can review access logs and detect unauthorized activities.

2.3 Architecture Diagram





3. System Components

3.1 Functional Components

Component	Technology	Description
Frontend	React /	User interface for doctors & patients
Backend	Spring Boot	Business logic & API management
Blockchain	Hyperledger Fabric	Stores EHR transaction logs & hashes
Database	MongoDB	Stores metadata & permissions

3.2 Subsystems & Modules

1. User Management Module

The **User Management Module** handles authentication, authorization for doctors, patients, and administrators.

Key Functions:

- **User Authentication:**
    - Uses **Hyperledger Fabric Certificate Authority (CA)** for identity management.
    - Ensures secure login and identity verification.
    - Issues cryptographic certificates for users.
- **Role-Based Access Control (RBAC):**
    - Assigns roles (e.g., Doctor, Patient, Admin) with specific permissions.
    - Doctors can view/edit EHRs based on granted permissions.
    - Patients have full control over their own records.
- **User Lifecycle Management:**
    - Registration, updates, and deactivation of users

2. EHR Management Module

The **EHR Management Module** is responsible for handling the storage and integrity of patient health records.

Key Functions:

- **EHR Data Storage:**
    - Patient records are stored securely in **MongoDB**.
    - Metadata is also maintained.

- **Blockchain Hash Storage:**
    - For each EHR entry, a **cryptographic hash** is generated.
    - The hash is stored on **Hyperledger Fabric**, ensuring integrity and tamper-proof records.
- **Data Retrieval & Verification:**
    - When an EHR is accessed, its current hash is re-generated.
    - The system compares the new hash with the blockchain hash to detect tampering.
    - If mismatched, the system alerts the user and blocks access.

3. Access Control Module

The **Access Control Module** enforces patient-centric access policies and permissions using smart contracts.

Key Functions:

- **Permission-Based Access:**
    - Doctors request access to a patient's record.
    - Patients receive a request notification and can **approve or reject** access.
- **Smart Contract Enforcement:**
    - **Hyperledger Fabric Smart Contracts (Chaincode)** ensure access rules are followed.
    - Only authorized users can view or modify EHRs.
    - Prevents unauthorized tampering or leaks.

4. Audit & Logging Module

The **Audit & Logging Module** provides real-time tracking and ensures transparency in the system.

Key Functions:

- **Transaction Logging:**
    - Every action (record creation, update, access request) is recorded in the **Hyperledger Fabric ledger**.
    - Immutable logs prevent data tampering.
- **Real-Time Monitoring:**
    - Patients can view access logs to see who accessed their data.
    - Alerts are triggered for suspicious activity or failed authentication attempts.

---

4. Data Design

4.1 MongoDB Schema

i. EHR Collection

```
Json
{
  "ehrId": "EHR123",
  "patientId": "PAT456",
  "doctorId": "DOC789",
  "documentHash": "abc123xyz",
  "createdAt": "2025-03-04T12:00:00Z",
  "updatedAt": "2025-03-04T15:00:00Z"
}
```

- **ehrId** (String) → Unique identifier for the EHR document.
- **patientId** (String) → ID of the patient to whom this record belongs.
- **doctorId** (String) → ID of the doctor who created or last updated the record.
- **documentHash** (String) → Cryptographic hash of the EHR document, used for integrity verification. This is stored on the **Hyperledger Fabric** blockchain.

- **createdAt** (Timestamp) → Date and time when the record was created (ISO 8601 format).
- **updatedAt** (Timestamp) → Date and time when the record was last modified.

ii. Access Request Collection

```
json
{
  "requestId": "REQ001",
  "doctorId": "DOC789",
  "patientId": "PAT456",
  "status": "Pending",
  "createdAt": "2025-03-04T12:00:00Z"
}
```

- **requestId** (String) → Unique identifier for the access request.
- **doctorId** (String) → ID of the doctor requesting access.
- **patientId** (String) → ID of the patient whose EHR is being requested.
- **status** (String) → The current status of the request. Possible values:
    - "Pending" → Waiting for patient approval.
    - "Approved" → Patient has granted access.
    - "Rejected" → Patient has denied access.
- **createdAt** (Timestamp) → Date and time when the access request was created.

iii. Patient Collection

Class: Patient

```
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@Document(collection = "patients")
public class Patient {
```

```
    @Id
    private String patientId;
    private String ehrId;
    private EhrDocument ehrDocument;
}
```

Schema (MongoDB Document Structure)

```
{
    "_id": "PAT123",
    "ehrId": "EHR456",
    "ehrDocument": {
        "encrypted ehr": "xyz789abc",
        "createdAt": "2025-03-04T12:00:00Z"
    }
}
```

- **@Document(collection = "patients")** - Maps this class to the **MongoDB patients collection**.
- **@Id private String patientId;** - Unique **identifier** for each patient (**Primary Key** in MongoDB).
- **private String ehrId;** - Links to the **Electronic Health Record (EHR)** for the patient.
- **private EhrDocument ehrDocument;** - encrypted ehr document.

iv. Pending Requests Collection

Class: Pending

```
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@Document(collection = "pending")
public class Pending {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private String requestId;
    private String pid;
    private String did;
    private String status;
}
```

Schema (MongoDB Document Structure)

```
{
  "_id": "REQ001",
  "pid": "PAT456",
  "did": "DOC789",
  "status": "Pending"
}
```

- **@Document(collection = "pending")** - Maps this class to the **MongoDB pending collection**.
- **@Id private String requestId;** - Unique identifier for each **access request**.
- **private String pid;** - Patient ID (**Foreign Key** referencing patients.patientId).
- **private String did;** - Doctor ID (**Foreign Key** referencing a doctor user).
- **private String status;** - Tracks **request status** (Pending, Approved, Rejected).

v. User Collection

Class: UserEntity

```
@Data
@Document(collection = "users")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
```

```
public class UserEntity {
    @Id
    private String id;
    @Column(unique = true, nullable = false)
    private String username;
    @Column(nullable = false)
    private String password;
    @Column(nullable = false)
    private String mspId;
    private String role;
}
```

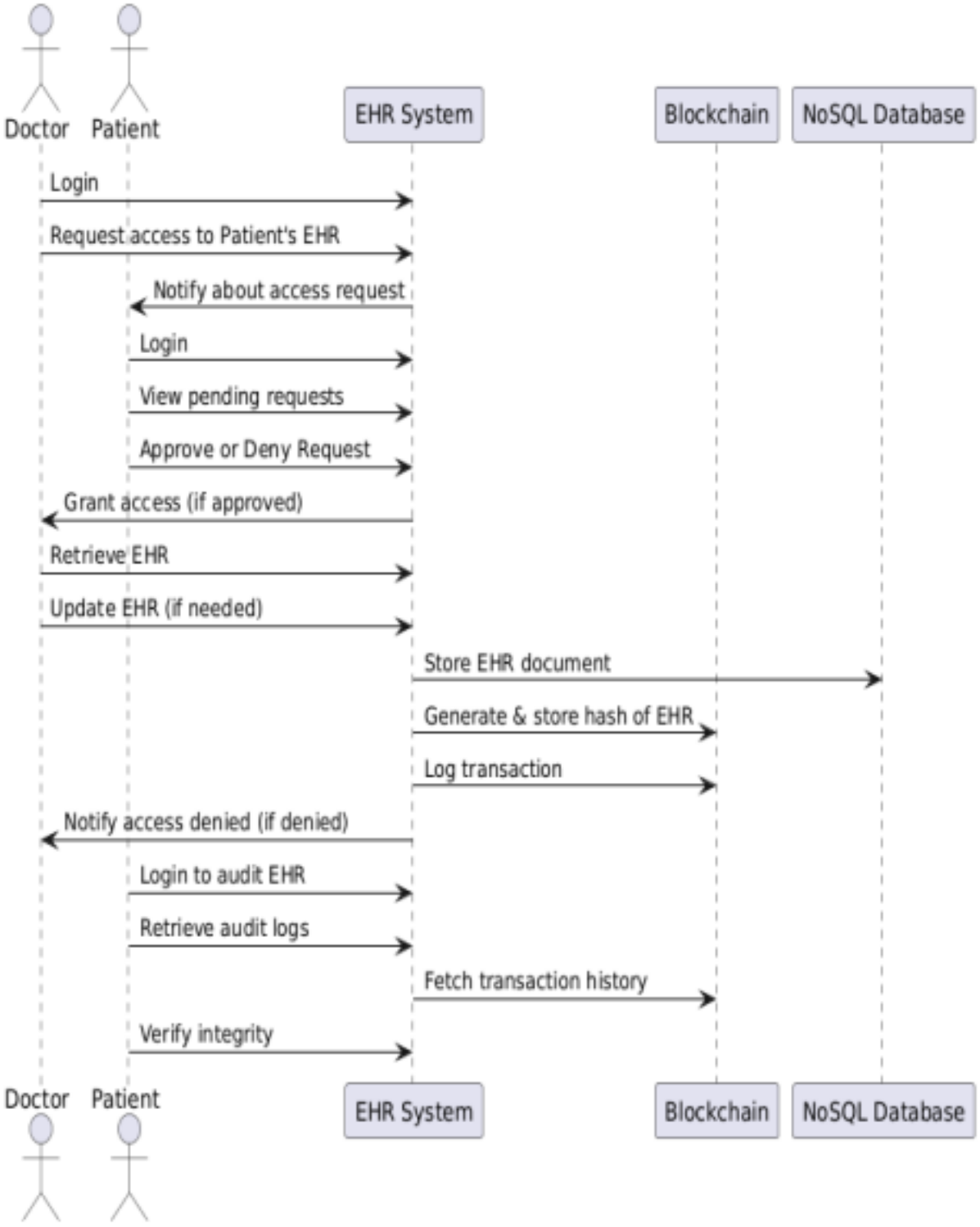
Schema (MongoDB Document Structure)

```
{
    "_id": "USER001",
    "username": "doctor123",
    "password": "$2a$10$hashedpassword",
    "mspId": "Org1MSP",
    "role": "Doctor"
}
```

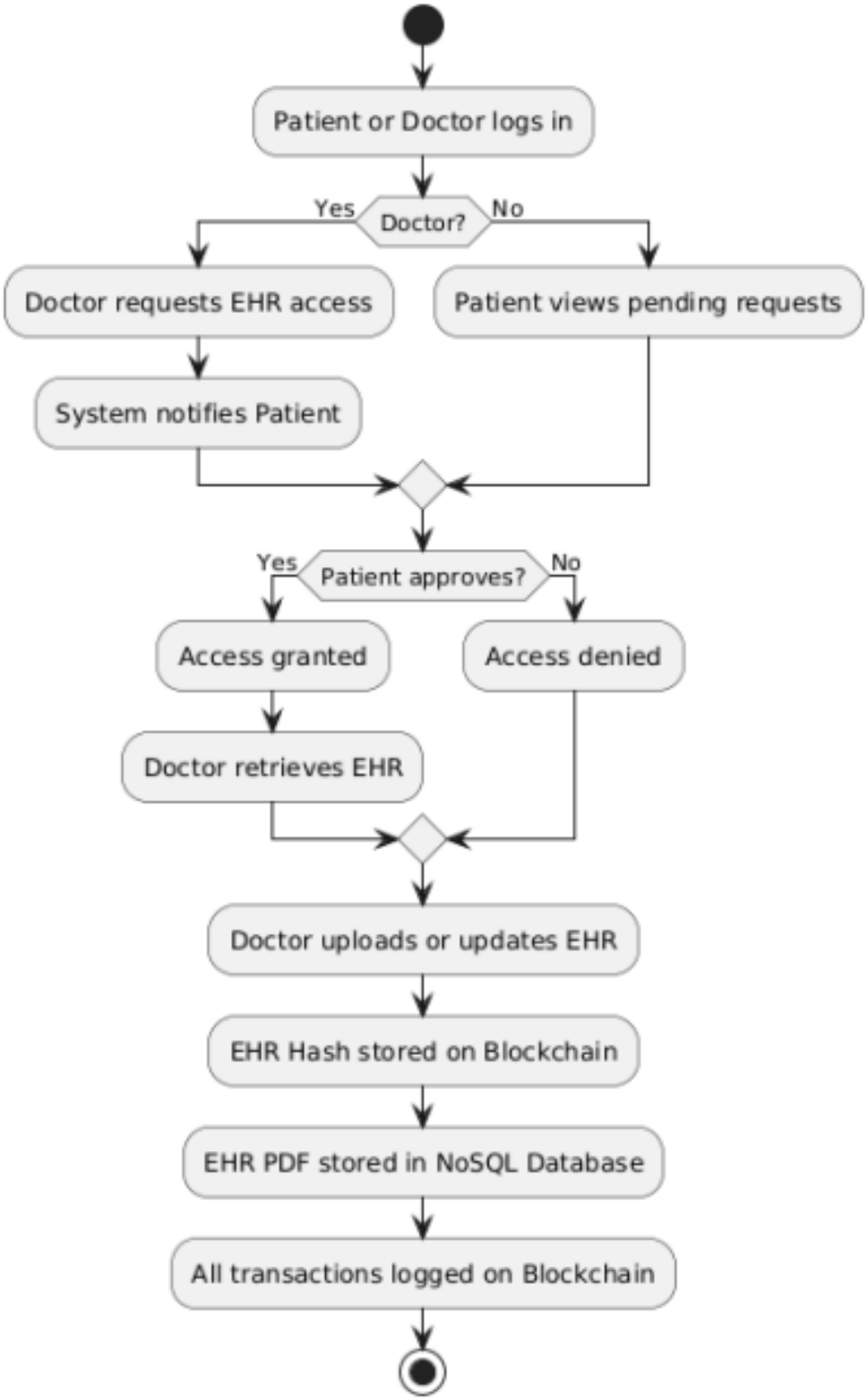
- **@Document(collection = "users")** - Maps this class to the **MongoDB users collection**.
- **@Id private String id;** - Unique identifier for each user (**Primary Key**).
- **@Column(unique = true, nullable = false) private String username;** - Unique username for authentication.
- **@Column(nullable = false) private String password;** - Stores **hashed password**
- **@Column(nullable = false) private String mspId;** - Stores the **Hyperledger Fabric MSP ID**, linking users to specific blockchain organizations.
- **private String role;** - Defines **user roles** (Doctor, Patient, Admin).

5. Workflow & Interactions

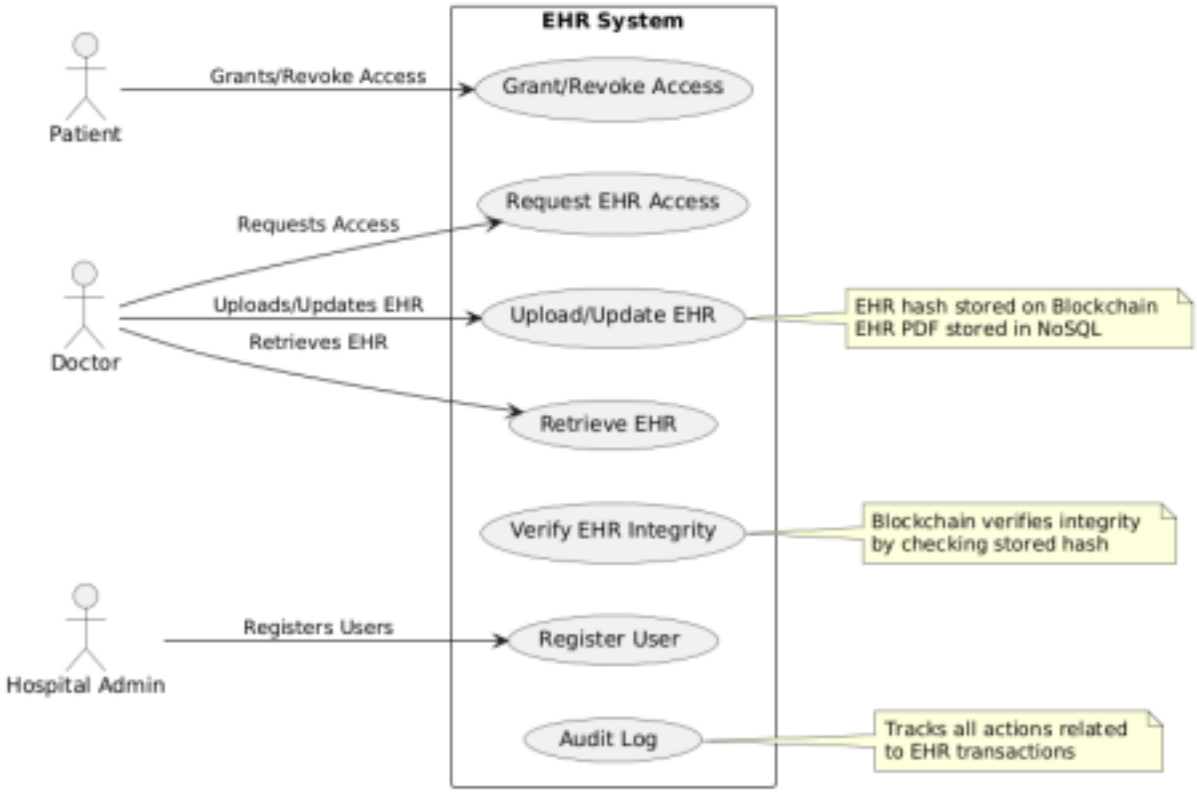
5.1 Sequence Diagram



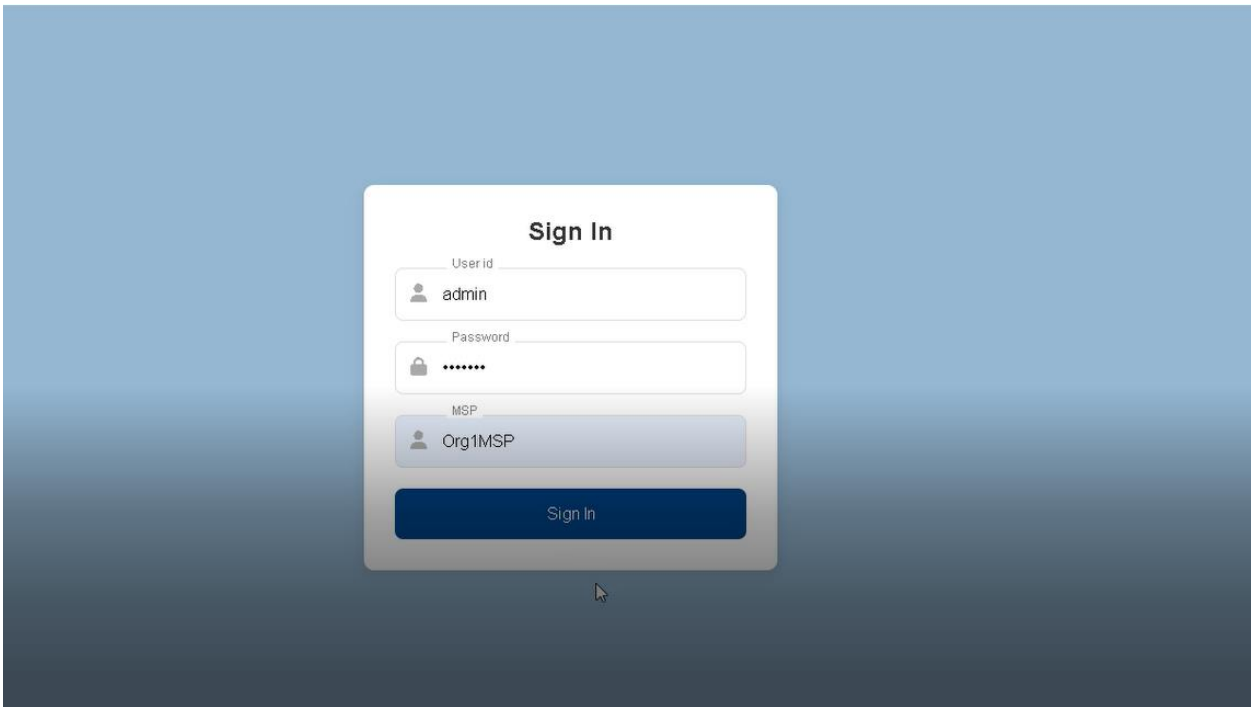
5.2 Activity Diagram

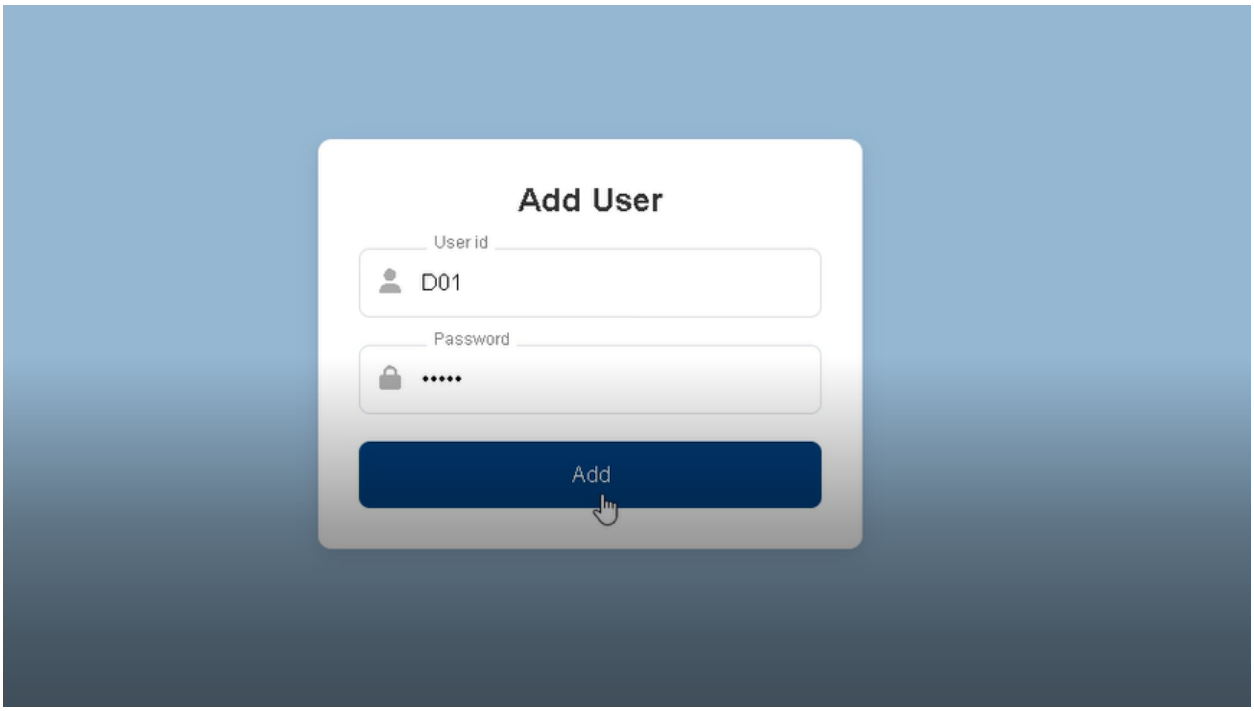


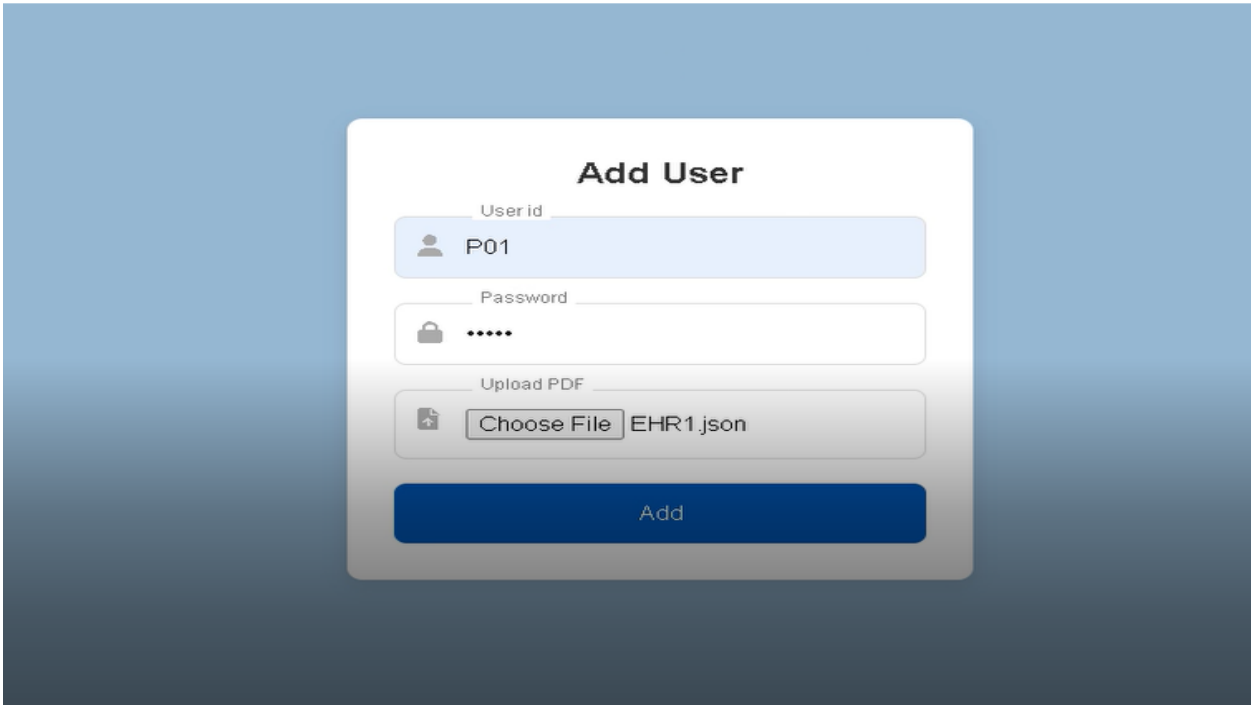
5.3 Use Case Diagram



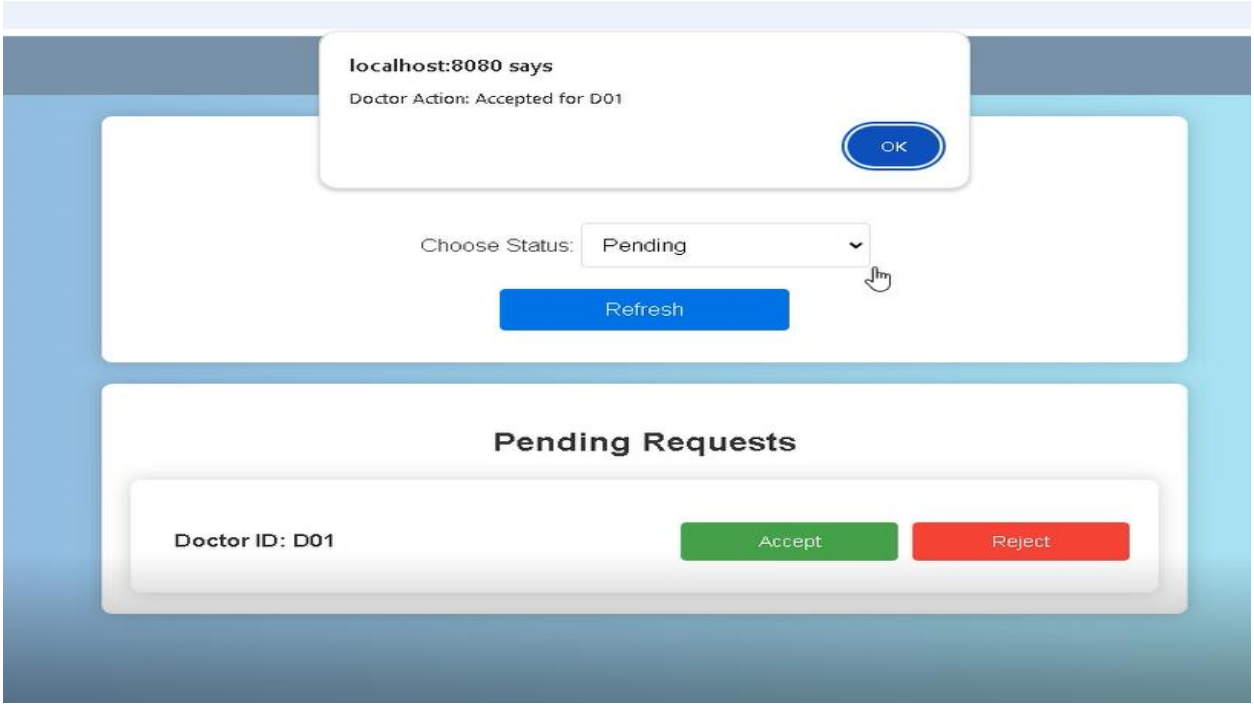
5.4 GUI

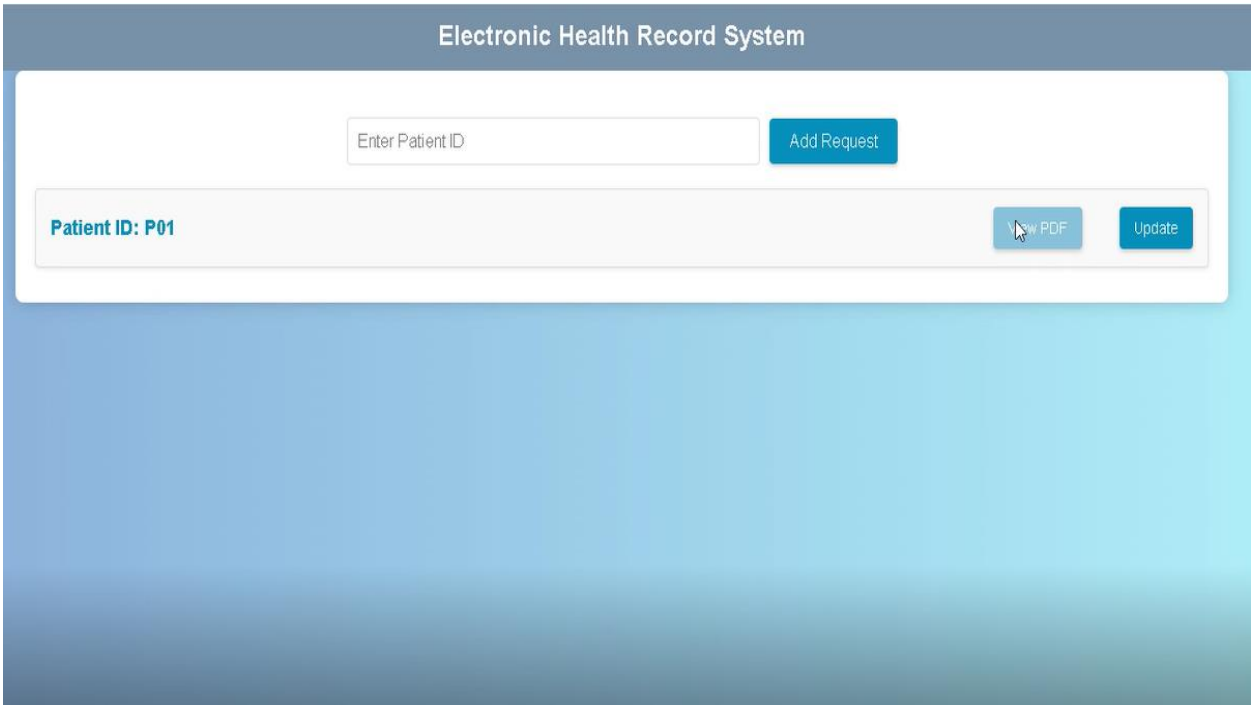


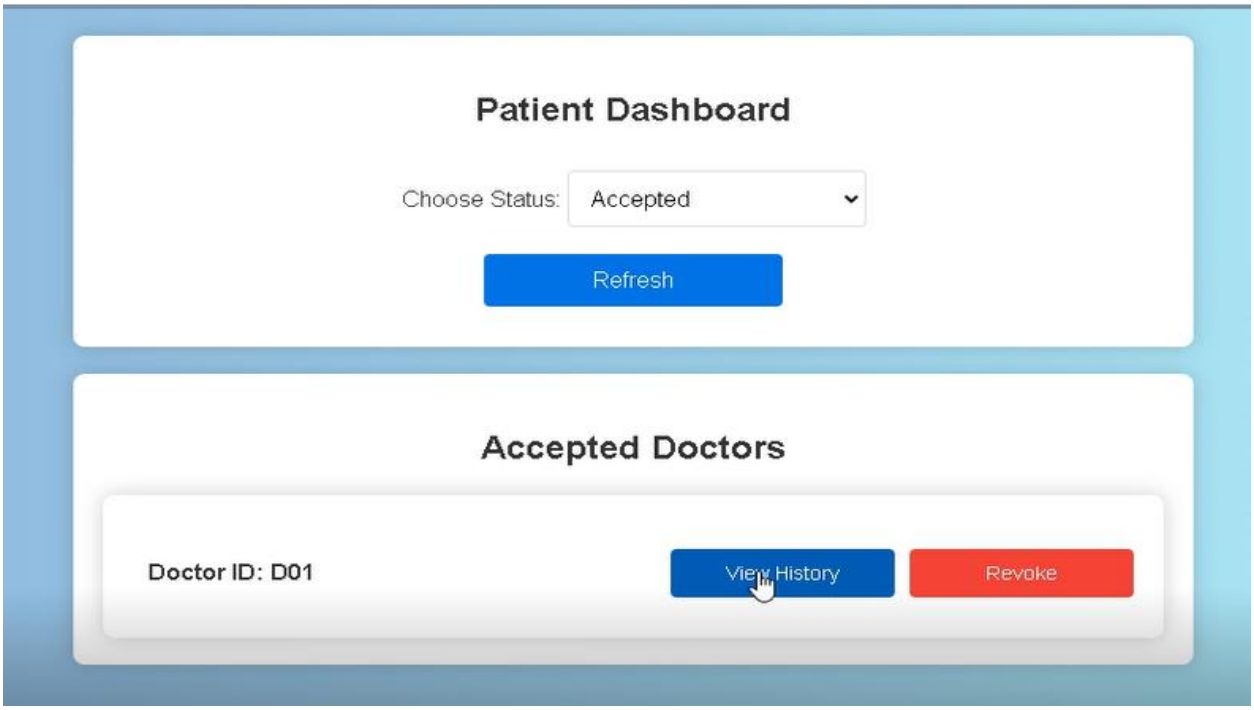

Add User
User id
D01
Password
•••••
Add

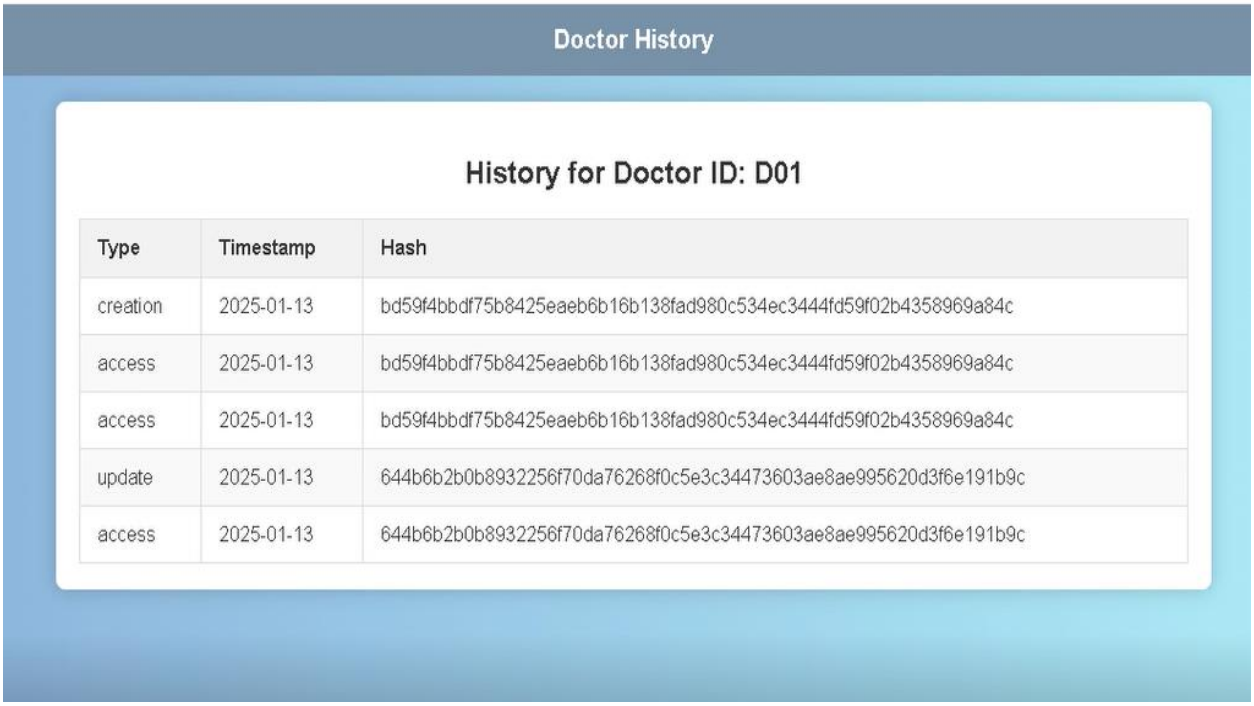

Add User
User id
P01
Password
•••••
Upload PDF
Choose File
EHR1.json
Add


Electronic Health Record System
Enter Patient ID
Add Request
P02
P01
P03


localhost:8080 says
Doctor Action: Accepted for D01
OK
Choose Status:
Pending
Refresh
Pending Requests
Doctor ID: D01
Accept
Reject


Electronic Health Record System
Enter Patient ID
Add Request
Patient ID: P01
Update


Patient Dashboard
Choose Status:
Accepted
Refresh
Accepted Doctors
Doctor ID: D01
View History
Revoke



---

6. Security Considerations

6.1 Authentication & Authorization

To ensure only legitimate users can access the system, a **multi-layered authentication and authorization** mechanism is implemented.

Key Security Measures:

1. **Identity Management with Hyperledger Fabric CA:**
    - Each **doctor, patient, and administrator** receives a **unique cryptographic identity** issued by **Hyperledger Fabric Certificate Authority (CA)**.
    - Hyperledger Fabric CA ensures that only authorized participants can interact with the blockchain network.
2. **JWT (JSON Web Token) for API Authentication:**
    - Once authenticated, users receive a **JWT token**, which must be provided in every API request.

- JWTs include role-based claims, ensuring **secure session management** and **stateless authentication**.

6.2 Data Integrity

Data integrity ensures that medical records remain **unchanged** and **tamper-proof** throughout their lifecycle. The system uses a **hybrid storage model** (MongoDB + Blockchain) to guarantee data integrity.

Key Security Measures:

1. **EHR Hashing with Blockchain for Tamper-Proof Records:**
    - When an EHR is created or modified, a **cryptographic hash** (SHA-256) of the record is generated.
    - The hash is stored on **Hyperledger Fabric**, ensuring immutability.
    - If someone alters the medical record in MongoDB, its new hash will **not match** the stored blockchain hash, indicating a potential security breach.
2. **Audit Logging for Full Traceability:**
    - Every action (EHR creation, modification, access request, approval/rejection) is logged in **Hyperledger Fabric's immutable ledger**.
    - **Patients and administrators** can view detailed logs to detect unauthorized access.
    - Logs cannot be altered, preventing malicious activity from being erased.

6.3 Access Control

To protect patient privacy, **strict access control policies** are enforced, ensuring that only **authorized users** can view or modify EHRs.

Key Security Measures:

1. **Patient-Centric Access Control:**
    - Patients **own** their medical records and can approve or reject doctor access requests.
    - No doctor can access a patient's record unless explicitly granted permission.
2. **Smart Contract Enforcement of Access Policies:**
    - A **Hyperledger Fabric Smart Contract (Chaincode)** is deployed to **automatically enforce** access control rules.
    - If a doctor attempts to access a record without patient approval, the smart contract will **block the transaction**.
    - Access logs are updated in real-time whenever permissions are changed.

---

7. Testing Strategy

7.1 Unit Testing

Test Type	**Tools Used**
Service Tests	JUnit 5, Mockito
Controller Tests	Spring Boot Test

7.2 Integration Testing

Component	**Tool Used**
API Testing	JUnit 5
Blockchain Testing	Hyperledger Fabric SDK

7.3 Security Testing

Aspect	Tool Used
Authentication	Spring Security Test

---

8. Deployment Strategy

The system is designed for a cloud-based deployment to ensure scalability, availability, and security. Each component is deployed strategically using Azure cloud services to provide high performance and fault tolerance.

Component	Deployment Environment
Frontend	Azure App Services
Backend	Dockerized Spring Boot
Blockchain	Hyperledger Fabric on Kubernetes
Database	MongoDB Atlas

---

9. Future Enhancements

9.1. IPFS Integration – Decentralized Storage for EHR Files

- **Current Challenge:**
    - EHRs are stored in **MongoDB**, which requires centralized storage and backup management.
    - Storing large medical files (e.g., scans, X-rays) directly in MongoDB can lead to **scalability and performance issues**.
- **Proposed Enhancement:**
    - Integrate **InterPlanetary File System (IPFS)** to store **EHR files in a decentralized manner**.

9.2. AI-Based Analytics – Predictive Healthcare Insights

- **Current Challenge:**
    - The system stores patient records but **lacks predictive analytics** to assist doctors and patients in proactive healthcare.
- **Proposed Enhancement:**
    - Integrate an **AI-powered analytics engine** to analyze historical EHR data and generate **predictive healthcare insights**.

9.3. Mobile App – Access EHR via Android/iOS

- **Current Challenge:**
    - Currently, patients and doctors access EHRs through a **web-based system**.
    - A lack of **mobile accessibility** limits usability, especially in emergency cases.
- **Proposed Enhancement:**

- Develop a **cross-platform mobile app** (Android & iOS) using **Flutter** or **React Native**.

---

10. Compliance & Standards

The system adheres to **industry-recognized standards and regulations** to ensure **data privacy, security, interoperability, and structured documentation**.

10.1. ISO/IEC/IEEE 26514:2022 – Documentation Structure

- **Purpose:** Defines best practices for **software documentation** to ensure clarity, consistency, and maintainability.
- **Application:**
  - System documentation follows a **structured format** covering **architecture, functionality, security, and deployment**.
  - Ensures **standardized terminology** and **readability** for different stakeholders (developers, admins, end-users).
  - Facilitates **compliance audits** and **regulatory reviews** with well-documented processes.

10.2. HL7 FHIR – EHR Data Handling Standards

- **Purpose: Fast Healthcare Interoperability Resources (FHIR)** is a standard by **Health Level Seven (HL7)** that enables seamless **exchange of EHR data** between healthcare systems.
- **Application:**
  - **FHIR-compliant data models** ensure structured storage and interoperability.

- Supports **FHIR RESTful APIs** for secure data sharing with other healthcare providers and systems.
- Ensures that **EHRs can be easily integrated** with hospitals, labs, and third-party health applications.

- **Key FHIR Resources Used:**
  - **Patient** → Stores patient demographics.
  - **Practitioner** → Represents doctors and healthcare providers.
  - **Observation** → Captures vital signs, test results, and clinical notes.
  - **Condition** → Stores patient diagnoses.

10.3. HIPAA & GDPR – Patient Data Privacy & Security

Regulation	Requirement	Implementation
HIPAA (U.S.)	Protect patient data & ensure confidentiality	- End-to-end encryption (AES-256, TLS 1.2+) - Access control (RBAC, Smart Contracts) - Audit trails for all EHR access
GDPR (EU)	Ensure data privacy & patient control over records	- Consent-based access control - Right to be forgotten (Data deletion on request) - Data minimization (Only essential data stored)

11. Conclusion

This **Software Design Document (SDD)** provides a comprehensive blueprint for the **Hyperledger Fabric-based Electronic Health Record (EHR) system**. The architecture is designed to ensure:

- **Tamper-Proof Storage** – EHR hashes are stored on the blockchain, preventing unauthorized modifications.
- **Controlled Access** – Patients retain control over their records, granting or revoking doctor access via smart contracts.
- **Auditability & Transparency** – Every transaction is logged immutably, ensuring compliance with healthcare regulations.