Software Requirements Specification

for

<HealthLink>

Version 2.0

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

<TKM College of Engineering>

Date: 30/01/2025

Table of Contents

Appendices

1. Introduction

1.1. Purpose

The purpose of this document is to define the functional and non-functional requirements for a blockchain-based Electronic Health Record (EHR) system. This system aims to provide a secure, decentralized, and transparent platform for managing patient health records, ensuring data integrity, privacy, and accessibility for authorized users while leveraging the unique advantages of blockchain technology

1.2. Document Conventions

- Lexical convention: The convention for language includes
  - Font Type: Times New Roman
- Headings: Bold and numbered hierarchically
- Requirement Priorities: o Requirements are classified as High, Medium, or Low priority.

1.3. Intended Audience and Reading Suggestions

- Intended audience includes -
  - Project Managers: To understand the scope, objectives, and deliverables of the project
  - Developers: For detailed requirements on system functionalities, design constraints, and integration needs.
  - Stakeholders/End Users: To review and confirm that the system will meet their needs and expectations.
- Reading suggestions -
  - Overview First: Start with the Introduction and Overall Description to grasp the system's purpose and context.
  - Specific Requirements: Review the Functional and Non-Functional Requirements for detailed system specifications.
  - External Interfaces: Examine External Interface Requirements for understanding system integration points.
  - Appendices: Check the Appendices for additional details, glossaries, or supplementary information.
  - References: Refer to the References section for source material and further reading.

1.4. Product Scope

The system includes the following key components:

- **Permissioned blockchain network** for secure and controlled access.
- **Smart contracts** to automate health record management.

- **Access control and consent management** for data privacy and user control.

1.5. References

- https://dspmuranchi.ac.in/pdf/Blog/srs_template-ieee.pdf
- Hyperledger Fabric Documentation v2.0
- HIPAA Security Rule Requirements

2. Overall description

2.1. Product Perspective

The system provides a secure platform for managing electronic health records using Hyperledger Fabric blockchain technology. It offers healthcare providers and patients a transparent, immutable system for managing medical records while ensuring data privacy and security. The system interfaces with healthcare providers and patients through secure dashboards, enabling real-time access to medical records, consent management, and secure sharing of health information. It helps in digitizing and securing the healthcare record system with two primary user groups: healthcare providers and patients.

2.2. Product Functions

Key features include:

- Patient Portal for viewing medical records, managing consent, and tracking record access
- Healthcare Provider Portal for managing patient records, requesting access, and updating medical information
- Smart Contracts for automated compliance and access control

2.3. User Classes and Characteristics

- **Patients**: Primary owners of their health records, requiring simple interface for viewing and managing consent
- **Healthcare Providers**: Medical professionals needing quick access to patient records
- **System Administrators**: Technical staff managing the blockchain network

2.4. Operating Environment

2.4.1 *Hardware Requirements*

Network Nodes

- **Minimum Server Requirements per Node**
    - CPU: 2 vCPUs (4+ recommended for production)
    - RAM: 4 GB (8 GB+ recommended for CouchDB)
    - Storage: 50 GB SSD (expandable based on ledger size)
    - Network: 1 Gbps connection
    - Recommended for Production
    - CPU: 4+ vCPUs
    - RAM: 16+ GB
    - Storage: 100+ GB SSD
    - Network: High-speed, low-latency connection

Client Systems

- **Healthcare Provider Workstations**
    - CPU: 4-core processor
    - RAM: 16GB
    - Storage: 256GB SSD
    - Display: 1920x1080 resolution minimum
- **Mobile Devices**
    - iOS 14.0 or later
    - Android 10.0 or later
    - Minimum 4GB RAM

2.4.2 *Software Requirements*

Blockchain Infrastructure

- **Platform**
    - Hyperledger Fabric v2.0 or later
    - Docker Engine 20.10.x or later
    - Docker Compose v2.0 or later
    - Nodejs 16.x

Operating Systems

- **Server**
    - Ubuntu Server 20.04 LTS or later
- **Client**
    - Windows 10/11 Professional or Enterprise
    - macOS 11 (Big Sur) or later

- Ubuntu Desktop 20.04 LTS or later

Database

- CouchDB 3.1 or later for state database
- MongoDB 5.0 or later for off-chain storage

Security

- TLS 1.3 for all communications
- X.509 certificate management system

2.4.3 Network Requirements

- Minimum 1Gbps network bandwidth
- Maximum latency of 100ms between nodes
- IPv6 support
- Load balancing capability
- Redundant network connections

2.5. Design and Implementation Constraints

- **Regulatory: HIPAA, GDPR compliance required**
  he system must comply with **HIPAA** and **GDPR**, ensuring data encryption, access controls, breach notifications, and user rights management. It should efficiently handle large medical files like DICOM and MRI scans, using scalable cloud storage and high-performance processing.
- **Performance: Must handle large medical files**
  Optimized for speed and security, it must support role-based access, and cross-border data compliance while maintaining seamless user access and regulatory adherence.

2.6 User Documentation

2.6.1 System Documentation

1. **System Architecture Guide**
   - The **network topology** defines how system components communicate securely and efficiently.
   - The **node configuration** specifies hardware and software settings for optimal system performance.
   - The **blockchain architecture** outlines the structure, consensus mechanism, and data integrity protocols.
   - The **security implementation** describes encryption, authentication, and access control measures.

   - The **API documentation** provides details on endpoints, authentication, and data exchange formats.
2. **Installation and Setup Guide**
   - The **hardware requirements** list the minimum and recommended specifications for system deployment.
   - The **software prerequisites** outline necessary operating systems, dependencies, and libraries.
   - The **network configuration** details IP settings, firewall rules, and connectivity requirements.
   - The **security setup** explains initial authentication, encryption, and compliance configurations.
   - The **troubleshooting procedures** offer solutions for common installation and setup issues.
3. **System Administration Manual**
   - The **node management** section provides instructions for deploying, maintaining, and updating nodes.
   - The **user management** section explains account creation, role assignments, and access control.
   - The **security policies** define best practices for data protection, compliance, and risk mitigation.
   - The **backup procedures** ensure data recovery through scheduled and manual backups.
   - The **performance monitoring** section describes tools and metrics for system health tracking.
   - The **maintenance schedules** specify routine tasks to keep the system running optimally.

3. External interface requirements

External interface requirements define how a system interacts with other systems, external components, or users. These requirements ensure compatibility, functionality, and communication between the system being developed and external elements.

3.1. User Interfaces (UI)

- A **web-based interface** will be available for all user types, providing secure and intuitive access to system functionalities.

3.2 Software Interfaces

- The **Hyperledger Fabric SDK** enables interaction with the blockchain network for transaction processing and smart contract execution.
- The **MongoDB** database stores data, ensuring efficient and scalable data management.

3.3 Blockchain Network Interface

- The **chaincode interfaces** facilitate smart contract execution within the Hyperledger Fabric network.

- The **peer-to-peer network protocols** enable secure communication between blockchain nodes.
- The **ordering service interfaces** manage transaction sequencing and consensus mechanisms.
- The **Certificate Authority interfaces** handle identity management and access control through digital certificates.

4. System Features

4.1. Patient Record Management

- Description: Enables creation and management of patient health records
- Requirements:
  - Record creation with unique identifiers
  - History tracking
  - Attachment support for medical images

4.2 Healthcare Provider Management

- Description: Manages healthcare provider registration and access
- Requirements:
  - Provider verification and credentialing
  - Role-based access control
  - Access level configuration

4.3 Consent Management

- Description: Handles patient consent for data access
- Requirements:
  - Consent revocation mechanism
  - Audit trail of consent changes and all accesses.

4.4 Smart Contract Implementation

- Description: Automated execution of business logic and compliance rules
- Requirements:
  - Access control enforcement
  - Consent validation

- Automated compliance checks
- Event logging

4.5 <u>Testing</u>

- ***Unit Testing (Spring Boot)***

  Goal**:** Test individual methods in services, repositories, and controllers.

  Frameworks & Libraries:

  JUnit 5 (5.10.0) : Main testing framework

  Spring Boot Test(3.2.0) : For testing controllers and services

- ***Integration Testing (Spring Boot,Hyperledger Fabric)***

Goal: Test the full workflow including API, blockchain, and database interactions.

Frameworks & Tools:

Testcontainers : Run Hyperledger Fabric & MongoDB(1.19.4) in Docker for real environment testing.

Spring Boot Test (3.2.0): For API & service integration.

Fabric Gateway SDK(2.2.14) : Connects to the Fabric blockchain

- ***End-to-End Testing (Hyperledger Fabric & NoSQL)***

  Goal: Simulate the entire Doctor-Patient interaction with real blockchain transactions.

  Tools Required:

  Postman V10 : Automate API request testing.

  Hyperledger Explorer (1.1.9) : Validate transaction logs.

  MongoDB Queries : Verify stored EHR data.

- ***Security Testing***
  Test used:

  Spring Security Test(3.2.0) : Validate authentication & JWT tokens

  Hyperledger Fabric Access Control Policies : Enforce patient-doctor permissions

5. Non-Functional Requirements

The non-functional requirements define the quality attributes, performance, and operational constraints that the system must adhere to, ensuring it performs efficiently and reliably.

5.1. <u>Performance Requirements</u>

- PR1: Response Time

i. The system should respond within 3 seconds for most interactions.

- PR2: Scalability

ii. It should support up to 10,000 active users without significant performance degradation.

- PR3: Throughput

iii. The system should handle at least 100 transactions per minute to accommodate high traffic volumes.

- PR4: Availability

iv. The system must ensure 99.9% uptime annually, with minimal downtime limited to scheduled maintenance.

5.2. Security Requirements

- SR1: User Authentication

v. The system should implement secure login with username/password combinations and for admins.

- SR2: Data Integrity

vi. Ensure data integrity using checksums or hashing to detect unauthorized data modifications.

- SR3: Access Control

vii. Role-based access controls must be in place to restrict functionalities based on user roles (clients, admins).

5.3. Usability Requirements

- UR1: User Interface Design

viii. The system should have an intuitive, easy-to-navigate interface that minimizes the learning curve for both clients and administrators. Buttons, menus, and fields should be clearly labeled and easily accessible.

- UR2: Consistency

ix. The design should maintain consistency across all pages and functions, ensuring uniform fonts, colors, and layouts to provide a cohesive user experience.

- UR3: Mobile Responsiveness

x. The system must be fully responsive, allowing clients and admins to access and use the system seamlessly on various devices, including mobile phones and tablets.

- UR4: Error Handling and Feedback

xi. Clear and helpful error messages should be provided when users make mistakes, along with appropriate guidance on how to fix the error. Feedback for successful transactions (e.g., bill payments) should also be displayed promptly.

- UR5: Accessibility

xii. The system should comply with accessibility standards (such as WCAG 2.1) to ensure that users with disabilities can easily access the platform. Features like screen reader compatibility, keyboard navigation, and color contrast must be considered.

6. User Objectives and Interfaces

6.1 Patient User Objectives

6.1.1 EHR Access Control Management

- **Permission Management**
  - Accept permission requests from healthcare providers to access their EHR
  - Reject permission requests from unauthorized or unknown healthcare providers
  - Revoke previously granted permissions at any time
  - Set expiration dates for granted permissions
  - Specify access levels (full access, partial access, emergency access)

6.1.2 Access History Monitoring

- **View Access History**
  - See complete chronological list of all EHR access attempts
  - View successful and unsuccessful access attempts
  - Filter access history by:
    - Healthcare provider
    - Date range
    - Access type (read, write, emergency)
    - Permission status (granted, rejected, revoked)

6.1.3 EHR Viewing Capabilities

- **Personal Health Record Management**
  - View complete EHR including:

        - Medical history
        - Prescriptions
        - Lab results
        - Diagnostic reports
        - Treatment plans
        - Allergies and medications
    - Download EHR in standard formats
    - Share specific portions of EHR with selected providers
    - Track record updates and modifications

6.1.4 Granular Permission Control

- **Detailed Access Configuration**
    - Set permissions for specific sections of EHR:
        - Allow access to only specific test results
        - Restrict access to sensitive information
        - Grant temporary access for consultations
        - Configure emergency access protocols
    - Manage permission hierarchies for:
        - Primary care physicians
        - Specialists
        - Emergency care providers
        - Insurance providers

6.2 <u>User Interface Requirements</u>

6.2.1 Permission Management Interface

- **Permission Request Handling**
    - Clear display of pending permission requests showing:
        - Requesting provider's credentials
        - Requested access level
        - Requested duration
    - One-click accept/reject buttons
    - Batch permission management for multiple requests

6.2.2 Access History Dashboard

- **History Visualization**
    - Timeline view of all access events
    - Graphical representation of access patterns
    - Detailed access logs showing:
        - Timestamp of access
        - Provider information
        - Type of access
        - Records accessed
        - Duration of access
    - Export functionality for access history reports

6.2.3 EHR Viewing Interface

- **Record Navigation**
    - Categorized view of health records
    - Search functionality within records
    - Filter options for different types of medical data
    - Mobile-responsive design

6.2.4 Smart Contract Implementation

- **Automated Permission Enforcement**
    - Smart contracts to enforce:
        - Access control rules
        - Emergency access protocols
        - Audit trail maintenance
    - Automatic notification triggers for:
        - Permission changes
        - Access attempts
        - Record updates

6.3 Technical Implementation Requirements

6.3.1 Blockchain Features

- **Permission Management**

    - Immutable record of all permission changes
    - Cryptographic verification of access rights
    - Decentralized storage of permission settings
    - Smart contract automation for permission enforcement

6.3.2 Security Measures

- **Access Control**
    - Encrypted communication channels
    - Secure key management

6.3.3 Performance Requirements

- **System Response**
    - Permission changes processed within 3 seconds
    - Access history retrieved within 2 seconds
    - Real-time notification delivery
    - Concurrent user support for large-scale deployment

6.3.4 Integration Capabilities

- **System Interoperability**
    - HL7 FHIR compliance for data exchange
    - API support for third-party integration
    - Standard format support for record export
    - Legacy system compatibility

6.4. <u>User Interactions</u>

Patient interactions

6.4.1 Authentication and Access

- **Login Process**
    - Multi-factor authentication using:
        - Username/password combination
    - Remember device option for trusted devices
    - Password recovery mechanism

    - Session timeout after 15 minutes of inactivity

6.4.2 Dashboard Navigation

- **Main Dashboard Elements**
    - Summary of recent activities
    - Pending permission requests
    - Latest health record updates
    - Upcoming appointments
    - Quick action buttons for common tasks

6.4.3 Permission Management Workflow

1. **View Permission Requests**
    - List of pending requests showing:
        - Doctor's name and credentials
        - Hospital/clinic affiliation
        - Requested access level
        - Purpose of access
        - Duration requested
2. **Handle Permission Requests**
    - Actions available:
        - Accept request with optional modifications
        - Reject request with reason
        - Request more information
        - Set custom access parameters
3. **Modify Existing Permissions**
    - View active permissions
    - Extend/reduce access duration
    - Modify access levels
    - Revoke access with reason

6.4.4 Health Record Access

1. **Record Navigation**
    - Hierarchical menu structure

    - Category-based filtering
    - Search functionality
    - Timeline view of medical history

2. **Record Viewing**
    - Detailed view of individual records
    - Zoom in/out for images
    - Download options
    - Print formatting
    - Share functionality

Healthcare Provider Interactions

6.4.5 Provider Authentication

- **Login Process**
    - Digital certificate validation
    - Role-based access initialization

6.4.6 Patient Record Access Workflow

1. **Request Access**
    - Search for patient using:
        - Patient ID
        - Name and DOB
        - Insurance information
    - Specify access requirements:
        - Access level required
        - Purpose of access
2. **Record Interaction**
    - View authorized records
    - Add new entries
    - Update existing information
    - Attach lab results/images
    - Add prescriptions

Administrator Interactions

6.4.7 System Management

- **User Management**
  - Create/modify user accounts
  - Assign roles and permissions
  - Handle account issues
  - Monitor user activities

6.4.8 Network Monitoring

- **Blockchain Network Management**
  - Monitor node status
  - View transaction metrics
  - Handle consensus issues
  - Manage smart contracts

Interface Requirements

6.4.9 Web Interface

- **Design Requirements**
  - Responsive layout
  - Consistent navigation structure
  - Clear visual hierarchy
  - Accessibility compliance
  - Support for multiple languages

7.Scalabilty

- The system must be scalable, supporting high-throughput processing of large medical files like DICOM and MRI scans. A microservices architecture with containerization (Docker, Kubernetes) for flexibility, allowing seamless expansion as data volume and user demand grow.

8. Other Requirements

- No additional requirements have been determined yet.

Appendices

Appendix A: Glossary

- **Chaincode**: In Hyperledger Fabric, chaincode is the smart contract that defines business logic and governs transactions on the blockchain.
- **MSP**: The Membership Service Provider (MSP) in Hyperledger Fabric manages identities, authentication, and access control for participants in the network.
- **Access Control** - A security mechanism that restricts access to data based on user roles and permissions.

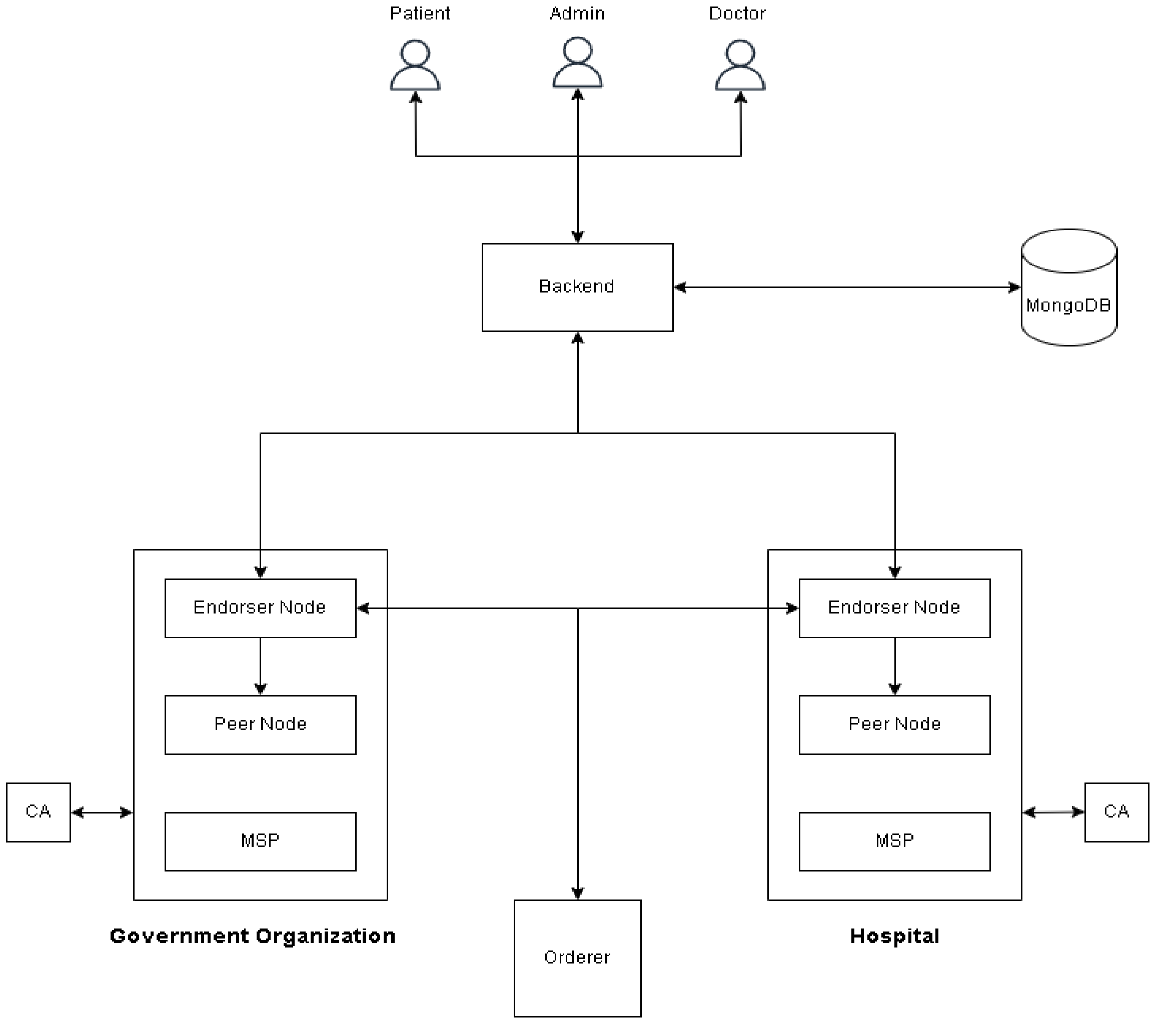
- **API (Application Programming Interface)** - A set of functions and protocols that allow different software applications to communicate with each other.

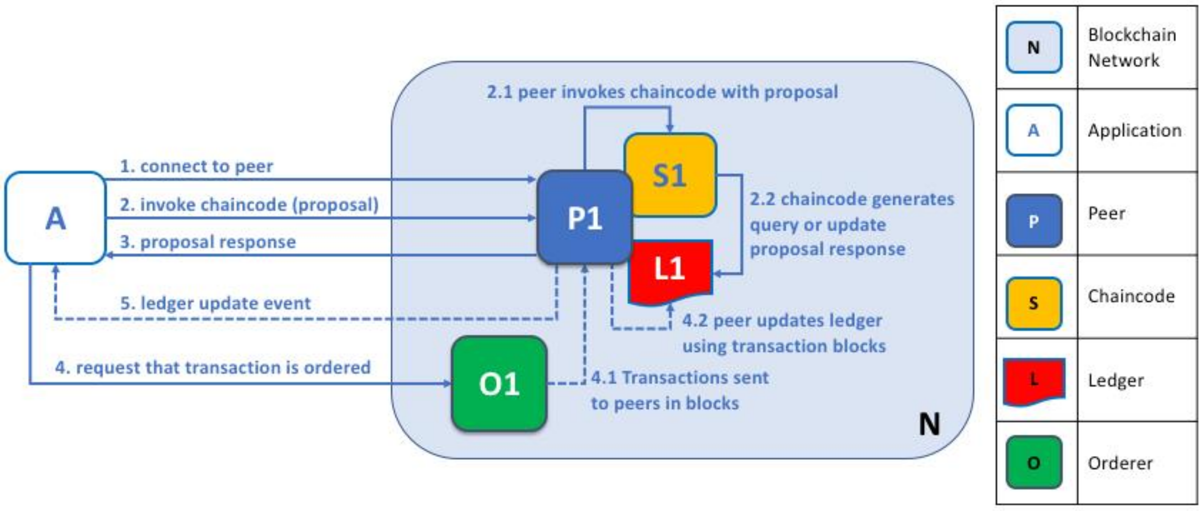
- **Audit Trail** - A record of all actions and changes made within the system, ensuring transparency and accountability.
- **Authentication** - The process of verifying a user's identity before granting access to the system.
- **Authorization** - A security feature that determines what a verified user is allowed to do within the system.
- **Blockchain** - A decentralized and distributed digital ledger used to record transactions securely and immutably.
- **Certificate Authority (CA)** - An entity responsible for issuing and managing digital certificates for identity verification.
- **CouchDB** - A NoSQL database used for storing state data in a blockchain network.
- **Consent Management** - The process by which patients grant or revoke access to their health records.
- **Cryptographic Hashing** - A security process that converts data into a fixed-length hash value to ensure integrity.
- **DICOM (Digital Imaging and Communications in Medicine)** - A standard for handling, storing, and transmitting medical images.
- **Digital Signature** - A cryptographic technique used to verify the authenticity and integrity of digital messages or documents.
- **Electronic Health Record (EHR)** - A digital version of a patient's medical history, maintained over time by healthcare providers.
- **Encryption** - A method of converting data into a secure format to prevent unauthorized access.
- **FHIR (Fast Healthcare Interoperability Resources)** - A standard for exchanging healthcare information electronically.
- **HIPAA (Health Insurance Portability and Accountability Act)** - A U.S. law that mandates data privacy and security provisions for safeguarding medical information.
- **Hyperledger Fabric** - A permissioned blockchain framework designed for enterprise use cases, including secure EHR management.
- **Immutable Ledger** - A data structure where once recorded, information cannot be altered or deleted.
- **Interoperability** - The ability of different healthcare systems to exchange and use health information seamlessly.
- **JSON (JavaScript Object Notation)** - A lightweight data format used for exchanging structured data between a server and a client.
- **Latency** - The delay between a user's action and the system's response.

- **Load Balancing** - A technique used to distribute network or application traffic across multiple servers to improve performance and reliability.
- **Microservices Architecture** - A software development approach where applications are composed of small, independent services that communicate via APIs.
- **MongoDB** - A NoSQL database used for storing off-chain medical data in the EHR system.
- **Node (Blockchain Node)** - A participant in a blockchain network that maintains a copy of the ledger and validates transactions.
- **Ordering Service** - A component of Hyperledger Fabric responsible for maintaining the sequence of transactions and ensuring consensus.
- **Permissioned Blockchain** - A blockchain that restricts access to approved participants, enhancing security and control.
- **Private Key** - A cryptographic key used to sign transactions and access encrypted data securely.
- **Public Key Infrastructure (PKI)** - A framework for managing digital keys and certificates.
- **RBAC (Role-Based Access Control)** - A security mechanism that assigns system permissions based on user roles.
- **Scalability** - The ability of the system to handle increasing amounts of workload or users efficiently.
- **Smart Contracts** - Self-executing contracts with predefined rules, enabling automated transactions and compliance checks.
- **TLS (Transport Layer Security)** - A protocol that secures communications over a network.
- **Transaction Per Second (TPS)** - A metric used to measure the number of transactions a system can process in one second.

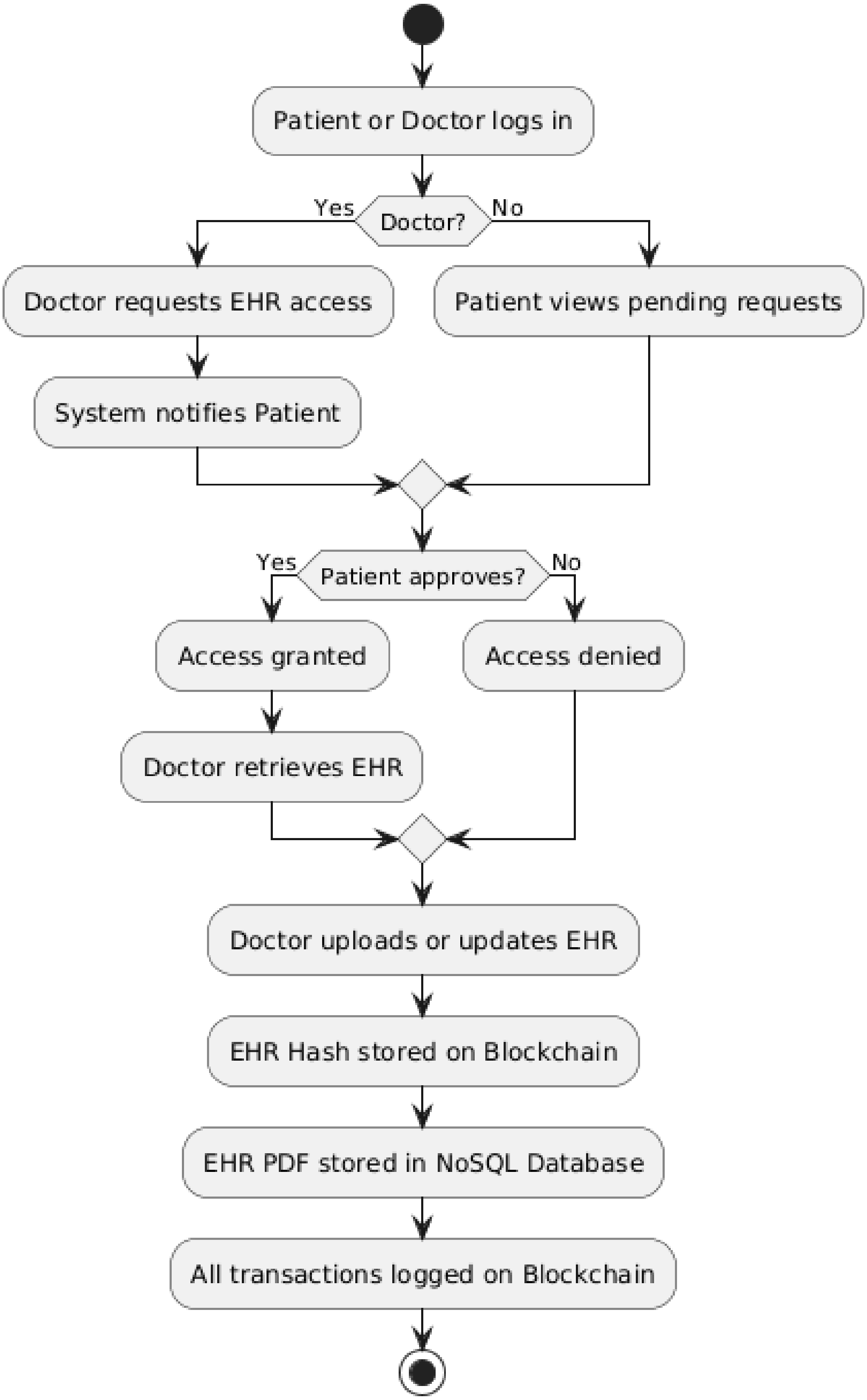
Appendix B: Analysis Models

i. *Architecture*


Patient
Admin
Doctor
Backend
MongoDB
Endorser Node
Peer Node
MSP
CA
Government Organization
Orderer
Endorser Node
Peer Node
MSP
CA
Hospital

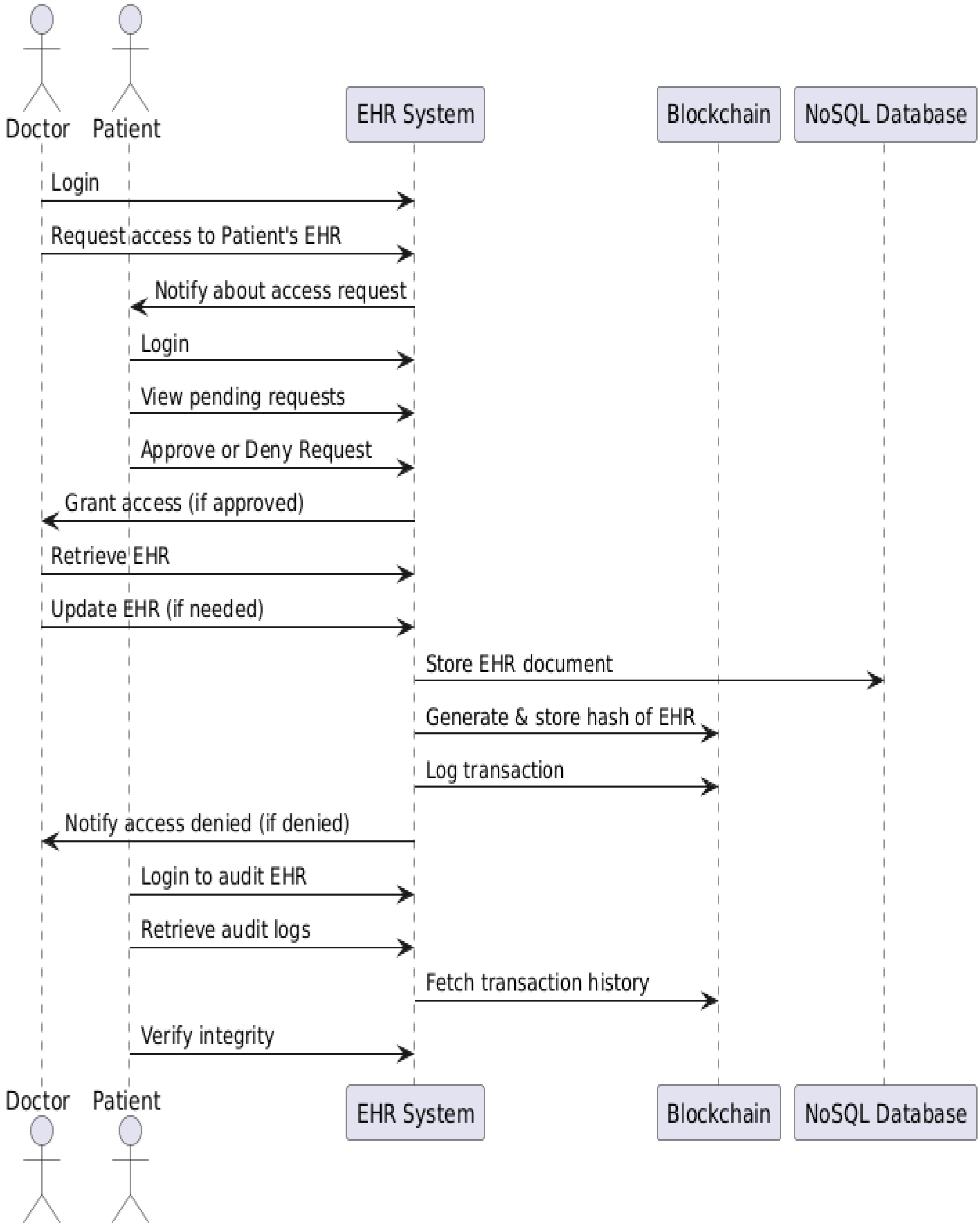

2.1 peer invokes chaincode with proposal
1. connect to peer
2. invoke chaincode (proposal)
3. proposal response
A
P1
S1
L1
2.2 chaincode generates query or update proposal response
5. ledger update event
4.2 peer updates ledger using transaction blocks
4. request that transaction is ordered
O1
4.1 Transactions sent to peers in blocks
N
N Blockchain Network
A Application
P Peer
S Chaincode
L Ledger
O Orderer

ii. Activity Diagram

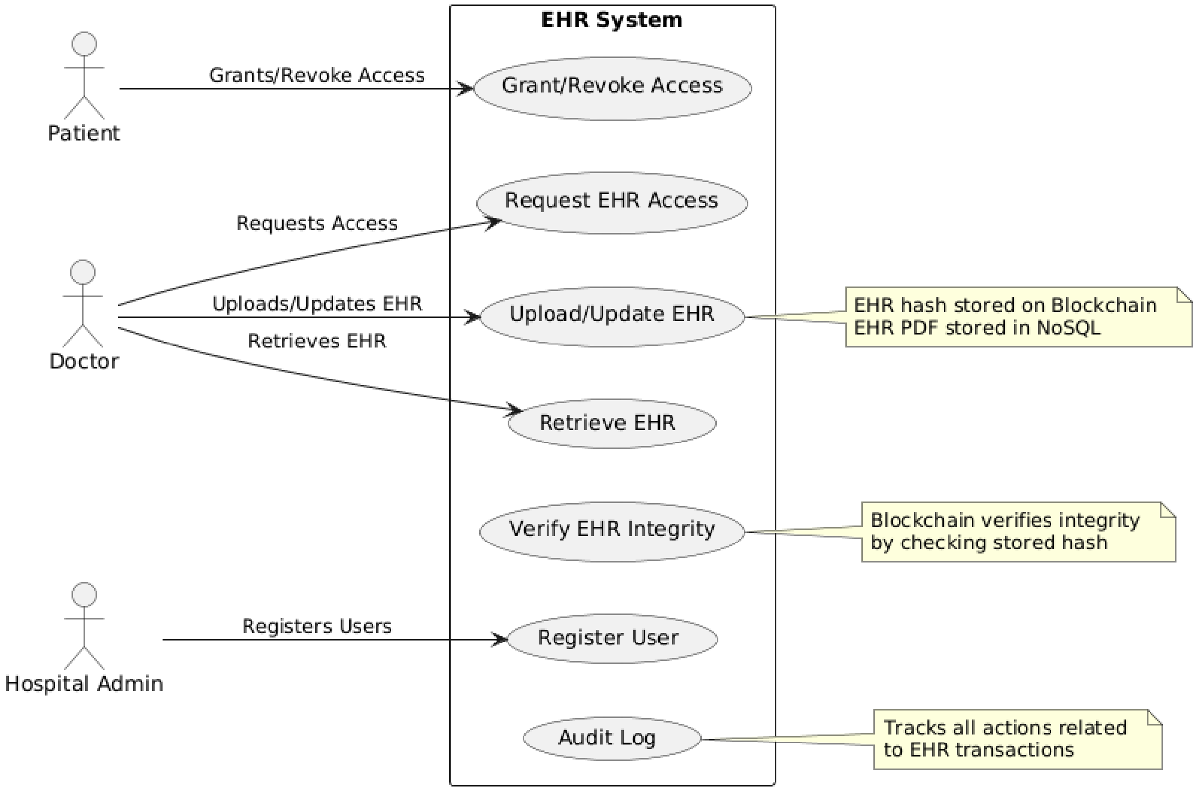


Patient or Doctor logs in
Yes
Doctor?
No
Doctor requests EHR access
Patient views pending requests
System notifies Patient
Yes
Patient approves?
No
Access granted
Access denied
Doctor retrieves EHR
Doctor uploads or updates EHR
EHR Hash stored on Blockchain
EHR PDF stored in NoSQL Database
All transactions logged on Blockchain

iii. Sequence Diagram



iv. Use Case Diagram



Appendix C: Issues List

- Performance optimization for large-scale deployment
- Key management infrastructure
- Consensus mechanism selection