Project Progress Report

Curis Catena

Version 1.0 approved
March 29, 2018

Prepared by:

Amaal Khalid
Shereen Sallam
Yasmin ElDokay

Revised by:

Dr. Sherif El-Kassas

Table of Contents

Revision History

Date	Reason for Updates	Version
29/3/18	Initial Draft	1.0

List of Figures

1. Introduction

Curis Catena is an EPRs management and exchange system that allows the secure and efficient sharing and modification of patient medical data, using a blockchain-based infrastructure, while abiding by HIPAA healthcare IT system standards.

1.1 Purpose

The purpose of this document is to show a narrative and graphical documentation of *Curis Catena's* project progress and proposed system design. This document is to be used as a reference tool for our future software development process, as it provides a more technical perspective of the system's low-level functionalities and operations, as well as insight into the design and structure of its sub-components, especially the blockchain-related ones.

1.2 Scope

The *Curis Catena* system is designed to be a middleware for interactions between patients and healthcare facilities' providers, where:

- Patients will be able to remotely view their EPRs and control who can access and modify them at any point in time.
- Healthcare providers will be offered remote access to their patients' EPRs, provided authorization from their patients and their own healthcare facilities.
- Healthcare facilities will have a secure system to maintain EPR communication with their patients, as well as control over the authorities of their healthcare providers and system administrators.

1.3 Overview

This report is divided into six main sections. The first section aims to introduce the system's key functionalities and provide an overview of the document's contents, references and terminology. The second section summarizes the fruits of last fall's project research, which involves blockchain architectures and design aspects, available blockchain development platforms, the system's entities and users and HIPAA compliance feasibility. The third section elaborates on the chosen layered architecture of the system and the roles of its key sub-systems, especially the blockchain-based one, which will be using the Hyperledger Fabric framework. The fourth section showcases the system's human interfaces for healthcare providers, their expected functionalities and currently available prototype. The fifth section illustrates the project's development timeline for this semester and the final one provides an explanatory sequence diagram to elaborate on the typical transaction flow taking place on our system.

1.4 Reference Material

The following references were used for the creation of this document:

1. Androulaki, Elli, et al. "Cryptography and Protocols in Hyperledger Fabric." Real-World Cryptography Conference.2017. https://cachin.com/cc/talks/20170106-blockchain-rwc.pdf
2. Antonopoulos, Andreas M. Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Incorporated, 2017. http://chimera.labs.oreilly.com/books/1234000001802/ch07.html#_structure_of_a_block
3. Ethereum. *Ethereum Whitepaper*. Ethereum/Wiki, GitHub, 18 Sept. 2017, github.com/ethereum/wiki/wiki/White-Paper.
4. “Hyperledger Architecture, Volume 1.” Hyperledger, 2017. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf
5. Sousa, Joao, Alysson Bessani, and Marko Vukolić. "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform." arXiv preprint arXiv:1709.06921 (2017). https://arxiv.org/pdf/1709.06921.pdf
6. McConaghy, Trent, et al. “BigchainDB: A Scalable Blockchain Database.” 8 June 2016, www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf
7. Wang, Jason. “How Do I Become HIPAA Compliant?”, TrueVault, 30 Oct. 2013, www.truevault.com/blog/how-do-i-become-hipaa-compliant.html.

1.5 Terminology Glossary

Term	Meaning
Anchor Peer	Peer node on a channel that all other peers can discover and communicate with
Block	Ordered set of transactions cryptographically linked to preceding block(s) on a channel
Chain	Transaction log structured as hash-linked blocks of transactions
Chaincode	Ledger software that encodes assets and transaction instructions for asset modification
Channel	Private blockchain overlay which allows for data isolation and confidentiality
Consensus	Transaction flow to order and validate a set of transactions constituting a block

Endorsement	Process where specific peer nodes execute a chaincode transaction and return a proposal response to the client application
EPR	Electronic Patient Record
EPRM	Electronic Patient Records Management system
Genesis Block	First block on a chain that initializes a blockchain network or channel
HF	Healthcare Facility system
HIPAA	Health Insurance Portability and Accountability Act of 1996
HPS	Healthcare Provider system
Hyperledger Fabric CA	Default Certificate Authority system component
Ledger	Channel's chain and current state data which is maintained by each peer on the channel
Member	Legally separate entity that owns a unique root certificate for the network
Membership Services	Service that authenticates, authorizes, and manages identities on a permissioned blockchain network
MSP	Membership Service Provider
Ordering Service	Defined collective of network nodes that order transactions into blocks
Peer	Network entity for maintaining a ledger and running chaincode containers
Proposal	Request for endorsement aimed at specific peers on a channel
SDK	Software Development Kit

2. Research Overview

2.1 Blockchain Architectures

Blockchain is a peer-to-peer distributed ledger that utilizes smart contracts alongside consensus algorithms to establish agreement between mutually distrustful parties using a single source of truth in real-time.

2.1.1 Blockchain Network Architecture

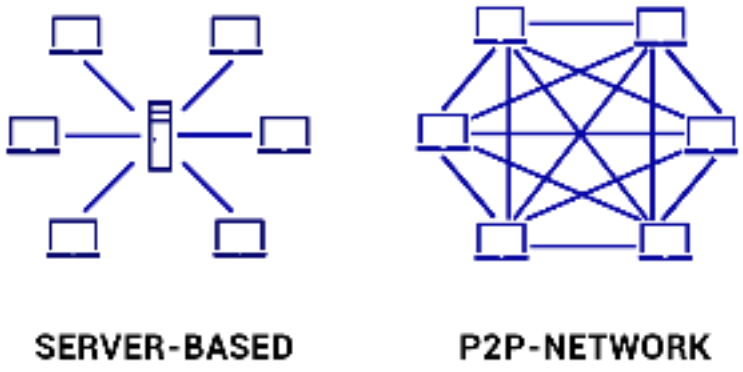

Figure 1 | Server-based versus P2P Network Architectures

As shown in figure 1, blockchain networks are designed as purely peer-to-peer networks, which compared to server-based networks, depend entirely on network peers for computational power and storage. This decentralized model, compared to depending on a central server, ensures that no single point of security or communication failure could take place.

2.1.2 Blockchain Data Structure

Transaction data on a blockchain network is stored using a distributed ledger, where each network peer has a copy of the most up-to-date ledger and the network's employed consensus algorithm ensures its synchronization and validity. Each ledger, as shown in figure 2, is designed as a back-linked data structure where each block of transactions, following the first block known as the *genesis block*, is linked to the previous block using its transaction hash value.

Each ledger block contains a timestamp, nonce, the hash of its precedent block's transactions and a hash of all the transactions committed to the ledger at the point of its creation using a merkle tree structure, as illustrated by block 11 in figure 2. Merkle trees are binary hash trees

that provide an anti-temperament summary of each block's transactions that can be used for quick ledger verification, as well as for the prevention of any attempts at altering transaction records. As shown in figure 3, if a malicious peer were to attempt to change the value of Transaction A to that of Transaction A', they would have to change the tree's root hash value in addition to those of all subsequent blocks to block 11, and then recalculate all of the blocks' nonce values, which is easily prevented by currently in-use consensus algorithms.

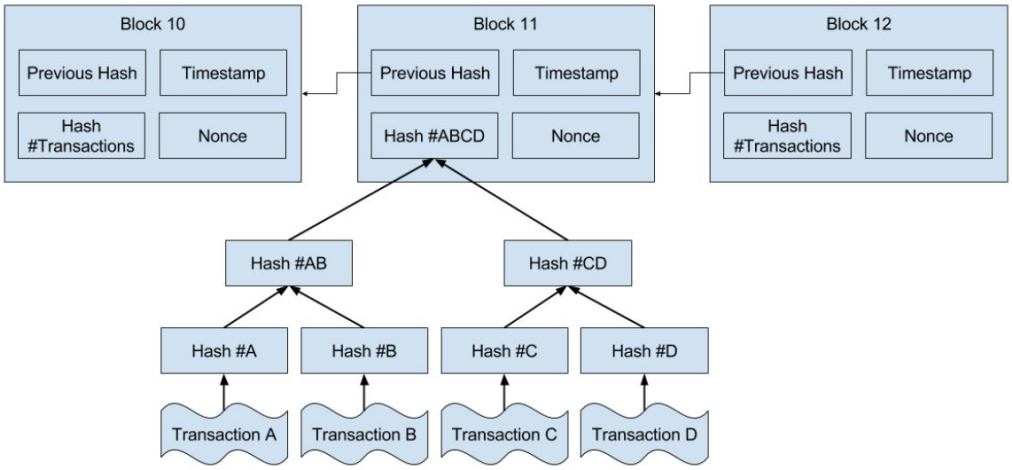

Figure 2 | Blockchain Ledger Data Structure

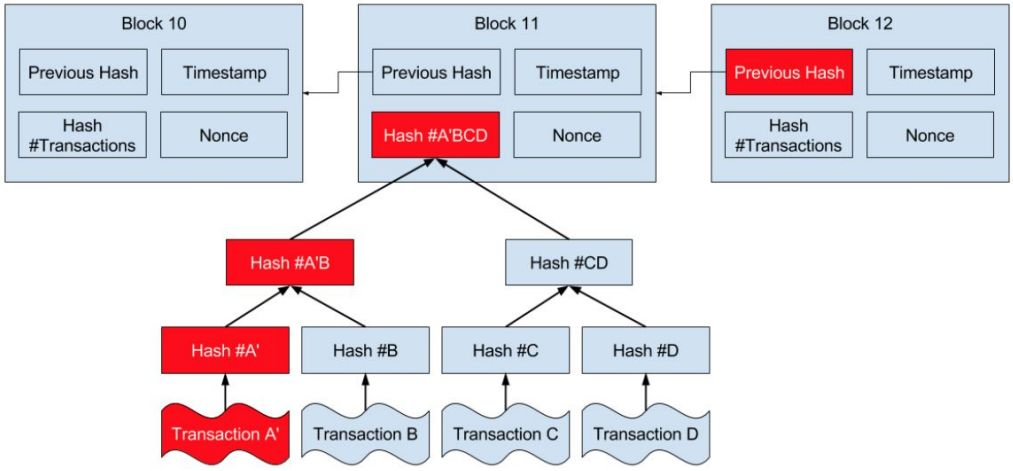

Figure 3 | Immutability of Blockchain Ledger Data Structure

2.1.3 Types of Blockchain Architectures

There are two main types of blockchain architectures: permissioned (private) and permissionless (public) blockchains, which serve different business models and use-cases.

Permissioned blockchains, such as the ones followed by Hyperledger blockchain architectures, provide closed networked distributed ledgers where all peers are have known unique identities

and must be authorized to access the network, while permissionless blockchains grant access to any users, use public protocols, and allow user anonymity. Known permissionless blockchains include Bitcoin and Ethereum.

Permissioned blockchains have better uptime and quality of service, as they ensure better security services and access control capabilities compared to permissionless blockchains. However, permissionless blockchains tend to be faster and more scalable.

Hybrid blockchain models have also been proposed, which can have varying access control protocols depending on user groups on the blockchain and provide a high scalability compared to permissioned blockchains, as well as allow various users anonymity when suitable. The only currently launched hybrid blockchain platform we found is called BigchainDB.

2.1.4 Smart Contracts

For every transaction taking place on the network, a smart contract is a set of instructions that get executed when predefined network policies are met during the exchange of assets between the transaction parties, as shown in figure 4.

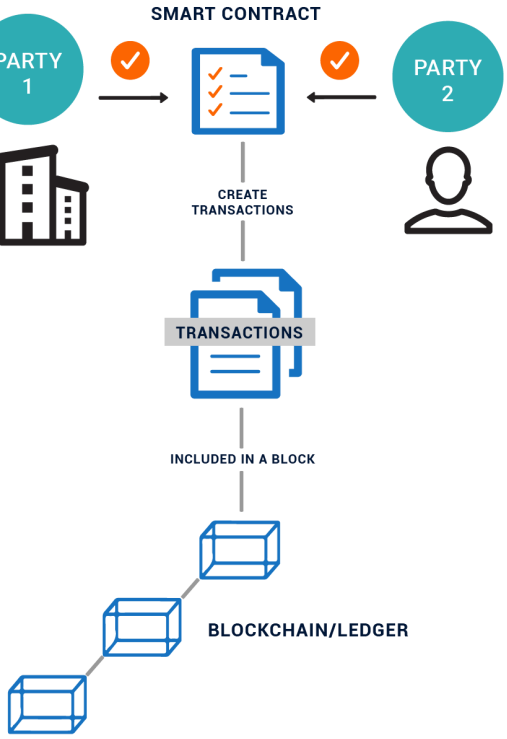

Figure 4 | Smart Contract Execution

2.1.5 Consensus Algorithms

The key purposes of consensus algorithms lie in ensuring synchronization and agreement between all the peers on a blockchain network, and preventing malicious peers from altering the distributed ledger data. There are various consensus algorithms that cater for different blockchain types and fault tolerance models, and they can be broken down as:

- **Lottery-based Consensus Algorithms**

  These are consensus algorithms that operate by choosing a winner peer to propose and transmit a block for validation to the rest of the network's peers using a lottery. They allow their network models to be highly scalable, at the cost of a longer time to ledger finality as these algorithms may lead to multiple ledger forks when multiple winners propose blocks and each fork must be resolved before finality is reached. Examples of such algorithms include:

  - **Proof of Work (PoW)**

    PoW is the most popular permissionless blockchain consensus algorithm, currently used by Bitcoin and Ethereum, that gives the lottery advantage to peers with the highest computational power on the network.

    The algorithm operates by giving network peers computationally challenging puzzles to be solved, a process commonly referred to as *mining*, in order to create new blocks of transactions. Such peers, known as *miners*, are given a share of tokens as incentives to abstain from malicious behavior. However, PoW consumes a huge amount of energy that is not environmentally sustainable for huge networks. It is also vulnerable to various attacks such as the Sybil attack, where a cluster of peers can control the network's ledger data when they take over half of the network's controlling power.

  - **Proof of Stake (PoS)**

    PoS is considered as a generalization of PoW that does not operate on incentives and gives the lottery advantage to peers with the highest stake of the network's assets. Instances of this algorithm include Proof of Deposit (PoD) where advantage is given to peers with the highest security deposit and Proof of Burn (PoB), where advantage is given to peers with the most burnt PoW-mined cryptocurrencies.

    The PoS algorithm randomly selects peers to validate and transmit blocks, with priority being given to peers with the highest stake. Ethereum are currently shifting towards using PoS.

- #### Proof of Elapsed Time (PoET)

  Similar to PoS, PoET combines a random peer selection algorithm with a first-come-first-serve method of choosing the lottery winner, referred to as the leader, to validate and transmit every new block to the ledger.

- ### Voting-based Consensus Algorithms

These consensus algorithms operate by allowing the majority of the network peers to validate transaction blocks, and if consensus is reached between them, finality is reached. They tend to be a lot more communication-intensive, as peers need to transfer messages to each other to reach voting consensus, which makes these algorithms less scalable than lottery-based ones, as shown in figure 5, however, they provide lower latency to finality and tend to be much faster.

- #### Kafka

  This algorithm operates by letting a single validator peer, also commonly referred to as an endorser, bundles proposed transactions to form a new block and transmits it to the rest of the network's peers. Consensus is reached when a specific minimum number of peers sign the block as valid and the block is committed to the ledger.

- #### Simplified Byzantine Fault Tolerance (SBFT)

  This algorithm operates similarly to the Kafka algorithm, however, it also takes into account the number of available peers on the network to prevent what's referred to as a Byzantine fault, where faulty peers are ones undergoing typical system failures and should be excluded from votings.

- #### Proof of Authority (PoA)

  This algorithm operates by declaring a group of peers as the network's authorities, and similarly to the Kafka algorithm, consensus is reached when a specific minimum number of authority peer sign the block as valid.

As previously illustrated, lottery-based consensus algorithms have the higher edge when it comes to scalability, but not ledger finality, however, when applied to permissionless networks, as shown in figure 5 for Bitcoin's PoW, the speed of the algorithms and their finality performance drop drastically.

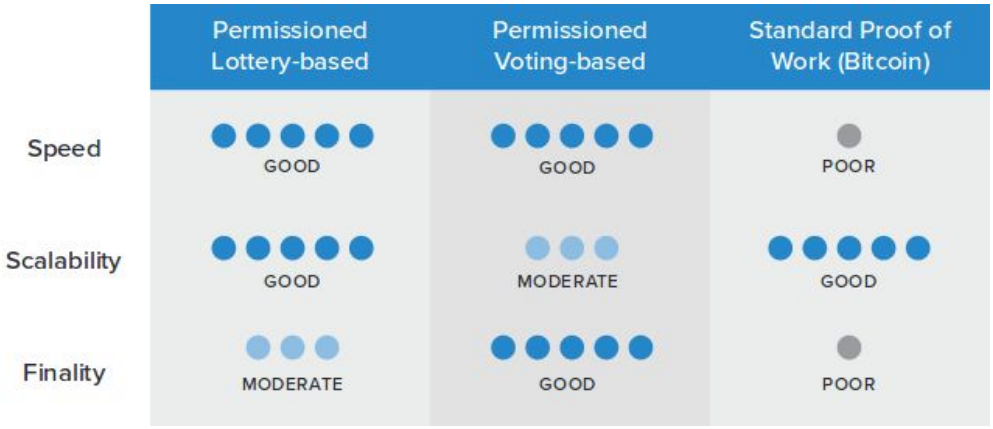

Figure 5 | Comparing Different Consensus Algorithms

2.2 Blockchain Platforms

- **BigChainDB**

  BigchainDB, often referenced as the only known hybrid blockchain architecture. It offers a typical blockchain layer, with varying access control lists and network protocols, that directs queries to a mongoDB database, granting it the blockchain benefits of decentralization, immutability and support for the creation and transfer of assets.

  It is an open-source platform, however, the code was not well-documented and maintained, and very few resources elaborated on application development using it.

- **Ethereum**

  Ethereum is a permissionless open-source blockchain platform. It currently depends on the PoW to achieve consensus, which affects negatively the performance of the blockchain network, as previously illustrated. The identity of the users on the network is not anonymous, since the identity of a user can be inferred through the ledger due to its transparency to all network users.

  Ethereum has its own cryptocurrency, known as Ether, which is used for payment when submitting transactions, known as gas, and for incentivizing network miners. Ethereum used smart contracts for their transactions, called Solidity, and run on the Ethereum Virtual Machine (EVM), which is used by all nodes on the network to maintain consensus.

Through our established research of consensus algorithms, permissionless blockchain architectures seemed unsuitable for the Curis Catena business model and use-cases, which favored a permissioned, or possibly hybrid, blockchain architecture.

- **Hyperledger Fabric**

Hyperledger is a linux foundation project umbrella that overlooks a consortium of companies, tools and frameworks offering open sourced blockchain services. Of those, the oldest in the project and the most well-maintained is Hyperledger Fabric, a permissioned blockchain platform, first released in July 2017, that operates using a modular approach, where one can use different consensus algorithms on the network that best suit its needs. It allows network entities to conduct transactions in separate private blockchain channels, which was specifically designed to cater for industry use-cases, including healthcare. Additionally, certificate authorities are a pluggable component in the network. Hyperledger Fabric has no inherent token, but one can make a cryptocurrency application using the platform if needed, and it uses smart contracts for transactions, referred to as chaincodes, that operate similarly to those in Ethereum.

Hyperledger Fabric is the platform chosen for the system, as it was the most applicable to the system's functionalities and the most well-maintained.

2.3 System Entities and Users

- **Certificate Authorities**

Entities that authenticate patients, healthcare facilities and providers to manage and control EPR access lists.

- **Healthcare Facilities (Super Administrators)**

Users in the healthcare facility who have higher privileges than system administrators, that include the ability to create and revoke system administrator accounts, in addition to system administrator functionalities.

- **Healthcare Providers**

Users in the healthcare facility that provide healthcare services such as: doctors, nurses and pharmacists, and who can be granted access to modify or view patients' EPRs upon authorization.

- **Patients**

  Users who require identity authentication to be registered on the system, view or modify their own EPRs, and add or revoke a healthcare facility's access to it.

- **System Administrators**

  Users in the healthcare facility who can add and remove healthcare providers from the facility's system, authorize providers to patient cases, and charge patients for healthcare services. Additionally, system administrators can take on the roles of:

  - **Endorsers**

    A blockchain network peer role responsible for simulating transactions and preventing unauthorized transactions from passing through the network, see the appendix transaction flow for further illustration.

  - **Committers**

    A blockchain network peer role responsible for appending validated transactions to the channel-specific ledger, see the appendix transaction flow for further illustration.

- **Orderers**

  Blockchain network entities in the Hyperledger Fabric framework that form the ordering service. The role of ordering service nodes (OSNs) is the ordering of transactions into blocks according to the network's chosen consensus implementation before delivering them to the appropriate channel, see the appendix transaction flow for further illustration. The ordering service can be one of the following:

  - **SOLO**

    Default ordering mechanism that uses a single ordering node, that will be used in the current development stages of the system.

  - **Kafka**

    Recommended ordering mechanism for production use using Hyperledger Fabric. It uses the Apache Kafka open source stream processing platform.

- **SBFT**

  Byzantine fault tolerant ordering mechanism that is soon to be released in Hyperledger Fabric.

- **Databases**

  Cloud-based or on premise databases in a healthcare facility responsible for the storage of its own patients' data, which represents the subset of each patient's medical history and records that are affiliated with this healthcare facility only.

2.4 HIPAA Compliance

Curis Catena system is intended to be HIPAA compliant. It is mainly concerned with the requirements of the HIPAA security and privacy rules.

The required specification implementation followed for the HIPAA security rule:

- Unique user Identification, username or ID, to be able to track users
- Requirement of strong passwords for account creation
- Implementing access to EPRs in emergency cases
- Encrypting EPRs before storage in healthcare facility databases and decrypting them for use
- Implementing access control lists for integrity control to ensure that EPRs are not accessed or modified in any unauthorized manner
- Recording and examining system activities for audit control
- Authenticating all system users to validate their access control rights
- Logging out users automatically after a certain time of inactivity

The required specification implementation followed for the HIPAA privacy rule:

- Giving patients the right to view their EPRs and control who could access them
- Not granting any users the right to delete or close any EPRs

3. System Architecture Overview

The Curis Catena system can be decomposed into the following four main subsystems, as shown in figure 1.

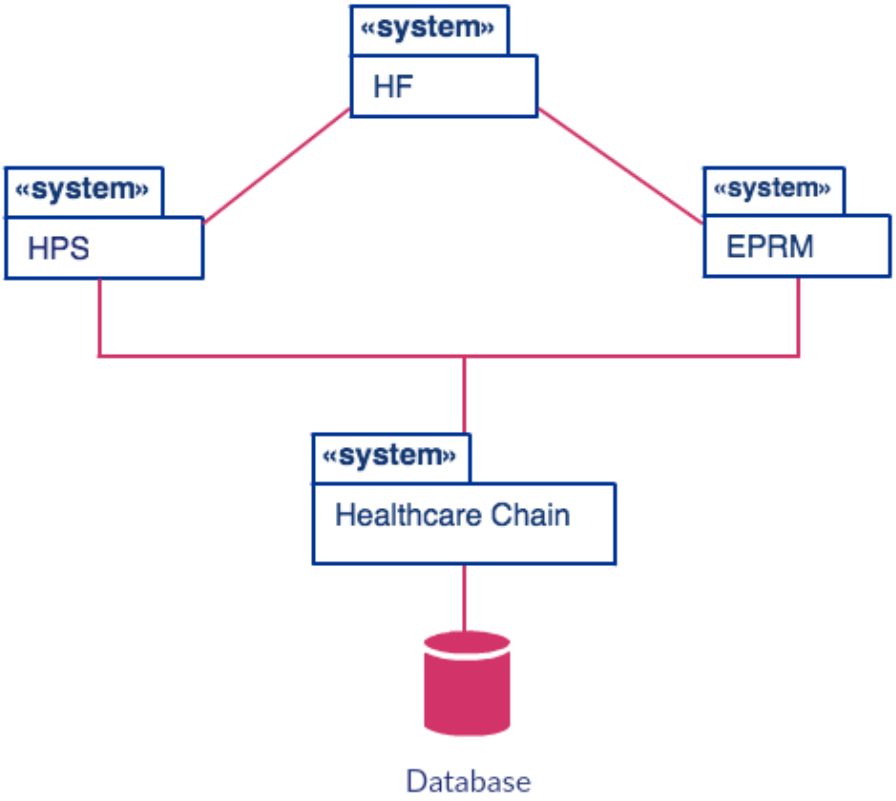

Figure 6 | System Overview

3.1 Electronic Patient Record Management System

The EPRM system includes features related to the patient's interactions with the system's patient web application interface and certificate authorities for authentication.

3.2 Healthcare Facility System

The HF system includes features related to the system administrators' and the healthcare facility's interactions with the system. It is the only connection between the HPS and EPRM system, to ensure that all healthcare providers on the system are authenticated.

3.3 Healthcare Provider Services System

The HPS system includes features related to the healthcare provider's interactions with the system's healthcare provider web application interface, as well as their interactions with their healthcare facilities and certificate authorities for authentication.

3.4 Healthcare Chain System

Both HPS and EPRM have access to stored EPRs through the Healthcare Chain system, which is a blockchain-based system that connects transactions to the healthcare facility's database through pointers. The Healthcare Chain system is implemented using the Hyperledger Fabric framework, where each of its components serves a certain functionality aspect, and some could be replaced by our own implementation. See the appendix transaction flow for further illustration of the functionalities of this system.

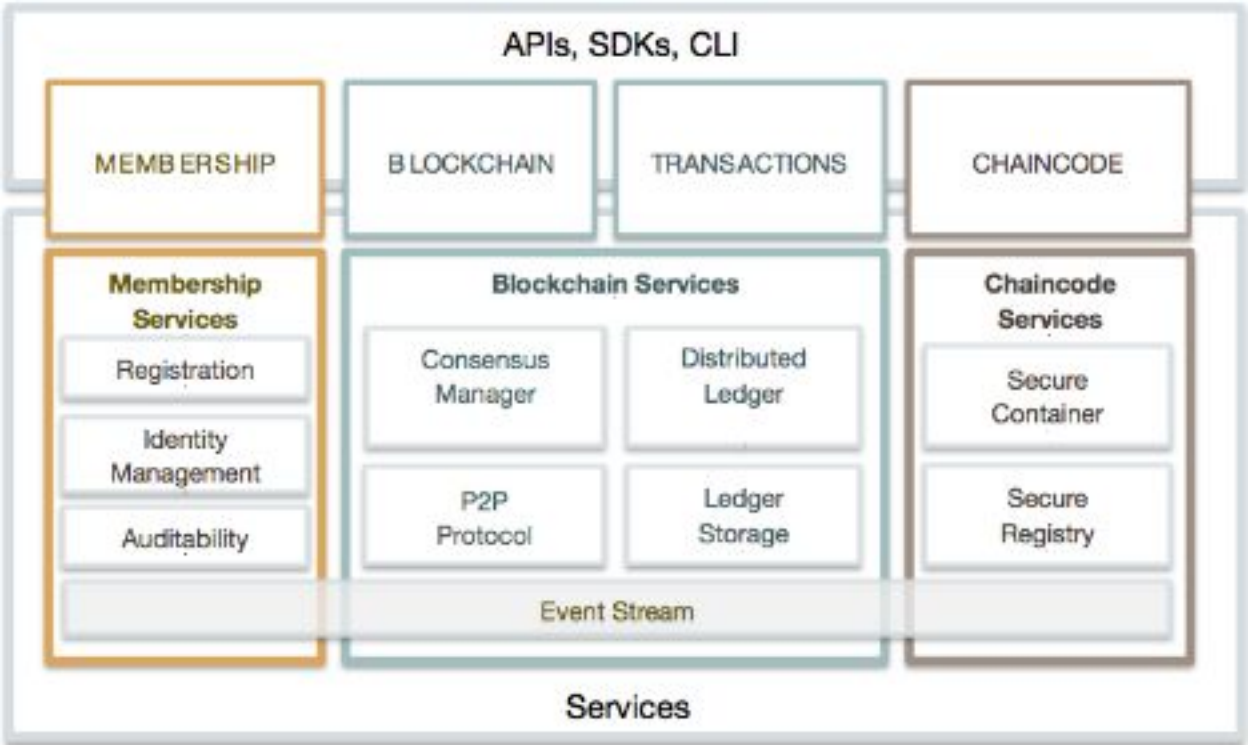

Figure 7 | Healthcare Chain System's Hyperledger Fabric Layers

The most significant sub-layers of this system include:

1. Application Layer

It uses the APIs, SDKs and CLIs from the framework to link the framework to our application interfaces.

2. Distributed Ledger Layer

It contains a world state which is basically the state of the ledger at a specific time, and it contains the transaction log which contains all the transactions that resulted in the values that are found in the world state. The world state is the system's database, and it can be implemented in LevelDB or CouchDB.
The world state transactions contain the signatures of the endorsing peers. In order to make sure that the blocks getting added to the chain maintain the integrity of the data, before appending any block, a check is made in order to make sure that the values contained within the assets that were previously read have not changed since the execution of the chaincode.
Since Hyperledger Fabric allows the network to have multiple channels where each have their own ledger to ensure its privacy, the ledger of such channels has a configuration block at the beginning of it. It specifies the policy that the channel follows, who is allowed in, and other information. The configuration of the channel can be updated later but it would need to be done by an authorized user. Each channel contains membership service provider instance that authorizes and authenticates the user on the network.

3. Chaincode Layer

It is the business logic behind the policies that are implemented. The chaincode layer is responsible for processing transaction requests and validating them. Chaincode defines an asset, and the policies on how to modify the asset.
When a chaincode invocation happens, it gets executed against the world state through a transaction proposal. This execution outputs a write set that, if validated, gets broadcasted to the network where all the peers can then apply that information.
Since Hyperledger Fabric is modular, the execution of chaincode gets done in parallel with the ordering service in order to increase the efficiency of the network.

4. Event Stream Layer

It is the one responsible for the notifications that are getting sent whether by the blockchain layer or the chaincode layer.

5. Consensus Layer

It is the algorithm responsible for the common agreement that all people reach in order to determine which transactions should be added on the blockchain. This agreement is decided when the block's order and the values written inside the transactions match the policy checks.
During the lifecycle of a transaction, multiple checks happen. For example, a check must be done to ensure that the number of allotted endorsers is met on a certain

transaction. Also, there must be system chaincodes present to make sure that policies are met.
Before committing a transaction, the participating nodes utilize the system chaincode to make sure that the correct number of endorsers is present and that these endorser signatures are verified.
Another check gets done to ensure that the world state of the blockchain is agreeable before committing a new block upon it.

6. Membership Layer

It is the one responsible for authenticating and authorizing users on the network.
The membership layer contains a Membership Service Provider (MSP) component. We can make multiple MSP instances, and they identify which root certificate authorities and intermediate certificate authorities are authorized to distribute identities to the members of an organization.
MSPs can also help us by defining the role of the person in the network, like for example the administrator role.

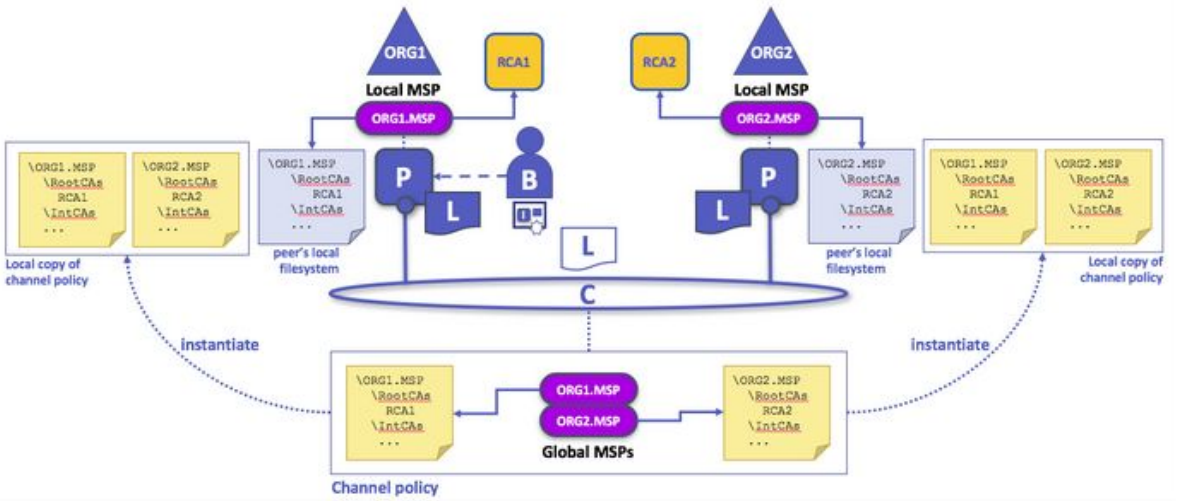

Figure 8 | MSP Demonstration

In figure 8, we can see that every peer on the network has their own local MSP and there is only local MSP per node. Local MSPs are only defined on the peer's local filesystem; however, in the channel MSP, every node participating in the channel gets a copy (instance) of the channel policy. The channel MSPs are synchronized according to the consensus agreed upon by the participating peers.
Also, every organization must have its own MSP. As we can see in figure 8, org1 has ORG1.MSP, and org2 has ORG2.MSP. Both of them have their own different root certificate authorities though that might not always be the case. If an organization wishes to join a channel the MSP responsible for that organization must be present in the channel's policy, if not, the transactions will not be accepted from that organization.

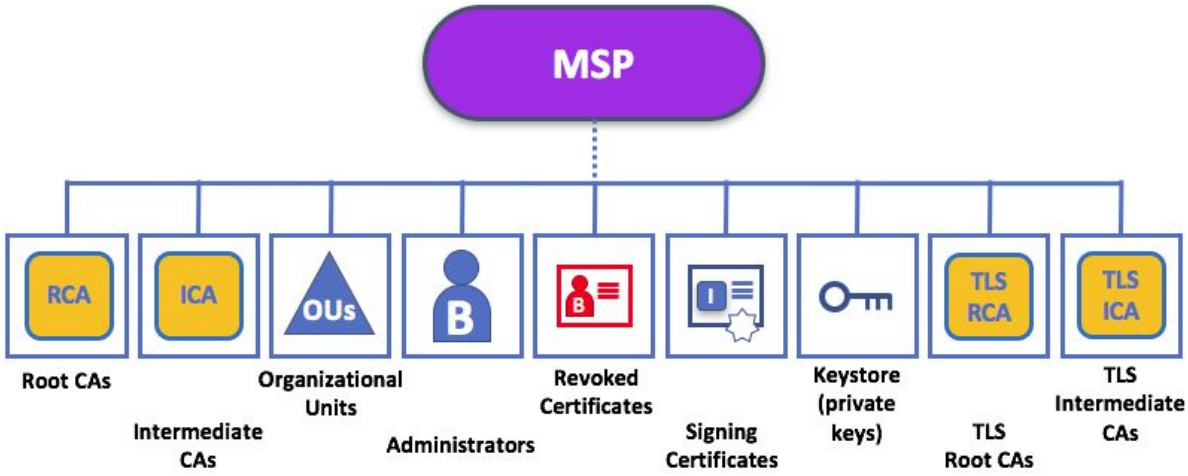

Figure 9 | MSP Structure

Every MSP must have a certain structure consisting of different folders, which are:

- **Root CAs Folder:**
  It consists of a list of certificates that are issued by the trusted Root CAs representing an organization.

- **Intermediate CAs Folder:**
  It consists of a list of the certificates that are issued by the trusted intermediate CAs representing an organization. Every certificate must be traced back to the root CA.

- **Organizational Units Folder:**
  It consists of a list of the organizational units in an organization. Members coming from a certain organization unit can be restricted from accessing the network.

- **Administrators Folder:**
  It consists of a list of the certificates of the nodes in the network who are defined as administrators.

- **Revoked Certificates Folder:**
  It consists of a list of identifying information about an actor that has been revoked from the network.

- **Node Identity Folder:**
  It contains the identity of the node, which includes the certificate.

- **Keystore**:
  It contains the signing key of the node, where it gets defined on the local level.

- **TLS Root CAs:**
  It consists of a list of certificates that are issued by the trusted Root CAs representing an organization for TLS communications.

- **TLS Intermediate CAs:**
  It consists of a list of the certificates that are issued by the trusted intermediate CAs representing an organization for TLS communications. Every certificate must be traced back to the root CA.

4. Human Interface Design

The tentative application currently developed for the system has several components contributing to its initial look.

The configuration files component works in the background to form the infrastructure of the blockchain network, while the web interface component showcases the interface that will be shown to the healthcare provider.

4.1 Configuration Files

```
OrdererOrgs:
  - Name: Orderer
    Domain: example.com
    Specs:
      - Hostname: orderer
PeerOrgs:
  - Name: Org1
    Domain: org1.example.com
    Template:
      Count: 1
      # Start: 5
      # Hostname: {{.Prefix}}{{.Index}}
    Users:
      Count: 1
```

Figure 10 | crypto-config.yaml

In figure 10, this file crypto-config contains the configuration information needed to generate the certificates of the organizations in the system. We defined our system to have only one organization (the hospital), and the orderer organization.

```
Profiles:
    OneOrgOrdererGenesis:
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
    OneOrgChannel:
        Consortium: SampleConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
Organizations:
    - &OrdererOrg
        Name: OrdererOrg
        ID: OrdererMSP
        MSPDir: crypto-config/ordererOrganizations/example.com/msp
    - &Org1
        Name: Org1MSP
        ID: Org1MSP
        MSPDir: crypto-config/peerOrganizations/org1.example.com/msp
        AnchorPeers:
            - Host: peer0.org1.example.com
              Port: 7051

Orderer: &OrdererDefaults
    OrdererType: solo
    Addresses:
        - orderer.example.com:7050
    BatchTimeout: 2s
    BatchSize:
        MaxMessageCount: 10
        AbsoluteMaxBytes: 99 MB
        PreferredMaxBytes: 512 KB
    Kafka:
        Brokers:
            - 127.0.0.1:9092
    Organizations:
Application: &ApplicationDefaults
```

Figure 11 | configtx.yaml

In figure 11, the configtx.yaml file generates the orderer's genesis block, the channel configuration transaction.

4.2 Web Interface

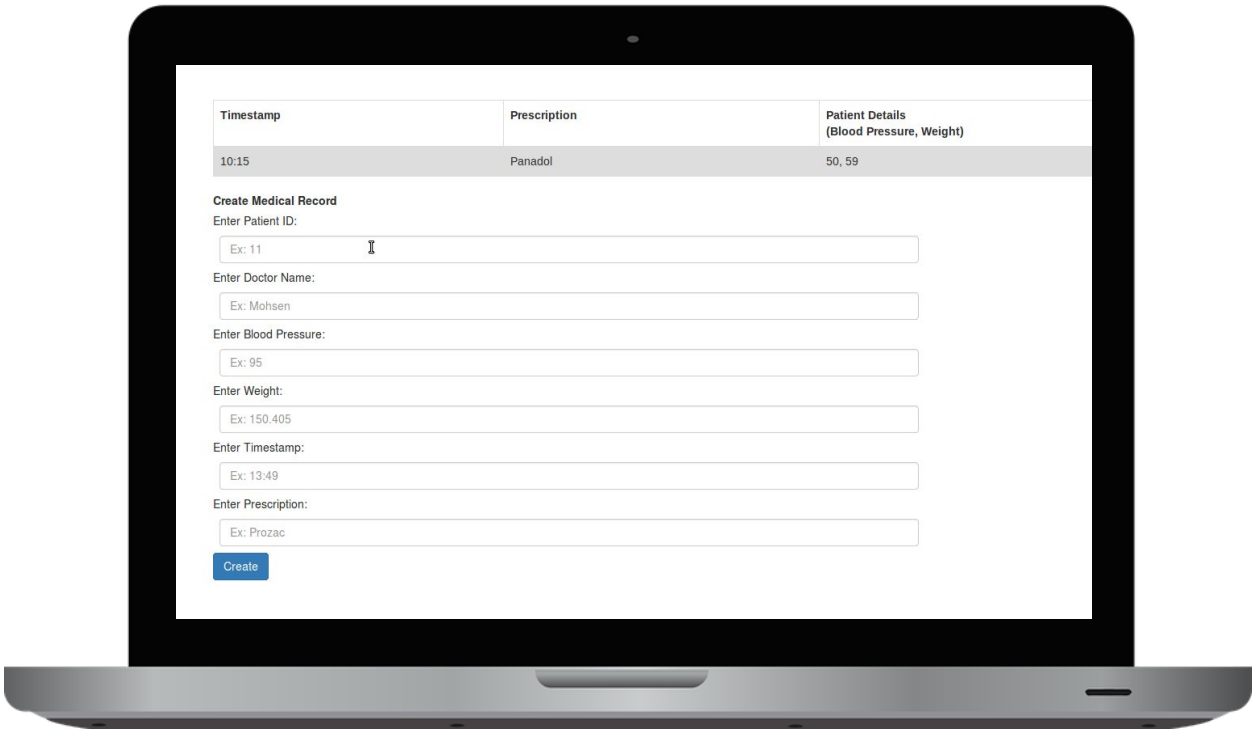

Figure 12 | Creating Medical Record

In figure 12, the healthcare provider can:

- Insert a patient ID and have the system return an error if the ID is already taken.
- Insert the doctor's name.
- Insert health-related information about the patient such as: blood pressure, weight, temperature.
- Insert the prescription that the doctor prescribed to the patient.
- Create a medical record that includes all the aforementioned information.

Figure 13 | Querying Medical Record

In figure 13, the healthcare provider can:

- Query a specific health record by entering the patient's unique ID.
- Query all existing medical records on the network that the health provider is authorized to see.

5. Project Timeline

In figure 14 below, the timeline for the system development progress this semester, along with the planned milestones for coming months are listed.

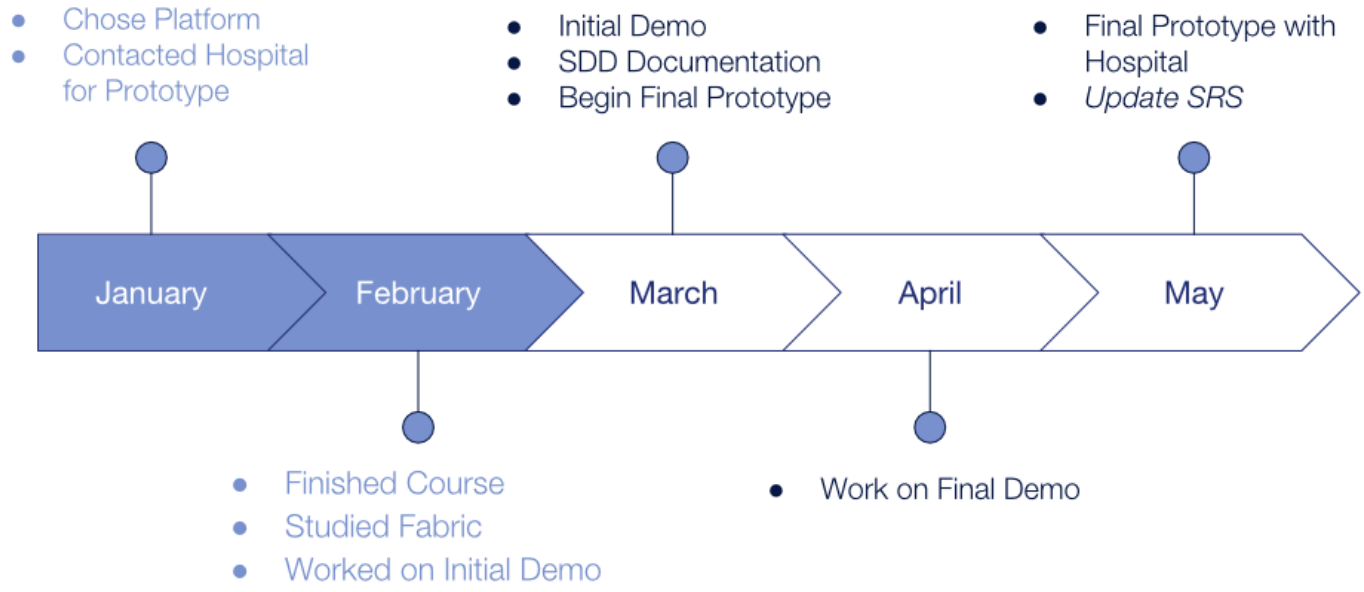

Figure 14 | Progress Timeline

Appendix: Transaction Flow Sequence Diagram

Any transaction taking place on the Healthcare Chain system undergoes the following stages:

1. **Transaction Endorsement**
2. **Ordering Service**
3. **Validation and Commitment**

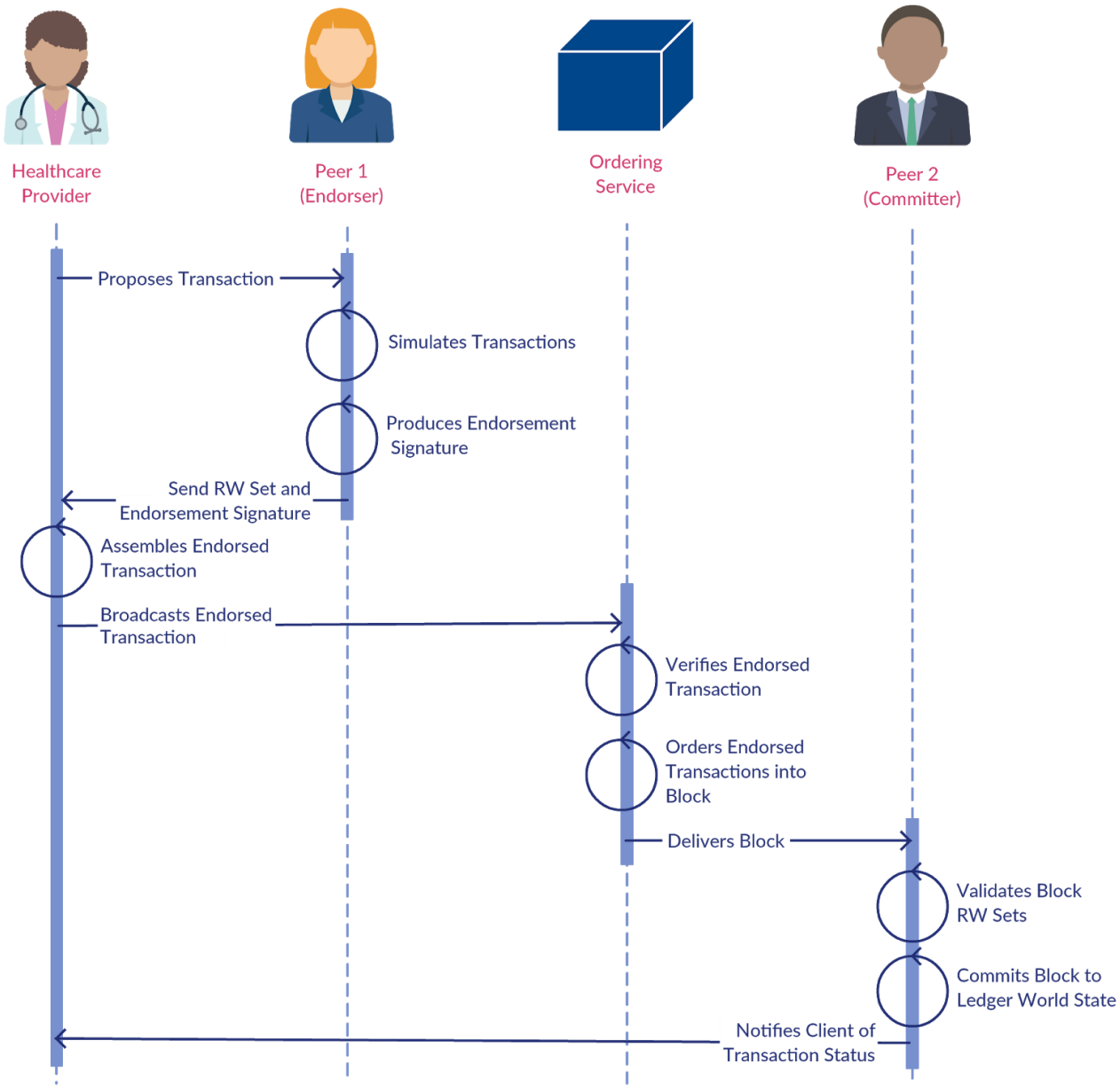

Figure 15 | Transaction Flow Sequence Diagram