SECURE EXCHANGING OF ELECTRONIC HEALTH RECORDS (EHRs) USING TRUST BASED BLOCKCHAIN NETWORK

Bachelor of Technology in Computer Science and Engineering

Submitted by

Rahul Parihar (167150)
R.V.S.Vyshnavi (167154)
Pinninti Satyateja (167147)

Under the guidance of
Dr. E Suresh Babu
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY,WARANGAL-506004

CERTIFICATE

This is to certify that this project report entitled "**SECURE EXCHANGING OF ELECTRONIC HEALTH RECORDS (EHR) USING TRUST BASED BLOCKCHAIN NETWORK**" by **Rahul Parihar (167150), R.V.S.Vyshnavi (167154), Pinninti Satyateja (167147)** submitted in partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering of The National Institute Of Technology Warangal, during the academic year 2019-2020, is a bonafide record of work carried out under my guidance and supervision.

Dr. E Suresh Babu
(Assistant Professor)
Department of Computer Science And Engineering

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide **Dr. E Suresh Babu** for giving us the opportunity and guidance to work on this topic. It would never be possible for us to take this project to this level without his help and support and his innovative ideas and encouragement.

CONTENTS

ABSTRACT

Health records are classified information about the healthcare of the individuals and are needed to be shared to relevant people such as doctors, pharmacists and researchers . Sharing the health records sometimes can cause issues which violate the privacy of the patients and hence utmost care should be taken while sharing these documents with others. Blockchain as a technology provides ways in which the transactions can be monitored and applications can be built with trust, transparency and proper accountability. This can be used to develop a decentralized system to share the EHR data with the hospitals , doctors , pharmaceutical companies , researchers and other interested parties. Using blockchain we propose a solution to share , store and manage EHR data. The proposed work will reduce the single point failure in case of centralized databases and will provide privacy , security , accountability and greater control over the health data to the patients.

The proposed solution leverages the hyperledger fabric a permissioned blockchain technology to establish a secured and trusted network where all the stakeholders of the medical treatment field come together and share data as well as the private health records of the patient are well secured with a proper access control mechanism. The members participating in the network are authorised by the admin of the network and hence the privacy and security of the chaincode is always maintained. The network can be scaled quickly for large numbers of people and the business logic of the network ensures that minimum collision takes place between various organisations in the network.

The application that results from this provides a user friendly platform with a highly secure blockchain network so that all the basic activities can be imitated and by all the stakeholder and the entire supply chain can be traced so that the accountability increases and the loss of crucial documents as well as unauthorised sharing of the documents can be limited.

INTRODUCTION

Overview of HealthCare :

Health Care involves improvement and maintenance of health using diagnosis followed by treatment and cures any kind of impairments in people.Healthcare industry provides services to patients by treating them with utmost care.The modern health care sector is divided into many sub-sectors which involves trained professionals to meet health needs of individuals.The healthcare industry includes establishments which ranges from private practices of physicians in small villages, who have only one medical assistant to busy multispeciality hospitals in cities that provide thousands of different jobs. Healthcare industry needs constant innovation under enhanced rules as it makes practitioners face many challenges everyday.

Increasing population and widespread long-term diseases are the main reasons which makes demand for development of medical practices used in the healthcare industry. In the future,there will be great demand for products used in medical procedures.

Overview of Electronic Health Records :

An electronic health record (EHR) is a digital copy which includes complete information about a patient’s health. EHRs’ are important , highly sensitive and classified information about any patient's healthcare. They are needed to be shared if required to relevant people such as doctors, hospitals, pharmacists, researchers and pharmaceutical companies.EHRs are patient-centric and

provides information securely to the users who are authorized.One of the main features of these records is that health information can be created and organized by approved providers in a digital format which can be shared with other providers across many other health care organizations such as laboratories, researchers, pharmacies,health insurance companies.

Advantages of Electronic Health Records :

EHRs' have the capacity of exchanging health information and helps healthcare providers in providing better standards and secured supervision for patients.They help practitioners to provide finer medical care by:

1) Giving recent and entire information about medical history of patient
2) Allowing fast access to records for more systematic care to patients
3) Sharing health information securely with patients and other practitioners
4) Aiding providers to identify problems in patient effectively
5) Enhancing interaction between patient and healthcare provider
6) Providing supervised and genuine prescription
7) Helping to promote clear and entire documentation along with precise billing
8) Improving privacy for health information of patients
9) Assisting healthcare providers increase efficiency and balance in life
10) Being cost effective by reducing need for written documents and replication of checking, thereby increasing safety and providing good health.

Problems with existing Electronic Health Records :

As EHRs contain critical and sensitive information,maintaining access control is a difficult problem as it requires consent forms and all documents to provide the right to view health records.For diseases which pose a threat to life, it becomes necessary to keep the health record up-to-date. Now modification of the health records and addition of the new records could be tiresome and keeping all other peers updated with the new information not only makes it costly but also unfeasible sometimes. In case of diseases which require all the medical history to be analyzed before starting with the medication might be problematic as sometimes all the records are not in the same place.

Another threat that comes out in traditional ways is that there is a possibility of getting the

patient's right to privacy violated as institutions might share the healthcare data of the patient with research institutions without his consent. There is no proper control of the patient over his own data and can't control how his data is being used.In case of institutions the data is present at one place in a centralized manner providing a high possibility for single point failure.All the data at once place means the accessing of the data will be slow and the performance of the system will depend on that.

In his lifetime a patient might visit multiple health institutions and hence his records are scattered around with all those institutions. So if required to share the data present with one institution to another most of the time it requires signing consent form and also the extent upto which the data will be shared is also controlled by the institutions and hence all this delay and lack of proper medical history could result in undesired situations.

The sharing of the health data poses another level of threat as the traditional way of sharing data is either time consuming or not secure. While the health data is sent by post from one institution to another ,it takes lots of time and is generally avoided. Sending the data using some electronic media such as email might be considered a solution. But while being in transit there is a high chance of data being stolen. Even if all the data is stored at a single place then there are chances that all the opinions of all the institutions might come to be the same and might affect the proper treatment of the patient.

Overview of Blockchain Technology :

Blockchain is the solution to all the problems mentioned above. It provides a ledger which is trusted, secure, transparent and the history of storing and accessing data is transparent. The real control of the data can be given to the patient and he becomes the one to decide how the data is being used. Blockchain technology uses transactions and in this scenario the uploading, transferring and creation of Electronic health records are transactions happening in the blockchain.Blockchain in managing healthcare data has raised a lot of opportunities in this field and there is a need to devise a solution which works using the blockchain technology to store encrypted data of the patients and stores it in a decentralized manner.

Characteristics of Blockchain Technology :

1) **Transparency :** There will be a copy of ledger with each and every node present in the network.Any new transaction can be added to ledger only when majority of nodes accept it to be valid. In this way,transparency can be ensured.

2) **Enhanced Security :** As the need for central authority is eliminated,security can be enhanced since there is no possibility that only one person can change any characteristics of the network for their benefit.Using encryption, another layer of security can be added.
3) **Decentralized technology :** It is decentralized because there will not be any central authority that governs the entire network.
4) **Consensus :** Blockchain technology has consensus mechanisms which includes algorithms that help the network to make decisions.
5) **Distributed ledgers :** Ledgers will be present with all the users of the system distributing computational power across the computers to provide a good outcome.

Types of Blockchain :

1.Public Blockchain : Public blockchain,which can also be called permissionless, can be accessed by anyone who can sign in using the internet,to be a part of the blockchain network to become an authorized node.Any user who is present in the public blockchain,though anonymous, is authorized to access records and verify transactions to do mining.

Example: Bitcoin, Ethereum.

Disadvantages of Public Blockchain :

- As there is no need for the users to prove identity while using blockchain,this can cause difficulties as it helps people to carry out illegal activities.
- It uses consensus algorithms like Proof of work,in which users need to work out some difficult mathematical problems.But this is a costly one as it uses many resources for computation.
- It offers a block size which is a limited one.

As there are many disadvantages, a Permissionless blockchain like Ethereum is trying to use Proof of stake as a consensus mechanism instead of proof of work.

2.Private Blockchain :A private blockchain which is also termed as permission blockchain, can be used in a closed network like within particular organizations in which only required members can be part of the network.

Example:Hyperledger,Corda.

Advantages of Permissioned Blockchain :

- Decentralized platforms can be provided by private blockchain in which the information will be fragmented and stored at different places but not in a centralized one by making it accessible to users at any time.
- Immutable signatures will be provided for all records in this network.Transactions performed and data that is exchanged will be encrypted which ensures security of the entire system.
- Permissioned blockchains give more importance to immutability and efficiency over anonymity and transparency.
- Transactions and their validations will be easy and take place fast because this blockchain eliminates the necessity of a third party.

3.Consortium Blockchain : In this blockchain the network will be controlled by more than one organization.It is also called as semi decentralized type.Information can be exchanged and mining can be done in this network where more than one organization can act as a node.Government Organizations , Banks use this blockchain network.

Example :Energy Web Foundation, R3.

4.Hybrid Blockchain : Hybrid blockchain uses the features of both public and private blockchains, like we can have a permission based system which is a private one and a permission less system which is a public one.There will be some part of data that can be shared to public where the remaining part can be kept private.This system helps users to join in private blockchain using many public blockchains.

Example:Dragonchain.

Overview of Electronic Health Records using Blockchain :

Current advancement in technology is affecting people and changing their course of action of using things. There have been changes in technology used in the healthcare industry as well.Advantages of this recent technology includes improvement of security,user experience and other factors which will be provided by Electronic Health Record (EHR) and Electronic Medical Record (EMR) systems.Although there are some problems in providing security of health information,ownership of medical data etc.These issues can be solved using Blockchain Technology.It provides a platform which cannot be tampered and secure place for storing any information about health.

Before the advancement in technology,the healthcare industry used to store medical records in papers which were not efficient, secure, organized and tamper-proof. There are also some issues like replication of data and redundancy as there will be many records of patients in every institution.EHR systems have been used by many hospitals around the world because of its advantages.They include complete information ranging from patients' appointment management to billing and lab tests.This paper aims to provide security,privacy while sharing health records and presents a flexible and easy-to-use blockchain platform that can be used by every individual.

Rest of Thesis :

1. Problem Statement
2. Objectives
3. Literature Survey
4. Architecture
5. Future Work
6. References

PROBLEM STATEMENT

To model a distributed and trust based blockchain network for secure exchanging of **Electronic Health Records(EHRs)**

OBJECTIVES

- ❑ To setup a Hyperledger Fabric Blockchain Network with clients and Healthcare providers
- ❑ To identify and authenticate the clients and Healthcare providers
- ❑ To store the Health records in the Blockchain in a distributed manner
- ❑ To securely share health records of clients with the Healthcare providers
- ❑ To ensure the privacy of stored EHRs
- ❑ Securing client data

LITERATURE SURVEY

1. Using Blockchain to Address Interoperability Concerns in Healthcare

Governments and Health care Authorities are undertaking several initiatives to boost the health IT system . Electronic Health Records is one such initiative and the next step is to ensure a seamless flow of health information across stakeholders that will enable better decision making. The ability of health IT systems to share this information and use that information for better decision making is called **Interoperability**. Absence of interoperability creates several issues like restricted data sharing and lack of meaningful insights i.e, benefits of data mining are restricted by unavailability of complete data.
Several challenges have to be faced while enabling EHR Interoperability like data security and integrity , government regulations etc..
Blockchain Technology is considered to be one of the frontrunners in the race to resolve interoperability issues in EHRs. Blockchain features can help make interoperability a reality.
If a new platform is to be adapted by all the stakeholders , it needs more than one organisation to show up . Apart from technological challenges there are several issues to be overcome . All the stakeholders should adopt a new common technology platform , lot of investment in new technology is required.

2. Understanding Blockchain : Opportunities in Health care

This focus paper described how Blockchain works and highlighted real world applications such as interoperability , security and smart contracts in Health care systems . All the transactions in health care require verification , eligibility . Blockchain is a technology that verifies trustworthiness and makes it difficult to manipulate the transaction.
The IT systems in health care must be simple enough to share and read the transactions i.e, interoperability. According to Dr. John Halamka, Chief Information Officer at Boston Medical Center there are three models of interoperability among medical data. **Push** is used for sending health information to providers.**Pull** is used to query information from a provider. **View** can be used in which one can see information inside another's record. A blockchain based EHR could unlock the conditions of interoperability and security.

3. **Privacy preserving framework for access control and interoperability of electronic health records using blockchain technology**

Security issues are growing in EHRs which need to be taken care of. Obviously Blockchain is a good opportunity to provide security to these EHRs. This paper proposes Ancile framework and its unique type of smart contracts for operation.Ancile contains three principal units namely Ethereum-go client,database manager and cipher manager.Framework architecture for several operations and the steps involved in it are mentioned.Ancile uses Ethereum tools for making a system that uses blockchain technology and is cost effective.

4. **A Systematic Review of the Use of Blockchain in Healthcare Industry**

This paper provided various fundamental concepts of blockchain,consensus mechanisms , how blockchain was introduced to the public and reviewed research questions and selection of studies and references. Several research questions are discussed and answered like upto what extent research has been done , current research trends and future work of blockchain technology.

The fundamental tasks in the system involve getting connected to the network,updating the ledger,verifying transactions and forwarding only valid transactions to the network and making new blocks whenever required.To transform healthcare the increase in accuracy, management of data should be focussed much. The data in health records need not be modified or deleted , the immutable property of blockchain satisfies this condition.

5. **Exploring applications of blockchain in securing health records.**

Inorder to prevent attacks,security of the EHRs has to be ensured.Blockchain is one of the secured way of ensuring the security of the records.The shift towards EHRs is a method of replacing paper charts.

Blockchain can facilitate and maintain records safely and can do tamper-proof transactions but can't ensure privacy innately. Several models are proposed to ensure privacy measures such as the Mooti model, Enigma model.Blockchain has a particular way of storing health information which ensures safety and security for the data.

6. **Electronic Health Records using Blockchain Technology**

This article mentions how smart contracts will be helpful in dealing with integrity,management and interoperability of the data.It focuses mainly on data privacy which gives ideas about who are authorized to access data and data accessibility which gives the extent to which data is available.

Main issues that need to be focussed on are :

i) Interoperability between participants of the network.
ii) Data privacy and security
iii) Data format standardization is an additional challenge.

7. **Electronic Health Record Storage using Blockchain**

Current EHR systems use centralized databases in which medical data will be stored at a single place which increases the possibility of risks.Security and Integrity cannot be ensured by such databases.Application considers three types of users: Doctors, Patients and Researchers.When a user invokes a query,the current state of the blockchain will be evaluated and submitted by SDK.Requests will be sent to other peers by the blockchain for getting agreement.If majority of peers validates it,then the transaction will be added to the ledger after which a key-value pair will be generated.

8. **Design of a BlockChain based system to facilitate sharing of healthcare data**

Blockchain technology ensures secure sharing of data without any intermediaries.Smart Contracts consists of a logic which provides accessibility to patient's health data to right person at right place by ensuring safe and secure sharing of patient's health records. This paper provides information about how smart contracts are used in blockchain technology for secure sharing of data.

9. **Implementation of EHR system - a blockchain-based application**

Electronic Health Records (EHR's) requires to ensure data privacy,security,data integrity,interoperability between patients and healthcare providers.All the challenges presented above can be achieved using Blockchain which is immutable,decentralized and consensual. Challenges that are presented by using this technology for EHR systems are :

i) Scalability and Performance
ii) Usability
iii) Secure Identification

10. Electronic Health Records Using Blockchain Technology for storage and retrieval of medical records

The main concerns with some EHR systems is that having a centralized database which increases the possibility of having a single point failure.Moreover secure sharing of health records owned by a patient to other healthcare providers is also a difficult task.Such security issues needs to resolved and a decentralized architecture is required.To solve above mentioned issues,EHRs needs to implemented using blockchain technology.
In this article , the proposed architecture of the EHR system will be of three layers in which the first layer i.e front end is made with HTML,CSS,JS . Second layer with Hyperledger Fabric , permissioned blockchain and Third layer i.e back end with Blockchain , Database.

11. On the design of Blockchain-based system to facilitate Healthcare Data Sharing

Sharing of health data raises privacy and security concerns which needs to be taken care of for having a better use of EHR systems and blockchain technology can be a solution for this.Though it has been developed for having financial ledgers, its applications got expanded and can be used in the healthcare industry as well.This paper mentioned how blockchain technology can facilitate following using a blockchain based architecture design :
(i) Ensuring privacy while sharing health care data
(ii)Providing security for stored data

12. Applications of Blockchain in Healthcare Sector:Current Landscape & Challenges

The problems that are raised in the healthcare industry involve the following like not all medical treatments will be reported,fake medicines are getting produced etc..Blockchain has the capability of solving all the issues by providing trust though it eliminates need for a third party and hence this technology can have a wide range of applications in healthcare industry.

In this research paper, some of the most important use cases of blockchain technology in the health care sector like management of health information,research,management of prescriptions provided by doctors,billings are reviewed.

13. Blockchain based EHRs for Regulated Healthcare Jurisdictions.

Smart Contracts present in blockchain will be used to ensure privacy and integrity for medical data.The process of achieving privacy and security is explained in this paper using a mechanism.

Blockchain has a specific feature of providing decentralized system by which many advantages can be presented in its applications,some of which are mentioned below :

- Transactions once recorded cannot be changed.
- Capability to recover from difficult situations thus not affecting the stored data.
- Blockchain contains smart contracts which are used to control the unauthorized access to health information that is stored.

S. No.	Aim Or Title of Article	Description or Features	Technology Used	Advantages	Disadvantages	Implementation	Remarks
1.	Using Blockchain to Address Interoperability Concerns in Healthcare A survey	Provides the issues created by absence of Interoperability and challenges enabling it . Issues : Restricted data usage mainly Challenges : data sharing is difficult as suppliers and buyers work independently Barriers to adopt blockchain for interoperability : Scalability , Miners power etc These can be overcome using Tangle Network	-	Blockchain can make interoperability a Reality due to Disintermediation , Immutability of transactions , No single point of failure , Validation of transactions	Storage related Issues are to be considered	NO	The technology required to implement an EHR system exists, but its widespread adoption is a cultural challenge.

2.	Understanding Blockchain : Opportunities in healthcare	This paper explores how blockchain works and introduces the concept of smart contracts .	-	Blockchain is decentralized and scalable.	Latency and storage related issues must be taken care of.	NO	Although Blockchain based systems meet requirements of the healthcare industry it would take time for this industry to adapt blockchain technology. To make this change Government intervention is necessary.
3.	Privacy preserving framework for access control and interoperability of EHRs using blockchain technology	Provides details about the proposed Ancile Framework and the Smart Contracts for operations and Framework Architecture demonstrating how framework is used in various situations	Go-Ethereum Ancile Framework	Scalability can be promoted by storing small records and using hash values for large records	-	NO	-
4.	Systematic Review of the Use of Blockchain in Healthcare Industry	Depicts the need of Blockchain in Healthcare and reviews various references of the blockchain concepts and provided fundamentals of Blockchain technology.	-	Several research questions are discussed and answered.	This technology is a new one and requires testing	NO	-

5.	Exploring applications of blockchain in securing health records.	Explores security related issues of EMRs and various blockchain EMR models.(Mooti model , enigma model etc)	-	Blockchain provides tamper proof transactions and helps in secure maintenance of health records	Even by using Blockchain, some privacy concerns will be raised for which extra measures need to be taken.	NO	Caution should be taken while using any new technology, especially when valuable information is at stake.
6.	Electronic health records using Blockchain	Includes how interoperability and privacy concerns can be handled in EHRs using smart contracts of blockchain technology	-	This technology provides trust by securely sharing data even though there will not be any third party for verification	Solutions for latency and storage requirements should be provided for blockchain.	NO	Blockchain and smart contracts help the system to meet its requirements like privacy , data accessibility , ethical issues and openness.
7.	To Implement secure, immutable, decentralized EHR system	Detailed Report on EHRStorage using BlockChain Technology	Blockchain with smart contract written in permissioned blockchain, Hyperledger Fabric	The cost can be considered reasonable as this blockchain technology provides maximum data privacy Used smart contracts for consensus computing that enforces the correct execution of the EHR systems	-	YES	-

8.	To propose a protocol which offers secure sharing of medical records	Provides a design which ensures secure data sharing with the help of a blockchain based architecture	Blockchain Technology with Smart Contracts	Smart Contracts helps in secure sharing medical data and solves data access issue by making information available to authorized user	-	NO	-
9.	Evaluate the potential of distributed ledger technologies through an EHR system which is blockchain based		Consensus algorithm named Proof of authority is used	Cost effective No high traffic as like public blockchains hence this is fast. No threat from 51% attack on blockchain network.	-	NO	-
10.	To propose a method for storage and retrieval of medical information	EHRs using BlockChain Technology	Permissioned BlockChain, Hyperledger Fabric Symmetric Key Cryptography and Public Key Cryptography SHA-256 algorithm for hashing	* Verify user identities on the basis of their previously registered details. * Guarantees the integrity of each block as it is easy to detect any tampering of records * Only the patient can provide access to his medical records * Access to medical records will be provided to the right person	-	NO	-

11.	To propose a design of blockchain based system which facilitates sharing of healthcare data	Blockchain centric system architecture model to provide: i)Integrity of shared data ii)Pseudonymity of user iii)Invoking properties of blockchain like immutability,accountability using smart contracts that are present	Layer 1 as web/cloud platforms Layer 2 as cloud middleware Layer 3 as permissioned Blockchain network Hyperledger fabric	Security of healthcare data is ensured during sharing Sharing will be done based on approval of required participants in network	Cloud might not be able to store large data Cloud also becomes centralized as all the data will be stored in it giving hackers a chance to get healthcared ata	NO	Secure sharing of data is very important as EHR systems deals with most sensitive data Secure sharing can be ensured using smart contracts
12.	Applications of Blockchain Technology in healthcare industry : Current landscape and challenges	Reviewing use cases of healthcare industry like management of health information,research,billing management etc..using blockchain technology	Blockchain network	Ensuring privacy ,traceability , integrity and accountability of data	If every node processes transaction, overall productivity of blockchain becomes slow which makes it unfit for handling numerous transactions present in real world	NO	Blockchain platforms need to maintain three desirable properties like Decentralization , consistency and scalability

13.	Blockchain based EHRs for Regulated Healthcare Jurisdiction	Smart Contracts present in blockchain helps in having secure sharing of data and provides data integrity as well	Public Blockchain network and Smart Contracts	Smart Contracts will be of prime importance in blockchain , which provides new measures of transparency and data sharing	Scaling issue needs to be resolved in order to handle real world transactions	NO	Privacy as well as security issues can be resolved using smart contracts of blockchain technology

ARCHITECTURE

The proposed solution of the above mentioned includes three parts: a permissioned fabric blockchain network, an offline storage for storing the actual data and an interactive platform for the users to access the network.

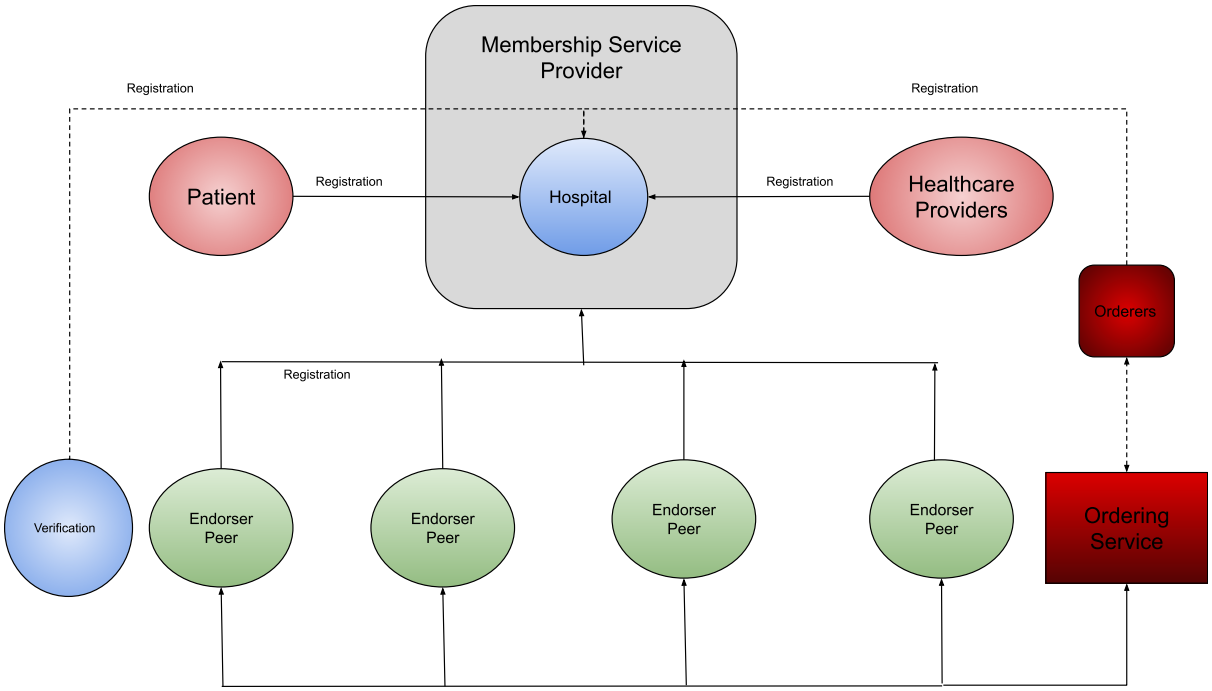


Fig.1. Registration of the users and other peers in the blockchain network

1. **Permissioned Blockchain Network**

   Hyperledger fabric provides a distributed and permissioned blockchain which ensures that all the participants are trusted and known to the user. The chaincode lays down the rules and regulations and the user can modify who all such as doctors, pharmaceuticals or researchers can access the data and who all can modify the data present.Once the participant is registered on the blockchain network it is provided with a unique id which is generated by the membership service provider (MSP).

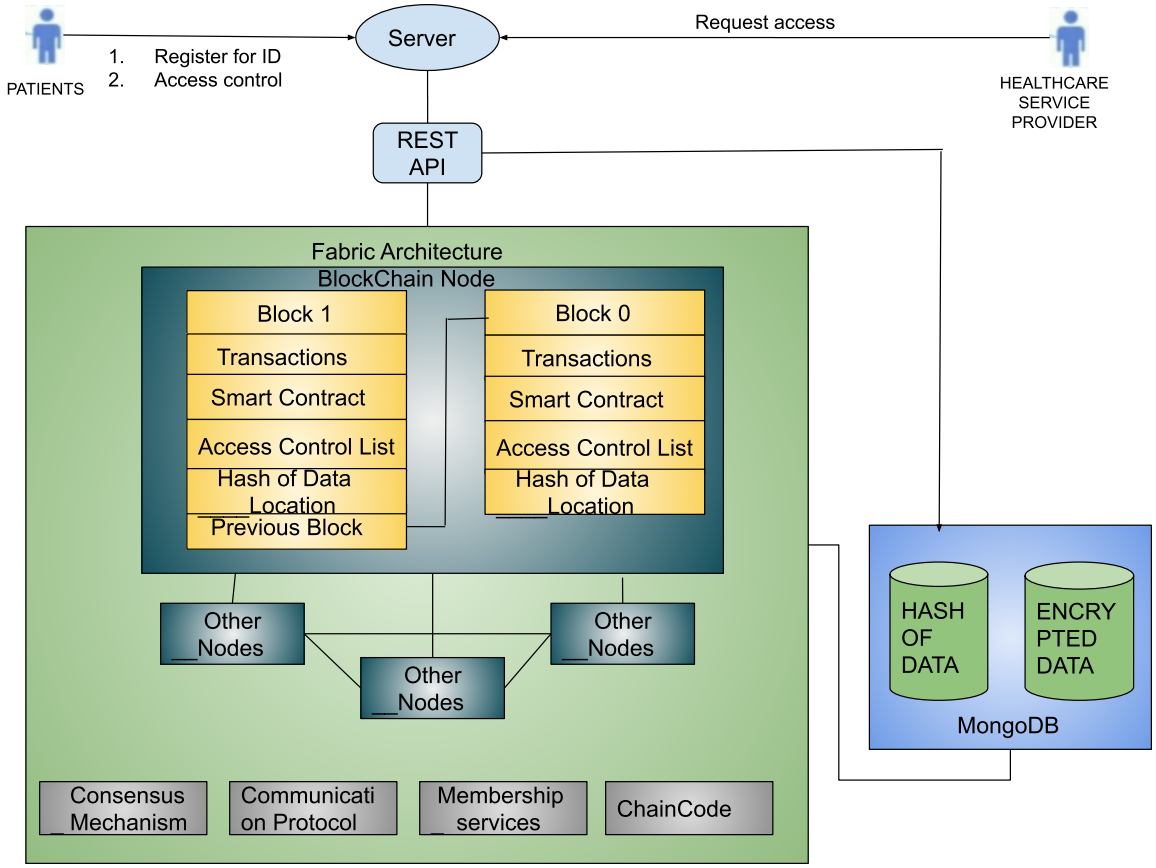
   The transactions performed are stored on the blockchain network in an immutable manner and can't be modified once stored. This ensures that the EHR's are secure and tamper proof.As all the participants are trusted and the chaincode terms allows the patient to make sure that his data is not misused.

2. **Mongodb**

Mongodb is a document oriented database program,which is a classification of NOSQL databases.The records which need to be uploaded will be stored in 64-bit format in mongodb. Hash value of records will be generated using MD5 hash algorithm and hash value along with required id will be stored in the world state database.As world state database keeps track of updated data and transactions,it's hash value will be stored in blockchain.This scheme of storage improves speed and performance.

3. **UserInterface**

A web based application,which helps users to communicate with the blockchain network is built,which allows all the participants to exploit the properties of the blockchain. The patients can register and then ask the doctors from whom got medical treatment to upload their EHR.The patient can also provide access to various other parties such as researchers, companies and pharmacies. Doctors can use that to look into the records as well as upload the record.


PATIENTS
1. Register for ID
2. Access control
Server
Request access
HEALTHCARE SERVICE PROVIDER
REST API
Fabric Architecture
BlockChain Node
Block 1
Transactions
Smart Contract
Access Control List
Hash of Data Location
Previous Block
Block 0
Transactions
Smart Contract
Access Control List
Hash of Data Location
Other Nodes
Other Nodes
Other Nodes
HASH OF DATA
ENCRYPTED DATA
MongoDB
Consensus Mechanism
Communication Protocol
Membership services
ChainCode

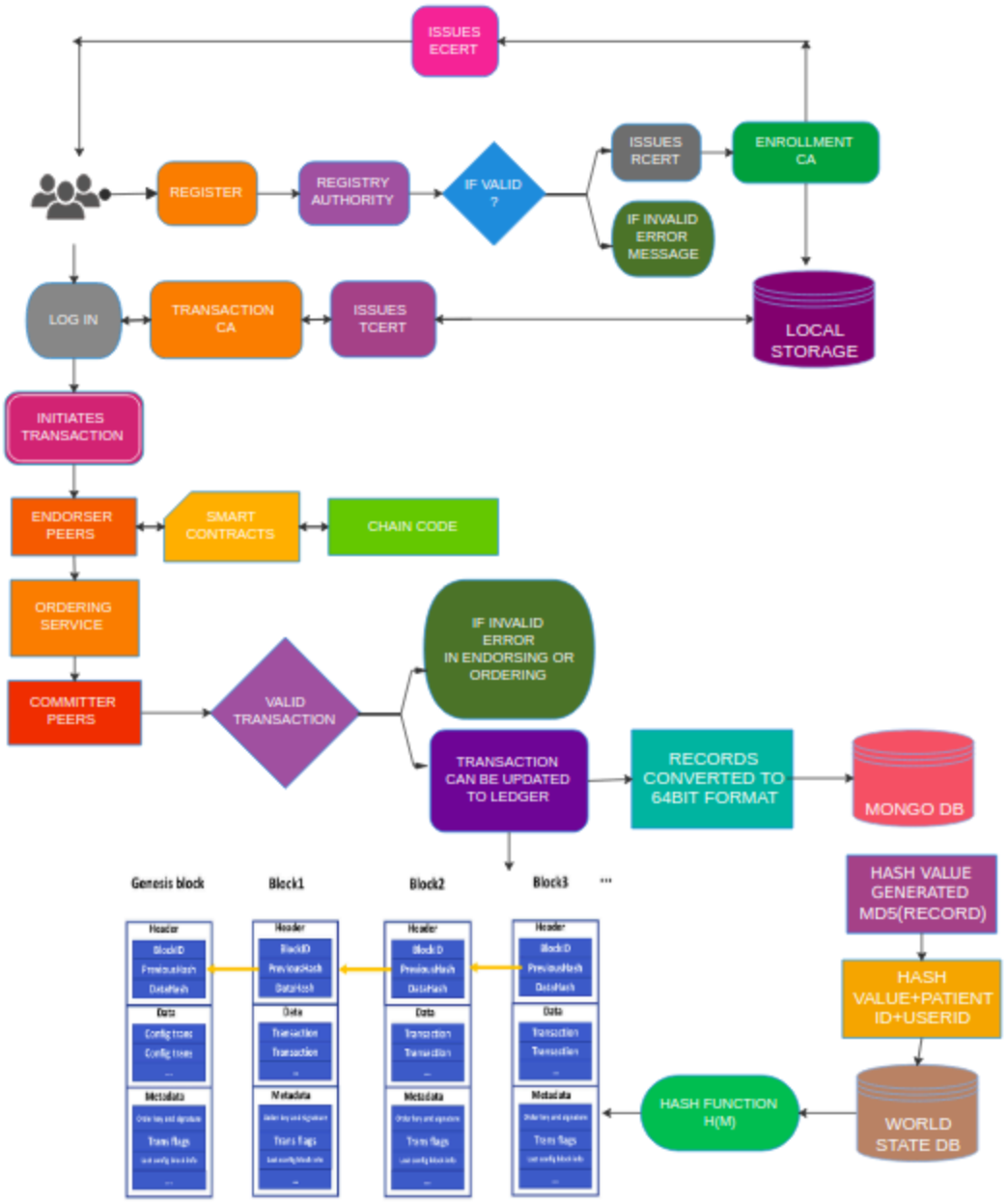


Fig 1: Flow Diagram for complete process

DESCRIPTION OF ARCHITECTURE :

Above diagram represents the high level architecture for secure exchanging of Electronic Health Records using a blockchain network.

Identity & Access Management :

Enrollment of users in hyperledger fabric network will be done through **MembershipServiceProvider(MSP)**.It consists of four entities namely Registration Authority(RA) ,Enrollment Certificate Authority(ECA) ,Transaction Certificate Authority(TCA) and TLS Certificate Authority(TLS-CA).These services are responsible for validating identity of user,issuing short term and long term certificates which can be used by participants to perform transactions.

Fabric CA is a Public Key Infrastructure(PKI) which makes use of **Elliptic Curve Digital Signature Algorithm**(ECDSA) mechanism which is secure for data exchange over public networks.

Identity & Access Control can be ensured by blockchain network as follows :

1) For the registration process , a user provides his Details to the Client Application
2) Fabric SDK in Client Application will be sending the Client-Register-Request to Enrollment CA which is a part of Fabric CA PKI . ECA then verifies the Enroll-Register Request of the client and validates the credentials.
3) ECA generates a ECert which contains the Public key of the User(K_{Pu}^{User}).These Ecerts are used for validating the User's registration Credentials.Along with ECert ECA generates self signed ECA-CertECA which main use is to prove user enrollment . ECA then responds back to the Client Application by sending ECert(K_{Pu}^{User}) and ECA-CertECA
4) At the Client side , ECert is decrypted using the ECA public key to retrieve the Client Public Key(K_{Pu}^{User}).

5) The Client User then sends a request for registration to TLS-CA for TLS-Cert .
6) After receiving the request TLS-CA creates the TLS-Cert containing TLS public key. Along with TLS-Cert it also generates self signed TLS-CertTLSCA and sends them to Client .
7) At Client side , TLS-Cert is decrypted using TLS-CA public key and its TLS public key is extracted.
8) Client Application then requests a TCA for Transaction Certificate containing key to the Key Derivation Function which is used to generate the Private key of Client.
9) TCA upon receiving the requests generates TCert and self signed TCA-CertTCA using the private key of TCA and responses back to Client.
10) Client decrypts TCert and generates Private Key by invoking DF-Key method.
11) Overall the Client stores Client Public Key from ECert, TLS public key from TLSCert, Generated Private Key(K_{Pu}^{User}) from TCert in its Local Database .

Initiating , Validating & Adding a Transaction to ledger :

1) After logging in, user can initiate or invoke a transaction like adding new records ,providing access or revoking access for updating health records.
2) For initiating a transaction, user requests the Transaction Certificate Authority (Transaction CA) for Tcert.
3) Tcert will be issued to the user by Transaction CA after verifying his Ecert.
4) When a transaction is initiated, it needs to be verified by peers in the network so that the transaction can be done and added to the ledger.
5) Verification of transaction follows execute-order-validate architecture which has the following three steps,

i) Endorser Peers are chosen by the endorsement policy.Once the user proposes a transaction,they are endorsed and executed by the endorsing peers by

performing the smart contracts using chaincode and then endorsed transactions are passed to the user.

ii) User then sends the endorsed transactions to the ordering peer who is responsible for ordering the transactions in some order.

iii) Committing or Validating Peers then validate whether the transactions have been endorsed and ordered correctly.

After the above steps , a transaction can be termed as a valid one and can be included in the ledger as well as the world state database.

Health records will be stored in Mongodb in 64-bit format.Hash value of record will be generated using MD5 hash algorithm.Hash value along with patient id and user id will be stored in world state database as a key value pair but not in blockchain since the amount of data that can be stored in blockchain is limited. As the world state keeps track of current state like data and transactions,it's hash value will be stored in the blockchain.

There will be three different records namely Electronic Health Records(EHR),Lab Records and Bills.Structure of records is mentioned below:

Structure of EHR :

Field	***Data type***
Patient_Id	*String*
Doctor_Id	*String*
Hospital_Id	*String*
EHR_Id	*String*
Time	*Timestamp*
Record	*StringArray*
Type	*String*

Structure of Lab Record :

Field	*Data type*
Hospital_Id	*String*
Doctor_Id	*String*
Laboratory_Id	*String*
Patient_Id	*String*
LabRecord_Id	*String*
Time	*Timestamp*
Record	*StringArray*
Type	*String*

Structure of Bill :

Field	*Data type*
Hospital_Id	*String*
Doctor_Id	*String*
Laboratory_Id	*String*
Patient_Id	*String*
Pharmacy_Id	*String*
Bill_Id	*String*
Amount	*String*

Time	*Timestamp*
Record	*StringArray*
Type	*String*

Cryptographic Preliminaries of the proposed system

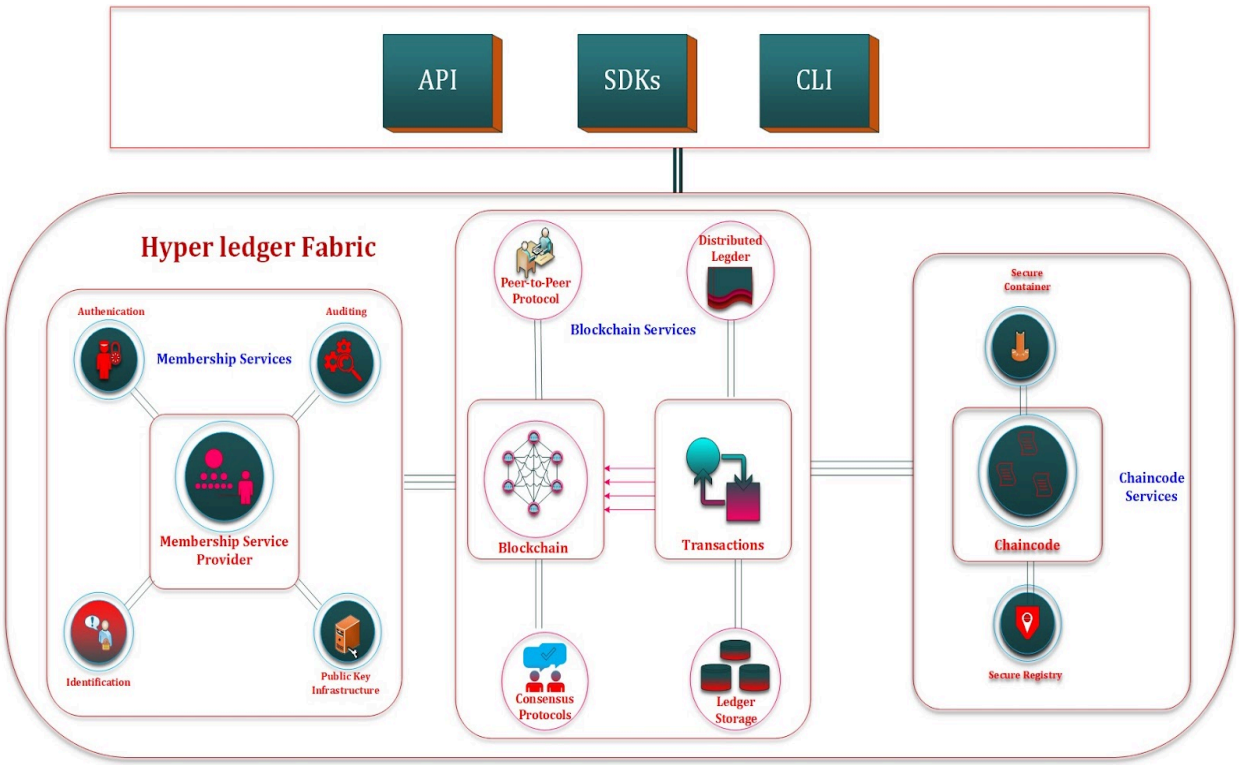
BID_{Issuer}	Blockchain Identifier of Issuer
K_{Pu}^{Issuer}	Public key of the Issuer
K_{Pr}^{Issuer}	Private key of the Issuer
$Sign_{Pr}^{Issuer}$	Signature of the Issuer
$BID_{Receiver}$	Blockchain Identifier of Receiver
$K_{Pu}^{Receiver}$	Public key of the Receiver
$K_{Pr}^{Receiver}$	Private key of the Receiver
EHRID	Identifier of requested Electronic Health Record
S_{Key}	Symmetric key of patient
MD5	Message Digest hash algorithm
UID	Unique Identification Number (Aadhaar , SSN etc..)
Rcert	Certificate issued by Registration Certificate Authority
Enrollment CA	Enrollment Certificate Authority
Ecert	Certificate issued by Enrollment Certificate Authority
ECAcert	Certificate of Enrollment CA used for verification
TLS-CA	Transport Layer Security Certificate Authority
TLScert	Certificate issued by TLS-CA
TLSCAcert	Certificate of TLS-CA used for verification

Transaction CA	Transaction Certificate Authority
Tcert	Certificate issued by Transaction Certificate Authority
TCAcert	Certificate of Transaction CA used for verification

ARCHITECTURE & OBJECTIVES :

Objectives can be achieved by above mentioned architecture as follows :

1) **To setup a Hyperledger Fabric Blockchain Network with clients and Healthcare providers :**



SYSTEM SETUP

Operating System : Ubuntu 18.04

Golang Installation

- sudo apt-get install golang-go
- export GOPATH=$HOME/go
- export PATH=$PATH:$GOPATH/bin

NPM Installation

- sudo apt-get install npm

Docker Installation

- curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
- sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
- sudo apt-get update
- apt-cache policy docker-ce
- sudo apt-get install -y docker-ce
- sudo apt-get install docker-compose

Hyperledger Fabric Installation

- sudo curl -sSL https://goo.gl/6wtTN5 | sudo bash -s 1.1.0
- sudo chmod 777 -R fabric-samples

Testing System Setup

- cd fabric-samples/first-network
- sudo ./byfn.sh generate
- sudo ./byfn.sh up
- sudo ./byfn.sh down

2) **To identify and authenticate the clients and Healthcare providers :**

i) For the Enrollment process , the user provides his details to the Client Application .

Enroll(User_Details)

ii) Fabric SDK in Client Application will be sending the Client-Register-Request to Enrollment CA

Client-Register-Request(Enroll_Request)

iii) Enrollment CA after receiving the request from Fabric SDK will then create the signature with an enrollment certificate termed shortly as ECert that contains the public key of client and generates self-signed ECA-CertECA.

Generate(Ecert(User_Details))
Sign ECert using ECA

iv) Enrollment CA then sends the signed response containing ECert,ECert(Public Key) , ECA-Cert to the Fabric SDK of the client user.

Client-Register-Response(Ecert,Ecert(Public-Key),ECA-Cert)

v) After Receiving the response from ECA , Fabric SDK of Client User verifies submitted public key and ECert public key are the same.

Verify_ECert(ECert,ECA-Cert)

vi) Client User sends a registration request for TLS certificate to TLS-CA . The TLS-Cert mainly carries the TLS public key of the client .

Client-TLS-Cert-Request(ECert)

vii) After receiving the request , TLS-CA verifies the client user credentials and creates the signed TLS-Cert containing client user TLS public key and also generates self signed TLS-CertTLSCA.

Generate (TLSCert(Ecert))

Sign TLS-Cert using TLSCA

viii) After generating TLSCert TLSCA responds back to Client User

Client-TLS-Response(TLS-Cert,TLS-Cert(Public key),TLS-Cert$_{TLS-CA}$)

ix) After receiving response ,the client user verifies the submitted public key and the TLS-Cert public key is the same . If the same TLS-Cert is decrypted using TLSCA public key in order to retrieve client TLS public key and store it in its local database.

Verify_TLS-Cert(TLS-Cert,TLS CA-Cert)

x) Next, the Client Application requests for Transaction Certificate to Transaction-CA(TCA).

Client-Transaction-TCert-Request(User Details)

xi) After receiving the request TCA returns a TCert containing Key-DF key which will be used to obtain the private key of the client.

Generate(TCert)

Sign TCert using TCA

Client-Transaction-TCert-Response(TCert(Private-key),TCA-Cert)

xii) Next , the client user verifies TCert by decrypting using TCA public key and invokes DF-key method using Key-DF key given by TCA .

Verify_TCert(Private key)

xiii) Now the Client User stores Ecert(Client Public Key),TLSCert(TLS Public key) , TCert(Private Key) in its local database.

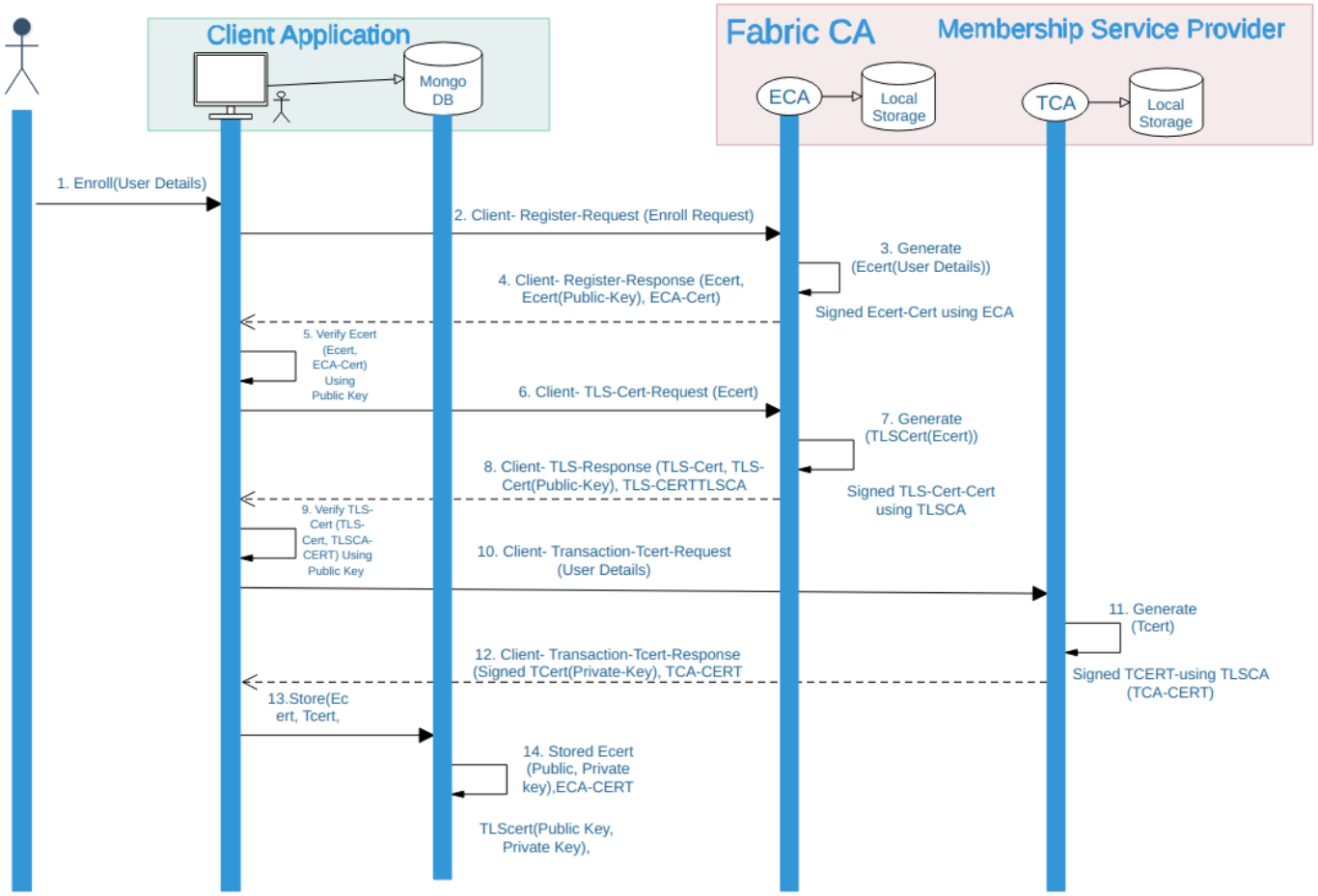


Fig 1:Registration & Enrollment of users on to the Blockchain Network

3) **To store the Health records in the Blockchain in distributed manner :**

Blockchain is a decentralized system in which each participant in the network contains a copy of ledger and contributes to the procedure of validation and certification of transactions.

i) Here data will be fragmented and distributed across the network rather than having complete data at a single place.
ii) These chunks are encrypted and uploaded onto the blockchain.
iii) They are distributed in a manner that all chunks are available even if part of the network is down. This is also known as redundancy.
iv) The transactions will be added to a shared ledger by participants of the network.
v) Algorithms will be made to run by users to validate the proposed transaction before adding it to the ledger.
vi) For a transaction to be added to ledger,majority of users need to approve that as a valid one.
vii)Whatever changes that are performed to the ledger,they get replicated in all copies in the network.
viii)If a transaction gets added to ledger,then it cannot be removed or changed.
ix) As all the participants will be having a copy of the ledger and changes can be made only when the majority approves that, it is impossible for a single user to change data.
x) The system can handle situations in which one or two nodes fails since every user will be having a copy of ledger.

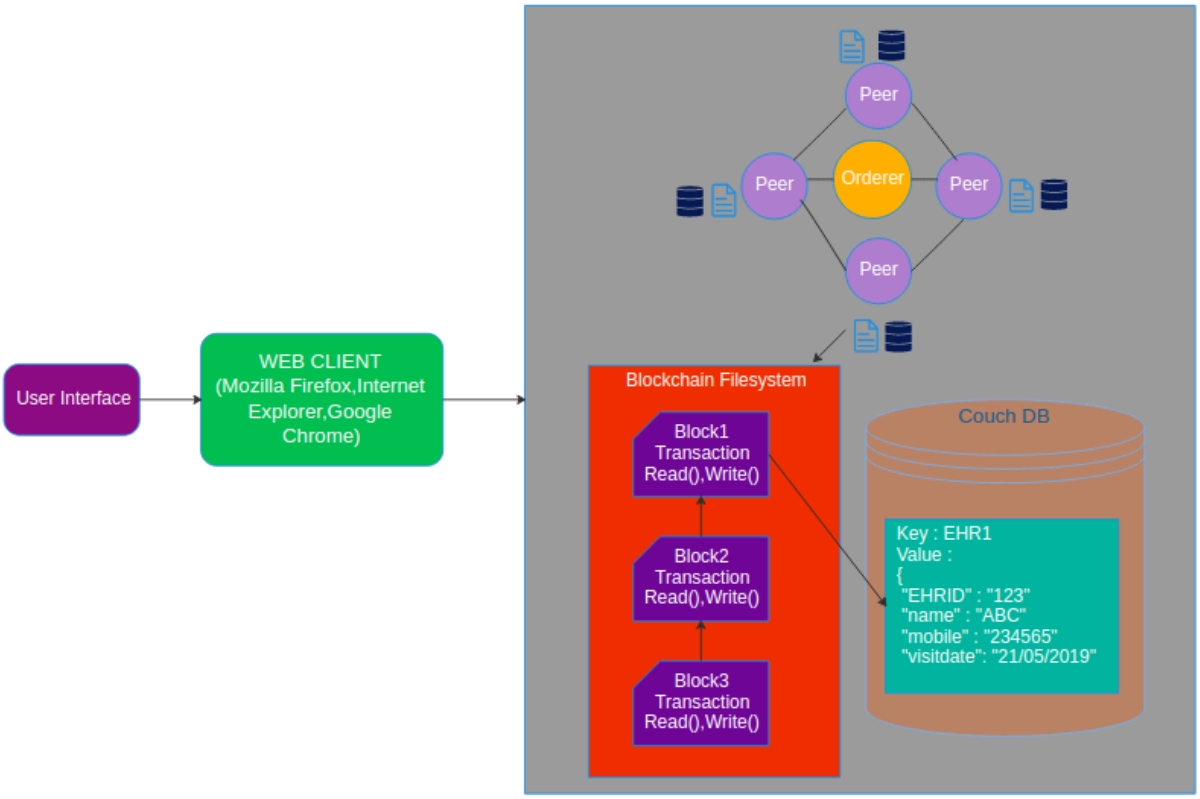


Fig 2:Storing Electronic Health Records in distributed manner

4) **To securely share health records of clients with the Healthcare providers :**

Sharing health records to healthcare providers is one of the transactions which needs to be endorsed, ordered and committed to the ledger after validating it with endorsement policy.

i) To initiate or invoke a transaction ,user requests Transaction CA to issue Tcert using Ecert and ECAcert, the certificates provided by Enrollment CA.

Transaction_Request(Ecert,ECAcert)

ii) Transaction CA provides Tcert to the user ,after verifying ECAcert whether the user is enrolled by validated Enrollment CA and Ecert stored in the database.

Verify(Ecert,ECAcert)
Tcert = Generates_Cert(Ecert)

iii) User makes a transaction using Tcert and sends the transaction proposal to endorser peers.

iv) Based on a specific endorsement policy, endorser peers verify the transaction and send it to orderer peers.

v) Orderer peers order various transactions in a particular order so that the system remains consistent.

vi) Orderer peers send the ordered transaction to committer peers , which validates the transaction and commits it to the distributed ledger.

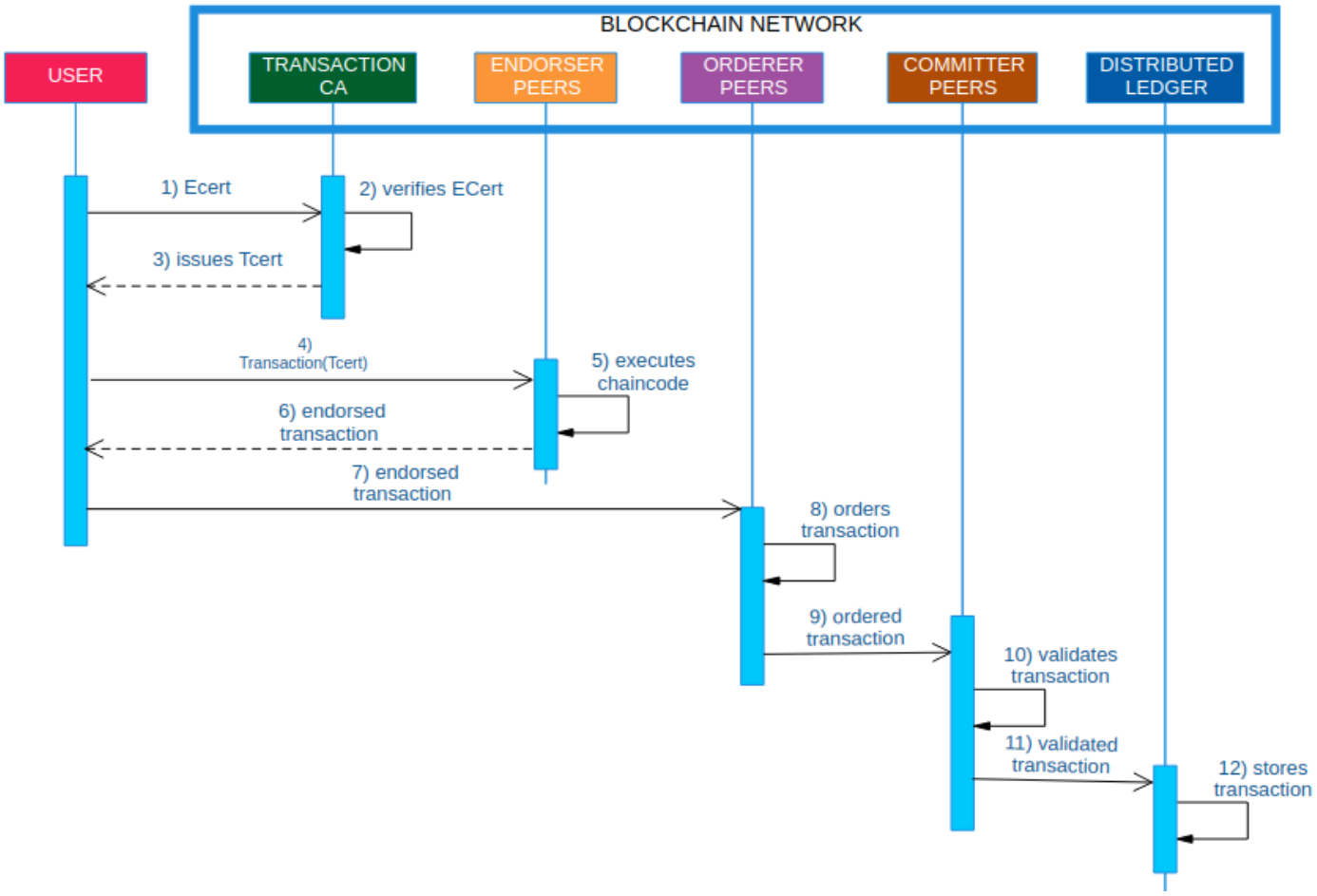


Fig 3:Transactional operational process in Secure Exchanging of Electronic Health Records

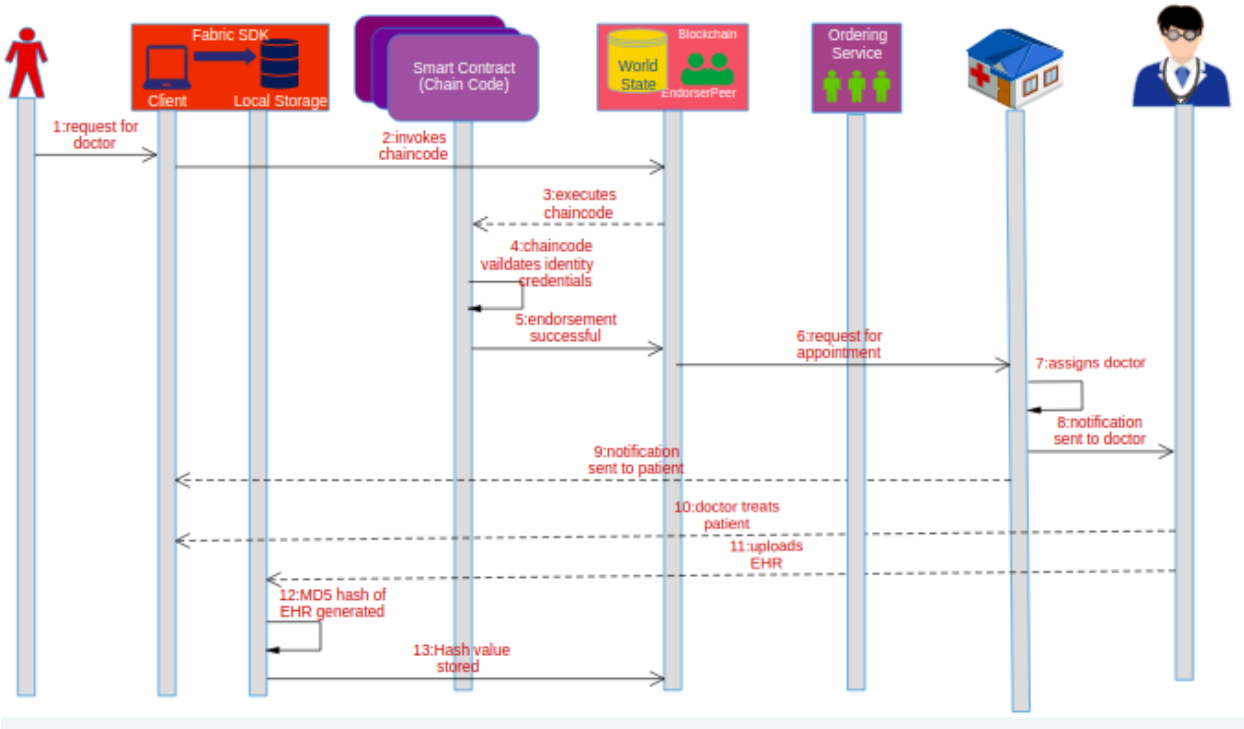


Fig 4: Doctor uploading records owned by patient

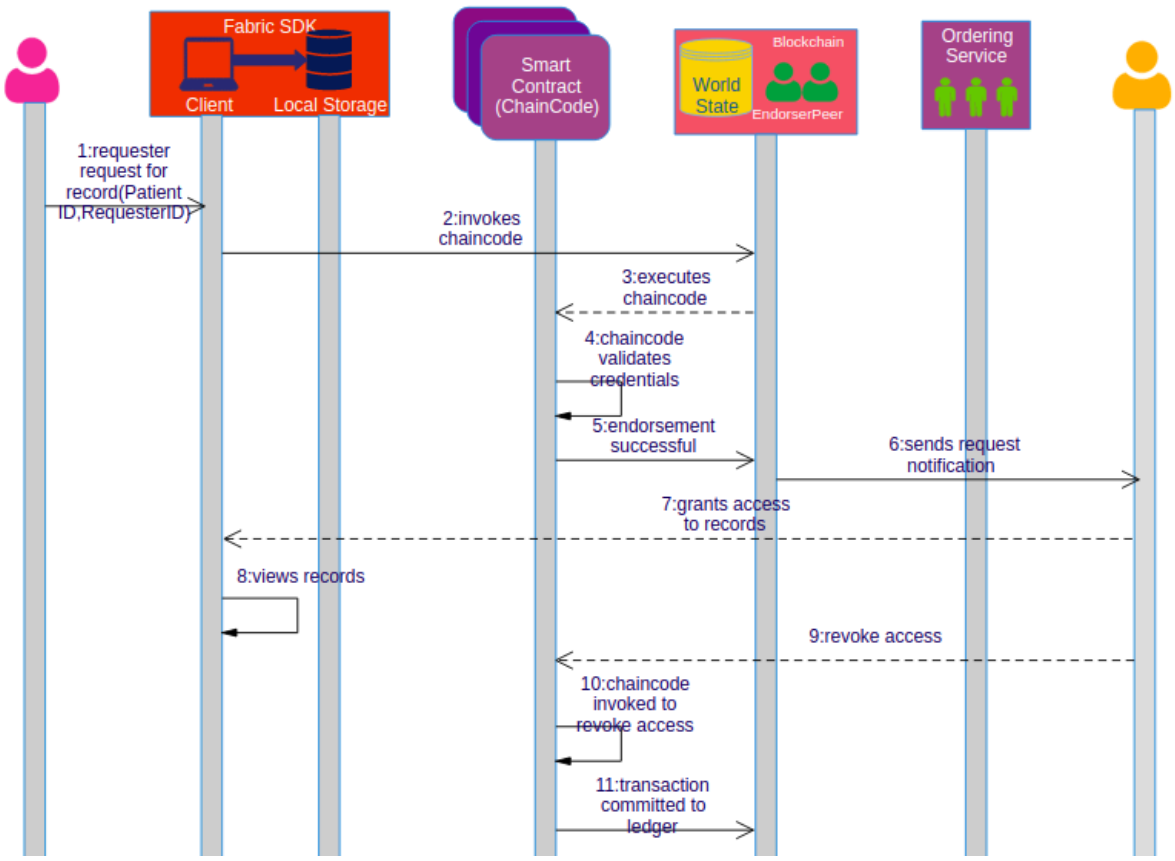


Fig 5: Process of sharing records owned by patient to requesters

Secure sharing of health records can be ensured by using encryption keys as follows :

i) Along with public and private keys generated for users at the time of registration, patients will have a symmetric key (S_{Key}) with which their health records get encrypted and stored in a database.

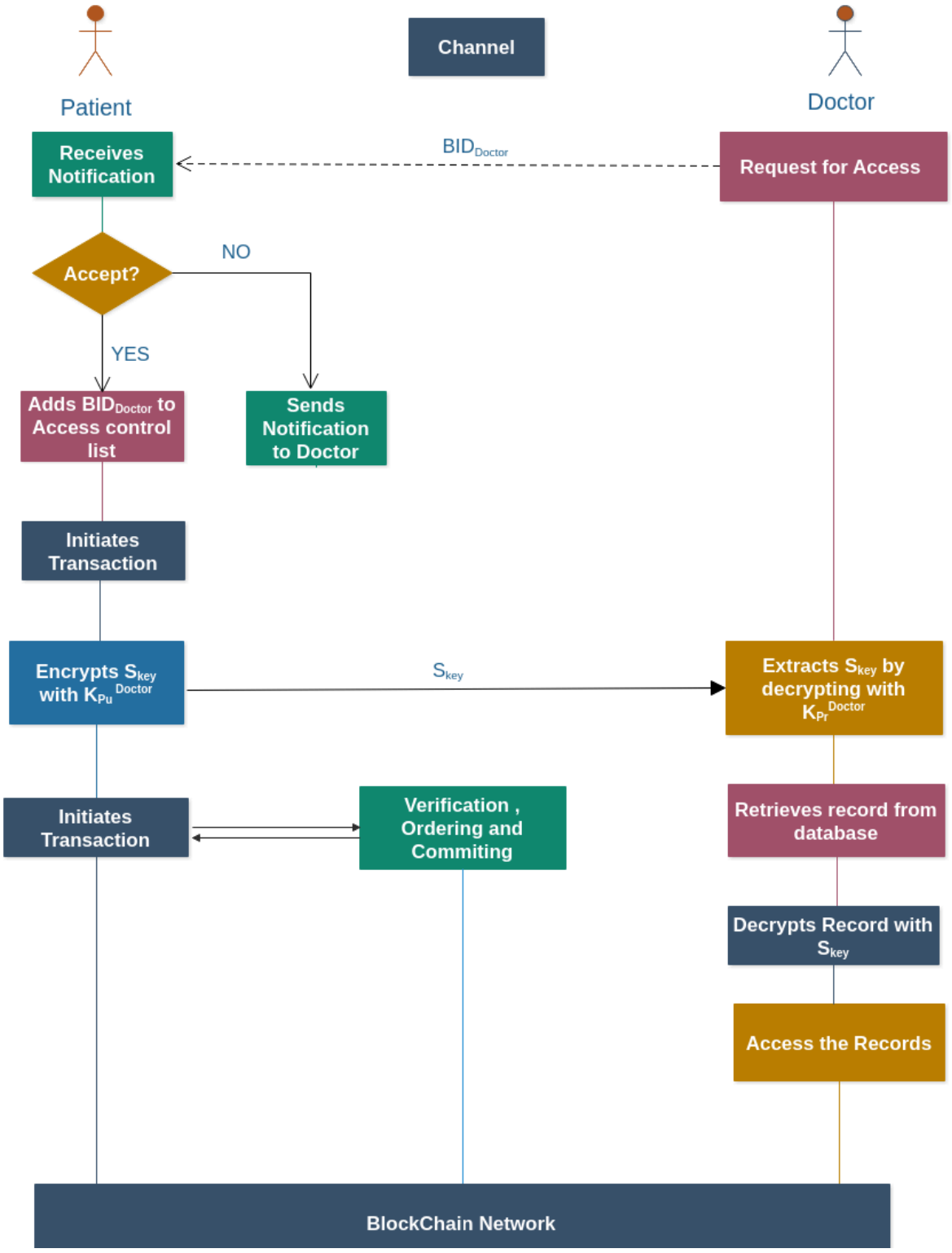
ii) When a patient's health records are requested to be provided by any healthcare provider using his public key ,it's completely the patient's choice to provide access or revoke access to his health records.

iii) If a patient wants to provide access to healthcare provider, he can share S_{Key} with healthcare provider by encrypting S_{Key} with receiver's (healthcare provider's) public key ($K_{Pu}^{Receiver}$)

iv) If the symmetric key (S_{Key}) is compromised, patient can generate a new one by running a pseudo random generator algorithm.

v) After getting key details, the healthcare provider decrypts S_{Key} using his private key ($K_{Pr}^{Receiver}$) and retrieves records from the database.

vi) He decrypts the record using S_{Key} and performs respective operations.



Steps for Secure Sharing of Health Records between Doctor and Patient :

Function *: Share_Record()*

1. Doctor requests the patient for access to his health record with an identifier EHRID and sends Doctor Id along with the request.Here Patient will be Issuer and Doctor will be Receiver.

   $Request\_Patient(EHRID, BID_{Issuer}, BID_{Receiver})$

2. Patient will now receive a notification and can either accept or reject the request sent by the doctor . If the patient clicks ACCEPT , $BID_{Receiver}$ (doctor) will be added to the Access Control list (ACL).

   $Add\_to\_ACL_{Issuer}(BID_{Receiver})$

3. Initiates transaction of Key Sharing :
   a. Patient will encrypt the symmetric key(S_{Key}) and EHRID with public key of the receiver (doctor) ,where EHRID is already encrypted with the symmetric key(S_{Key}) and sends it to the doctor .

      $Send\_to\_Doctor(Encrypt(S_{Key}||EHRID, K_{Pu}^{Receiver}))$

   b. Doctor will now receive the encrypted S_{Key}||EHRID as M and Decrypts it using Private key (private key of receiver)

      $Decrypt(M, K_{Pr}^{Receiver})$

4. The transaction is added to the blockchain.
5. Doctor can now retrieve the encrypted health record from the database using EHRID and decrypt using the S_{Key} and access the records.

   $EHRID = Retrieve\_Record(M, S_{Key})$

5) **To ensure the privacy of stored EHRs :**

Privacy in blockchains can be ensured using cryptographic keys named public and private keys.These keys are strings of numbers and alphabet generated randomly and are cryptographically related.One can share his public key with others because it doesn't reveal any kind of information about the user and it is difficult to derive private key just by knowing public key.

In our proposed architecture , we have two types of storages. They are :

i) **Online storage** :

In online storage,personal data about the clients,hash values of encrypted records and transactions will be stored.Ledger in the blockchain network is referred to as online storage.The link to corresponding data in offline storage is also available.

ii) **Offline storage** :

In offline storage, actual records of patients are stored in encrypted forms as key value pairs and the link to corresponding data in online storage is also available. CouchDB can be used as offline storage.Clients cannot access offline storage directly without permissions.Records can be viewed only when access is granted by a patient based on the access policy of the blockchain network.

i) When a third party requests access request handler for providing access to EHR data using his BID & EHRID , access request handler sends BID to access policy repository which is in blockchain to verify whether the third party can be granted access.

ii) Access policy repository verifies BID and sends Access Verification Response to Access Request Handler.

iii) If Access Verification is valid, then Access Request Handler provides Access Response using which third party can access EHR data.

iv) If Access Verification is not valid, then Access Request Handler queries patients whether or not a third party can be provided with access.

v) If the patient agrees to provide access , then Access Request Handler sends new access policy to Access Policy Repository and requests to verify again.

vi) After verification, it sends the result to Access Request Handler which then sends response to the third party to access resource.
vii) If the patient denies granting access to the third party, then a rejected request notification will be sent to the third party by Access Request Handler.

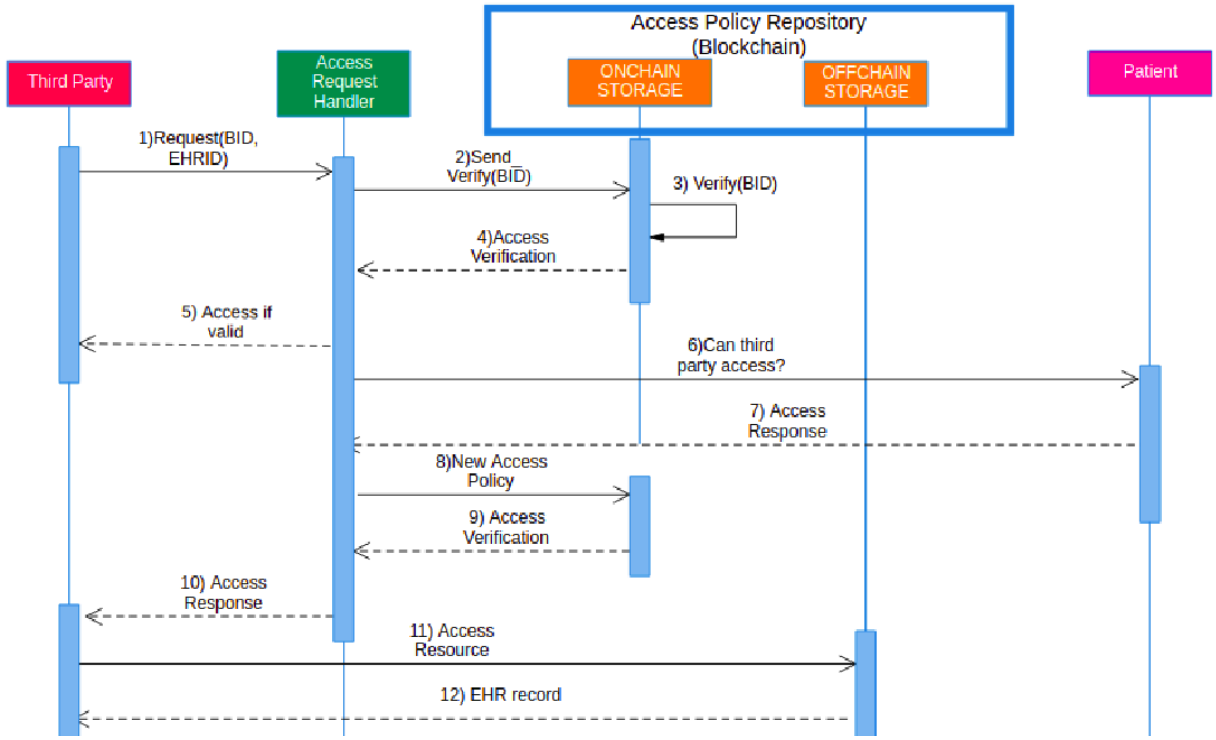


Fig 6: Ensuring privacy of Stored Electronic Health Records

6) **Securing client data :**

Every member of the blockchain network must follow certain guidelines related to privacy to protect their data.Any unauthorized way of accessing information can be restricted using some security measures.Blockchain servers use TLS (Transport Layer Security) protection to prevent individuals from accessing any kind of information they are not authorized for.

i) Client data will be encrypted using his own private key , which is generated at the time of registration using Elliptic Curve Digital Signature Algorithm (ECDSA)

$$\{K_{Pu}^{Issuer}, K_{Pr}^{Issuer}\} = Generate\_Keys\_ECDSA(UID_{user})$$

$$Encrypt(EHRID, K_{Pr}^{Issuer})$$

ii) Data will be stored in offline storage in encrypted form , which can be decrypted only when attackers get required keys .
iii) Hackers obtaining encrypted health data would need to also steal the keys to make use of the information they obtain.
iv) As sharing of keys depends completely on the patient,keys will not be compromised easily.
v) Combining keys with smart contracts prevents unauthorized parties from adding information to a patients' records, including outsiders trying to tamper with data for malicious or self-serving purposes.
vi) Moreover , encrypted data will be hashed using hash generating algorithm H_{256} and hash value will be stored in blockchain.

$$Stored\_in\_Blockchain = H_{256}(Encrypt(EHRID, K_{Pr}^{Issuer}))$$

vii) As mentioned above, by storing data only after encryption , security of clients' data can be ensured.

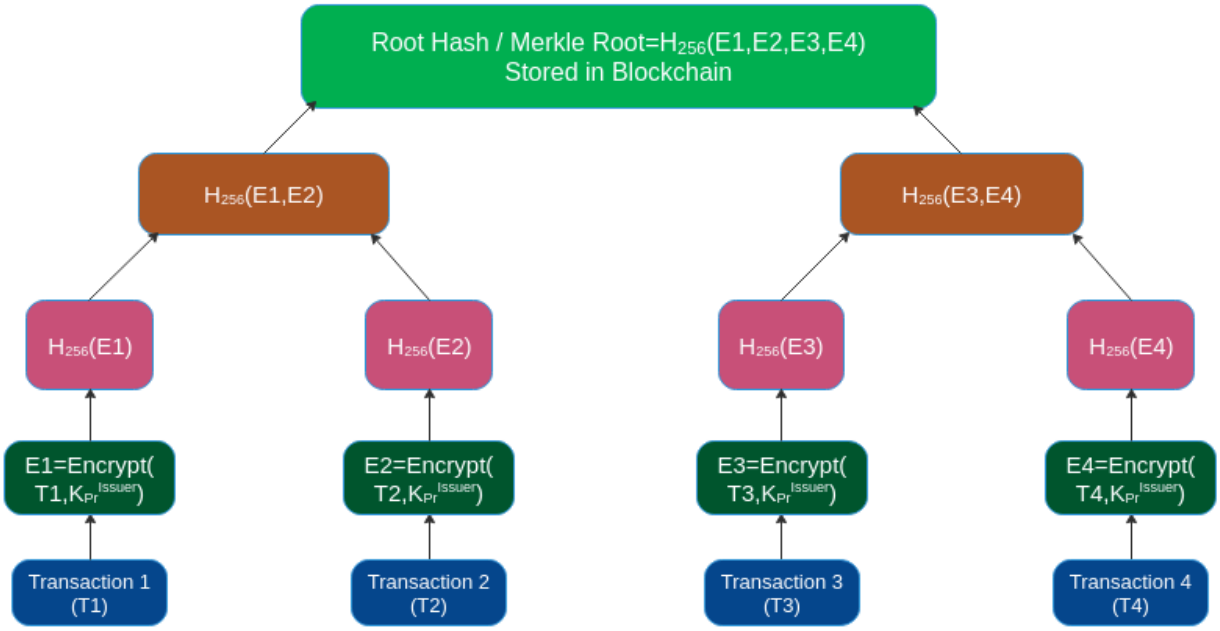


Fig 7:Securing clients' data

IMPLEMENTATION DETAILS AND RESULTS

The solution that we have proposed to the above mentioned problem using a permissioned blockchain network that is implemented in hyperledger fabric that provides our application services such as registration of various entities and as well verification of those entities. The blockchain network permits the various users of this application such as:the patients, doctors, hospitals and other hospital entities such as laboratories and pharmacies to enroll into the network through a trusted MSP which is responsible for creation of signing certificates as well as the public and private keys for authorisation and verification through the *Certificate Authority(CA).* The fabric network comprises of various services which are: *Membership service provider (MSP)* which is used for issuing the certificates through *Fabric Certificate Authority(CA)*, *Ordering service(s)* which orders the transactions, Endorser peer nodes used for the endorsement of the transactions, Distributed Ledgers, Channel(s) and Chaincode(s).

The ordering service is responsible for ordering the various transactions by using the consensus mechanisms.The configuration file for this service contains info about the channel(s) within the blockchain network. The file also contains the membership information (certificates) for all the members of the channel within the organization and policies of the channel within the network.

Compared to other blockchain solutions the fabric network provides multiple benefits first of which is its permissioned so any organisation looking to harvest the benefits can easily deploy this in their organisation, the other benefits that it give includes Efficient parallelism and concurrency, efficient commitments of the transaction into the ledger, multiple transaction executions, etc.

We have used *NodeSDK(software development kit)* for constructing the gateway through which we interact with the blockchain ledger and the endorser peers. The *API* enables us to invoke chaincode functions and at the same time submit and evaluate transactions also. This SDK is also responsible for enrolling the users and issuing the *TLS* certificates as well as storing the keys and the identity generated in a wallet.

The End user can interact with this gateway and the server through the React framework which is used to invoke the *REST APIs* and perform the *GET* and *POST* requests to the server.These requests are used to either send or retrieve information from the server which in turn sends the request details to the blockchain network which in turn either updates the world state or fetches data from the world state.

The EHRs and other similar documents generated are first stored in the distributed database in base64 format and the hash value is sent to the blockchain. While retrieving the files firstly the hash value is matched with the hash value of the document stored which ensures that the data is not tampered in any way.

Registration/Enrollment and Identification of Admin/Patients/Doctors /Hospitals and Other entities in Hyperledger Fabric}

A fabric blockchain network has multiple peers , while some of them are the default peers some of them are the client peers or the entities that will register on the network and invoke chaincode to do transactions. The network has peers such as endorser, orderer and anchor, in every organisation.Every participating peer will have an identity in the fabric blockchain network provided by the membership service provider (MSP) module of the fabric. This MSP module contains security infrastructure that issues and maintains the identities of the clients, verifiers endorsed peers, orderers in the blockchain system at the time of the registration process. These credentials are mainly used for identification, authentication, and authorization purposes.The registration of nodes and key management is the part of the MSP service, which uses standard PKI methods for performing the digital signatures on the transaction and authenticating the peers and can provide the certification authorities (CAs). The orderers and endorsed peers' credentials are also generated by a fabric-CA of the MSP module and distributed to all the peers in the network. The hyper ledger fabric provides a special peer called an admin, who has administrative control, registered with the Fabric Certificate Authority (CA) and is responsible for enrolling or removing the entities in the network. The Fabric-CA sends the enrollment call to the Fabric-CA root server to retrieve the enrollment certificate (ECert) for this admin user. The Fabric-CA root server invokes a certificate signing request (CSR) in the ECert of this Admin, which stores the ECert key material in the blockchain network and the identity and the keys are stored in the wallet meant for storing the keys and the identity info. The admin is mainly used to register and enroll a new Client, endorser Peers, and Orderers into the network. This admin peer is also used to install the chain code on every endorsing peer that has been enrolled.

Various clients use the NodeSDK, which accesses the API for performing the user identity registration and enrollment process using membership service process (MSP) and connects to the network to submit transactions for querying or updating the distributed ledger(updating details or storing records metadata or hash). When these clients register the identity as well as the public and private keys for these clients are generated by the above mentioned flows and the PKIs and identity details are stored in the wallet and will be used for the authentication as well as the digital signature for all the transactions they submit.

The UI/UX for the clients is developed using React-Redux and can be used by doctors, patients, hospitals, laboratories, pharmacies, researchers and insurance companies to register onto the network. The REST API calls are made by the clients to the server running using the Node SDK to connect a gateway to the network. All the details are sent with the POST API call which sends all the details and the username, registrationNo or the medicalRegistrationId is used to create the id for all the clients that register needs to register on the network.

Once registered the clients can logIn to their respective portals and start doing transactions which will be signed by the secret key that is stored in the wallet for every entity.

Authentication of the Registered Users

Admin is responsible for checking the authenticity of the users registered in the network. All the identities that are stored in the wallet are monitored by the admin and in case the admin believes after checking from the national database that the registered identities are fraud or are no longer licensed to do anything then the admin will remove those identities from the wallet and they no longer be able to perform any transaction.

Admin keeps monitoring all the registered entities and in case it finds that any of the users is not acting according to the norms set by the application then it may remove that user.

The national database present is not linked with this application and can be accessed via the Indian Medical Register.

Only the users that are verified will be allowed to keep submitting the transaction. The admin always monitora all the entities that are present via UI , it can remove those by making an API call which in turn removes the identity.

The authentication of the transactions are taken care of by the blockchain and the *endorser peers* and the *MSP*. The identity generated will be used to submit transactions and the private and the public keys along with the certificates are responsible for controlling the type of transaction the user can perform. The digitally signed transactions ensures that the users doing those transactions can be traced back to them. The world state *(couchDB)* stores all the data submitted to the blockchain network.

Privacy of the Users And UserDocuments

This is a permissioned blockchain network so only the users that are authenticated by the admin will be allowed to remain on the network and do the transactions. So the details that will be shared by the various client peers to others will be viewed only by them.

Admin after allowing the client nodes to register, ensures that the client identity will be used to do the transactions. The users through the UI makes requests for the documents, or the user information. So when the document is requested by the users who don't have access to those documents will only be accessed to those documents after given permission by the user possessing those documents.

The details are registered in the couchDB in the key value manner, so that the data stored in the blockchain database can only be accessed by the key holder through the gateway which is established using the NodeSDK.

The users whenever they make a transaction by entering the details through the React UI. The details are sent to the server through the REST API call the identity of the client is verified first. Only after verifying the identity they are allowed to submit the transaction to the network. The wallet stores the identity and makes sure that the authorised peers only submit the transaction.

The chaincode is invoked and the changes are made into the world state using the key of the client node. The transactions are digitally signed using the normal PKIs techniques.

Hospital Appointment and Doctor Assignment

The patients can invoke the chaincode exposed through the NodeSDk by using the React framework as the interactive UI. Whenever a user wants to book an appointment the patient searches for the nearby hospitals and inputs the time and the ailment the patient is feeling after selecting the hospital. After that on requesting for appointment the React framework does a POST call to the server.

As soon as the server receives a book appointment call , the server checks whether the identity of the patient exists in the server or not if not it suggests the user to register first. After verifying the existence of the identity of the patient a gateway is created which connects to the network and the chaincode is fetched. After that the chaincode is invoked and the parameters are passed to book an appointment in the respective hospital chosen by the patient.

When the chaincode is invoked by the gateway the endorser peer executes the chaincode and the credentials of the patient is verified. Once the endorsement is successful the ledger for that peer is updated and the endorsing peers return a proposal to the client .The endorsed transaction proposals are then ordered into blocks and then distributed to all peers for final validation and commit.

Once the transaction is committed the hospital gets a new notification about the new appointment request. The hospital based on the ailment mentioned selects the appropriate doctor from the doctors working in the hospital. After selecting the doctor the hospital makes a POST call to assign the doctor for that particular appointment.

On receiving the post call to assign a doctor, the server first checks for the existence of the identity of the hospital through the registration number of the hospital. Once the identity is verified the gateway created by the hospital identity checks in the connection profile for the peers and the certificates of the hospital. After that the contract is invoked which is already deployed on the various peers in the network. Once the chaincode is invoked the endorsing peers execute the chaincode with the parameters passed to the chaincode. The ledger for the user is updated and the endorsement is sent back as a proposal to the gateway which is ordered into blocks by the orderers in the gateway. Once the world state is updated the doctors and other peers ledgers are also updated. Once the doctor which was assigned the appointment is updated a notification is sent to that doctor informing about the coming patient.

Generation of EHR By the Doctor

On completion of the treatment of the patient by the doctor, the doctor generates the EHR for the patient. After that the doctor selects the appointment booked by the patient for which the patient was treated by the doctor. The doctor selects the EHR of the patient and uploads on the web-app. Once the document is uploaded, based on the requirement of the laboratory or pharmacy the doctor chooses the laboratory or pharmacy present in the hospital. The doctor makes a POST REST API call in order to upload the document.

The document is sent to the server. As soon as the request is received the multer servicer handles the file and makes it ready for upload in the database. The online database of this application is mongo which is used for storing various non essential details and also the files uploaded by the doctors, hospitals, laboratories and pharmacies. The grid-fs-storage package by mongo is capable of storing the files in chunks and one separate collection which stores the metadata as well as the hash (md5) of the file. The name of the file is then encrypted using a random function and stored in a different collection by the name of the patient username. The hash value will be used later to ensure that the record is not tampered with and to verify the hash value to gain access to the data. The file is stored in Base64 format and divided into chunks in order to facilitate the distributed nature of the database.

The new random encrypted name of the file and the hash value of the document uploaded is then sent back along with the patient and the doctor id and the doctor signs the transaction. The hash value and the file name is used to create a document in the world state and once the endorsement is complete the ordering services broadcasts the block to all the participating nodes in the network.

The file is updated in the world state of the patient and the access is completely transferred to the patient alone. Based upon the request and the need the access is given to various other clients.

Request For Document Access and Grant/Revoke Access

Various types of users can request access for the medical documents from the patient based on the need. Insurance companies want to check medical claims, the hospitals might want to verify bills, the doctors might want to study the previous medical history of the patient. All these users request the documents based on the need and depending on the type of the user the access is provided to these users by the patient.

These users either based on the previous treatment of the patients by them or upon knowing them by some other means use the patient username to request access to the documents. The POST call is sent to the server along with the patient's username and the requester type and the requester registration id.

The server checks for the user identity and on finding the existence of the identity in the wallet creates a gateway to invoke the chaincode, once the chaincode is invoked the endorser peer in the connection profiles executes the chaincode and verifies the credentials of the client node the request is updated in the patient world state and after that the endorsement is sent to the gateway client to ask the orderer to put it in blocks and create the next block.

Once the transaction is verified and added into the blockchain and the block is broadcasted to all the nodes , the patient receives the notification about the requester requesting for the documents.

The patient either can ignore the request or can grant access to some of the documents that he finds appropriate. After successfully selecting the list of the documents the patient makes a POST call to the server .

The server receives the parameters having the requester id, patient id and the list of the documents which the requester can access. Once the identity is checked and confirmed and the same flow for the transaction endorsement is followed and when everything is verified the transaction is being submitted with the digital signature of the patient.

The requesters are added to the *permissioned IDS* and the patient is also updated in the world state of the requester. Finally the block with this transaction is broadcasted to all the nodes in the network.

Generation of Bills and Insurance Claim

After the treatment is completed for the user the hospital is tasked with the creation of the bills based on the treatment that the patient took. The hospital uploads the bill as well as the amount in the UI and then makes an REST API call. In this case the patient and the hospital both will have access to the bills generated by the hospital.

Now as the bill is uploaded in the UI and sent to the server the multer again redirects the uploaded document to the mongo server and the file is uploaded in the *BillCollection* under the *EHR* database. The hash for the document is generated similar to the mentioned flow in the *EHR generation* . The hash value as well as the encrypted name is again stored in the ledger and the *billId* is updated in the patient's world state as well as the hospital's world state.

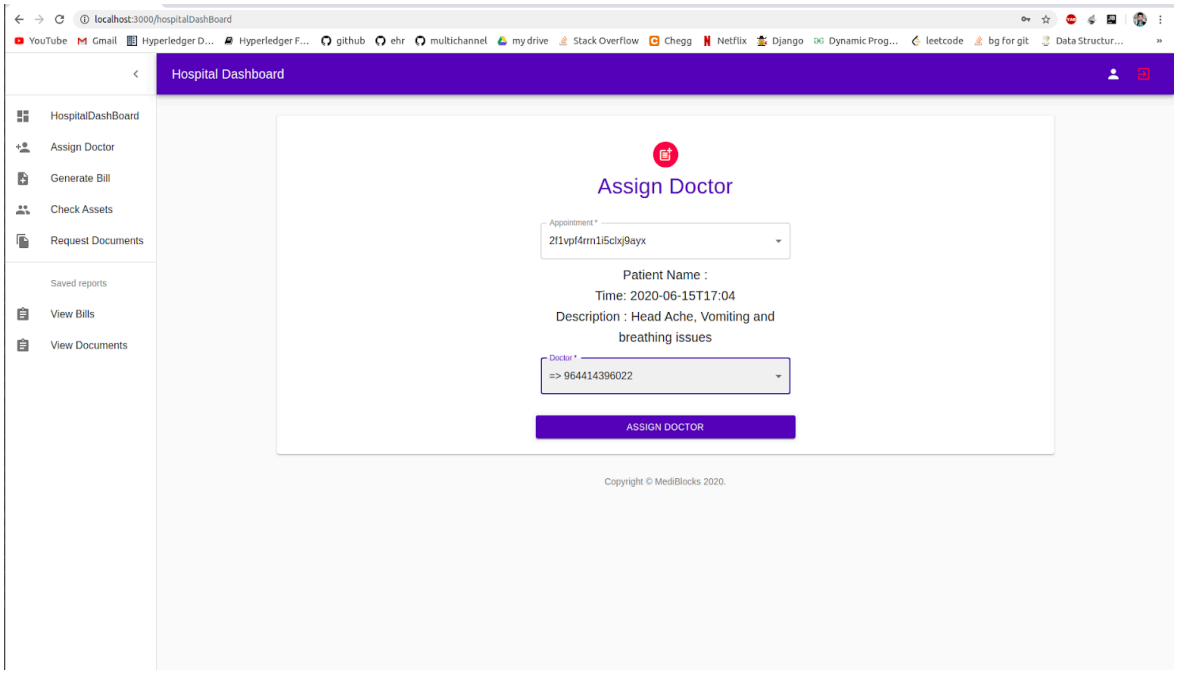
On successful endorsement the orderer orders the block and then broadcasts to all the nodes.

As soon as the patient receives the new bill he can request for the insurance claim if he had any health insurance. The patient grants access to all the relevant records as well as the bills in order to claim the medical insurance.

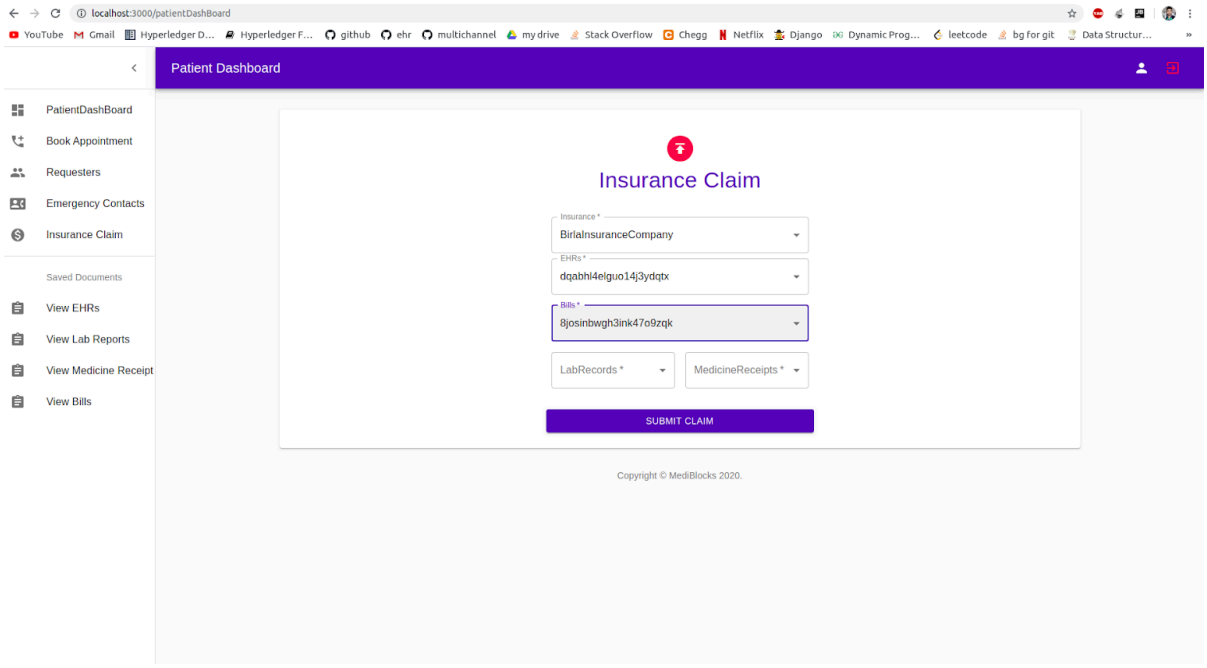
After the successful transaction to grant access to the insurance company the insurance company can view and process the claim made by the patient.

```
const FabricCAServices = require( id: 'fabric-ca-client');
const { FileSystemWallet, X509WalletMixin } = require( id: 'fabric-network');
const fs = require( id: 'fs');
const path = require( id: 'path');
const crypto = require( id: 'crypto');

const ccpPath = path.resolve(__dirname, '..', '..', '..', 'Blockchain-Network','first-network', 'connection-org1.json');
const ccpJSON = fs.readFileSync(ccpPath, options: 'utf8');
const ccp = JSON.parse(ccpJSON);

async function main() {
    try {

        console.log(ccpPath);
       // Create a new CA client for interacting with the CA.
        const caInfo = ccp.certificateAuthorities['ca.org1.example.com'];
        const caTLSCACerts = caInfo.tlsCACerts.pem;
        const ca = new FabricCAServices(caInfo.url, { trustedRoots: caTLSCACerts, verify: false }, caInfo.caName);

        // Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), '../../wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        console.log(await wallet.list());
        // Check to see if we've already enrolled the admin user.
        const adminExists = await wallet.exists( label: 'admin');
        if (adminExists) {
            console.log('An identity for the admin user "admin" already exists in the wallet');
            return;
        }

        // Enroll the admin user, and import the new identity into the wallet.
        const enrollment = await ca.enroll( req: { enrollmentID: 'admin', enrollmentSecret: 'adminpw' });
        const identity = X509WalletMixin.createIdentity( mspId: 'Org1MSP', enrollment.certificate, enrollment.key.toBytes());
        await wallet.import( label: 'admin', identity);
        console.log('Successfully enrolled admin user "admin" and imported it into the wallet');

    } catch (error) {
        console.error(`Failed to enroll admin user "admin": ${error}`);
    }
main()
```

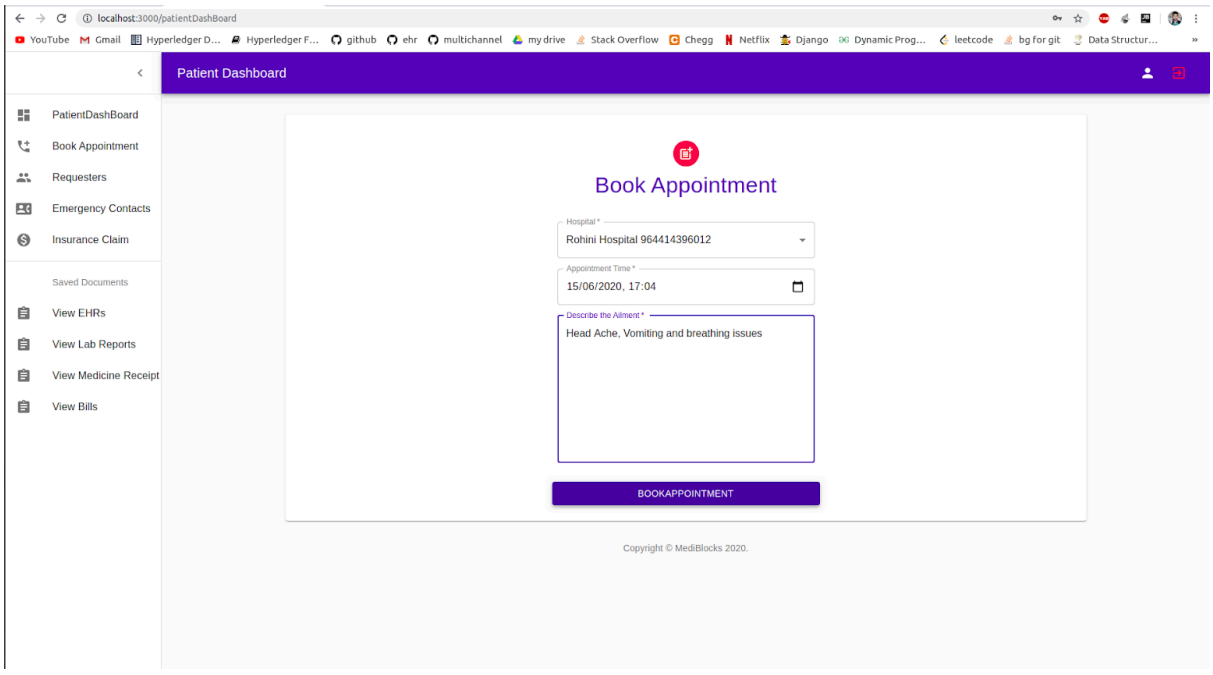
```
inal                                                                                    2 Eve
y(7) watch limit is too low. More details. (today 11:19 AM)          23:74  LF  UTF-8  4 spaces  Git: master
```

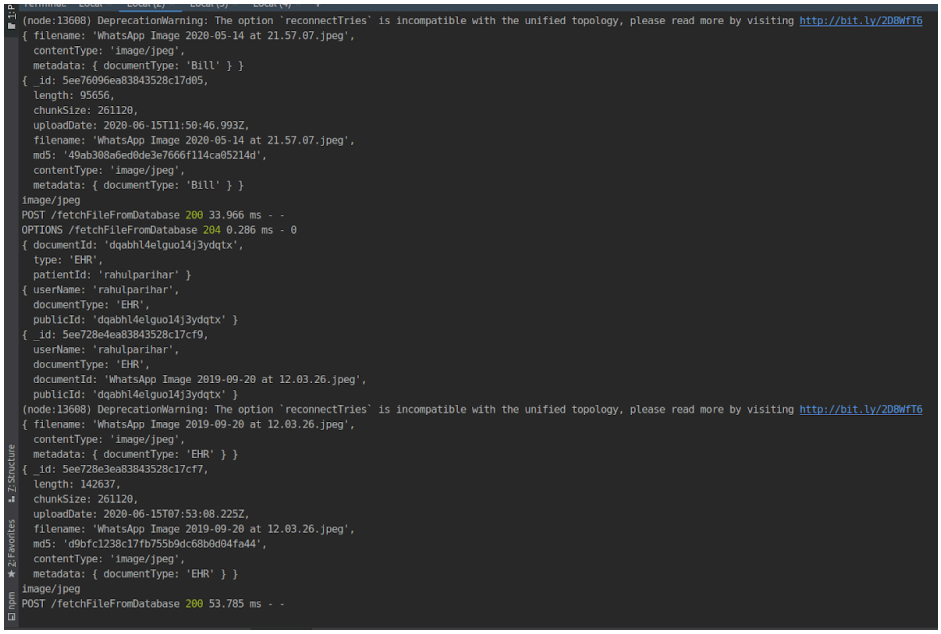


```
 */
async assignDoctor(ctx, args) {
    args = await JSON.parse(args);

    //check whether both the doctor and the appointment exists
    let doctorExists = await this.assetExists(ctx, args.doctorId);
    let appointmentExists = await this.assetExists(ctx, args.appointmentId);

    if (doctorExists && appointmentExists) {

        let doctorAsBytes = await ctx.stub.getState(args.doctorId);
        let doctor = await JSON.parse(doctorAsBytes);
        let appointmentAsBytes = await ctx.stub.getState(args.appointmentId);
        let appointment = await JSON.parse(appointmentAsBytes);
        let hospitalAsBytes = await ctx.stub.getState(args.hospitalId);
        let hospital = await JSON.parse(hospitalAsBytes);

        //update the patient and the appointment ids in the doctor's global state
        let appointments = doctor.appointments;
        appointments.push(appointment.appointmentId);
        doctor.appointments = appointments;

        await ctx.stub.putState(doctor.medicalRegistrationNo, Buffer.from(JSON.stringify(doctor)));

        //update the appointment with the doctor id and update global state
        appointment.doctorId = doctor.medicalRegistrationNo;
        await ctx.stub.putState(appointment.appointmentId, Buffer.from(JSON.stringify(appointment)));

        appointments = hospital.appointments;

        //remove that appointment from the appointment list of the hospital and update it in the global state of hospital
        let index = appointments.indexOf(appointment.appointmentId);
        if (index > -1) {
            appointments.splice(index, 1);
        }
        hospital.remainingPatientBills.push(appointment.patientId);
        await ctx.stub.putState(hospital.registrationId, Buffer.from(JSON.stringify(hospital)));

        let response = `Appointment with the appointmentId ${appointment.appointmentId} is assigned a doctor with id ${doctor.medic
        return response;
```

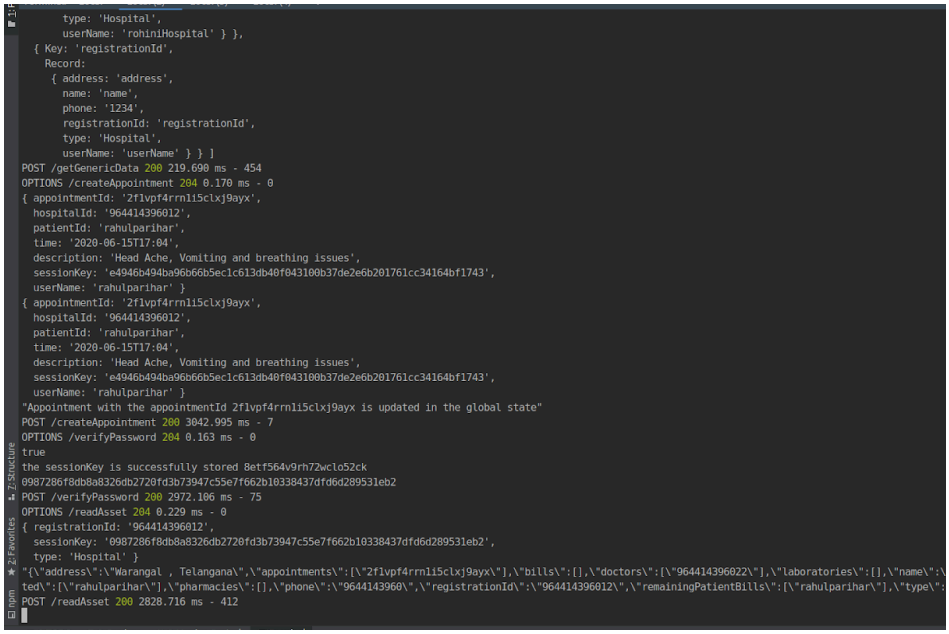


```
#
#   SECTION: Orderer
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for orderer related parameters
#
################################################################################
Orderer: &OrdererDefaults

    # Orderer Type: The orderer implementation to start
    # Available types are "solo","kafka"  and "etcdraft"
    OrdererType: solo

    Addresses:
        - orderer.example.com:7050

    # Batch Timeout: The amount of time to wait before creating a batch
    BatchTimeout: 2s

    # Batch Size: Controls the number of messages batched into a block
    BatchSize:

        # Max Message Count: The maximum number of messages to permit in a batch
        MaxMessageCount: 10

        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        AbsoluteMaxBytes: 99 MB

        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
        # the serialized messages in a batch. A message larger than the preferred
        # max bytes will result in a batch larger than preferred max bytes.
        PreferredMaxBytes: 512 KB

    Kafka:
        # Brokers: A list of Kafka brokers to which the orderer connects
        # NOTE: Use IP:port notation
        Brokers:
            - 127.0.0.1:9092

    # EtcdRaft defines configuration which must be set when the "etcdraft"
```

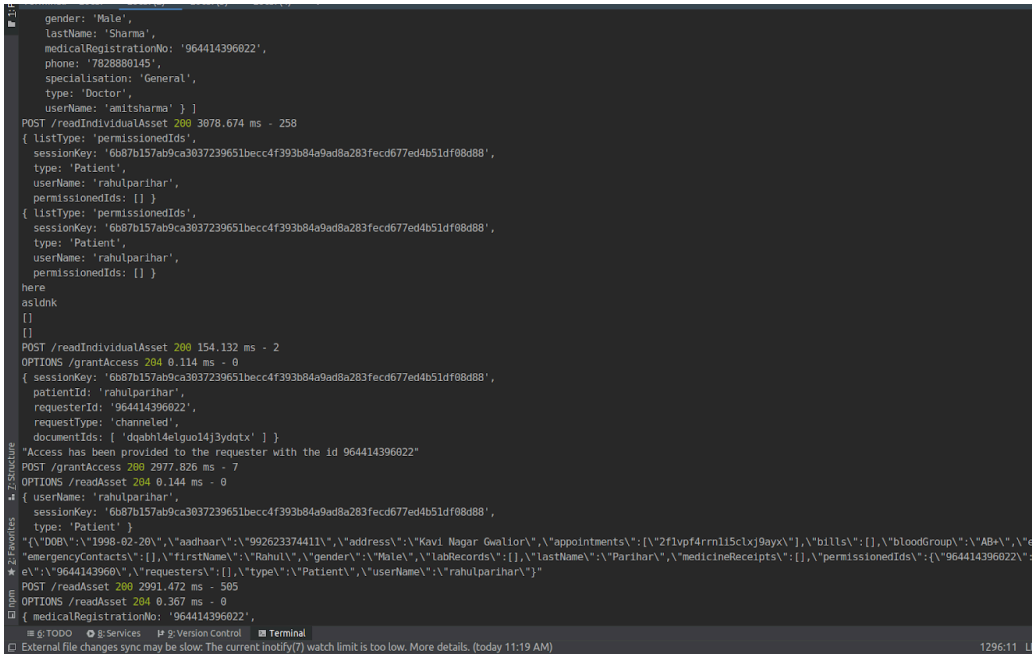


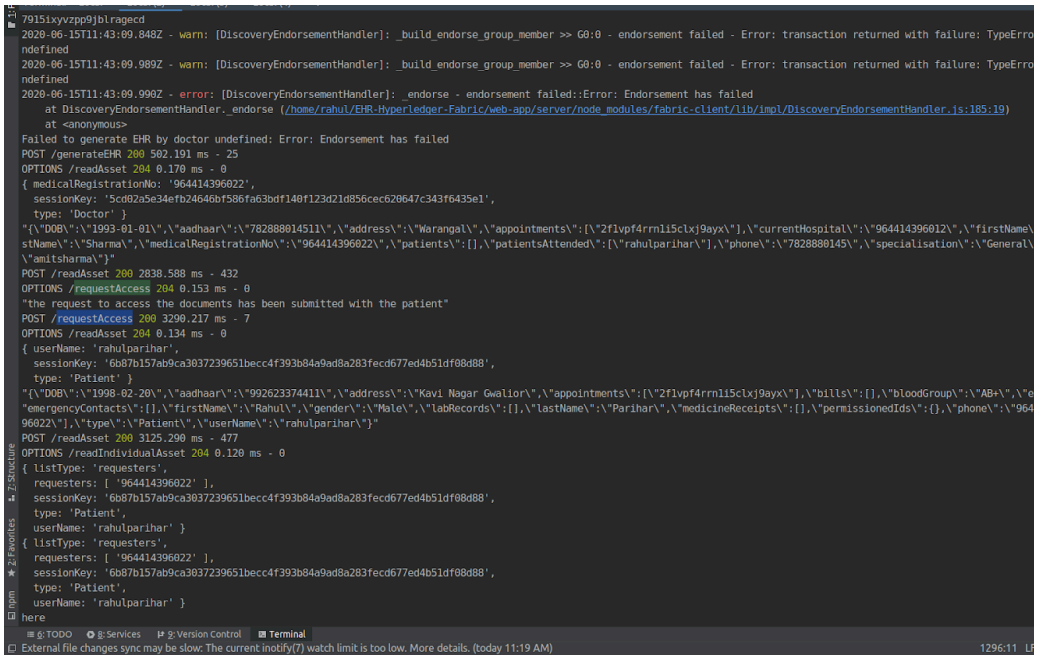


```
(node:13608) DeprecationWarning: The option `reconnectTries` is incompatible with the unified topology, please read more by visiting http://bit.ly/2D8WfT6
{ filename: 'WhatsApp Image 2020-05-14 at 21.57.07.jpeg',
  contentType: 'image/jpeg',
  metadata: { documentType: 'Bill' } }
{ _id: 5ee76096ea83843528c17d05,
  length: 95656,
  chunkSize: 261120,
  uploadDate: 2020-06-15T11:50:46.993Z,
  filename: 'WhatsApp Image 2020-05-14 at 21.57.07.jpeg',
  md5: '49ab308a6ed0de3e7666f114ca05214d',
  contentType: 'image/jpeg',
  metadata: { documentType: 'Bill' } }
image/jpeg
POST /fetchFileFromDatabase 200 33.966 ms - -
OPTIONS /fetchFileFromDatabase 204 0.286 ms - 0
{ documentId: 'dqabhl4elguo14j3ydqtx',
  type: 'EHR',
  patientId: 'rahulparihar' }
{ userName: 'rahulparihar',
  documentType: 'EHR',
  publicId: 'dqabhl4elguo14j3ydqtx' }
{ _id: 5ee728e4ea83843528c17cf9,
  userName: 'rahulparihar',
  documentType: 'EHR',
  documentId: 'WhatsApp Image 2019-09-20 at 12.03.26.jpeg',
  publicId: 'dqabhl4elguo14j3ydqtx' }
(node:13608) DeprecationWarning: The option `reconnectTries` is incompatible with the unified topology, please read more by visiting http://bit.ly/2D8WfT6
{ filename: 'WhatsApp Image 2019-09-20 at 12.03.26.jpeg',
  contentType: 'image/jpeg',
  metadata: { documentType: 'EHR' } }
{ _id: 5ee728e3ea83843528c17cf7,
  length: 142637,
  chunkSize: 261120,
  uploadDate: 2020-06-15T07:53:08.225Z,
  filename: 'WhatsApp Image 2019-09-20 at 12.03.26.jpeg',
  md5: 'd9bfc1238c17fb755b9dc68b0d04fa44',
  contentType: 'image/jpeg',
  metadata: { documentType: 'EHR' } }
image/jpeg
POST /fetchFileFromDatabase 200 53.785 ms - -
```



```
      type: 'Hospital',
      userName: 'rohiniHospital' } },
  { Key: 'registrationId',
    Record:
     { address: 'address',
       name: 'name',
       phone: '1234',
       registrationId: 'registrationId',
       type: 'Hospital',
       userName: 'userName' } } ]
POST /getGenericData 200 219.690 ms - 454
OPTIONS /createAppointment 204 0.170 ms - 0
{ appointmentId: '2f1vpf4rrn1i5clxj9ayx',
  hospitalId: '964414396012',
  patientId: 'rahulparihar',
  time: '2020-06-15T17:04',
  description: 'Head Ache, Vomiting and breathing issues',
  sessionKey: 'e4946b494ba96b66b5ec1c613db40f043100b37de2e6b201761cc34164bf1743',
  userName: 'rahulparihar' }
{ appointmentId: '2f1vpf4rrn1i5clxj9ayx',
  hospitalId: '964414396012',
  patientId: 'rahulparihar',
  time: '2020-06-15T17:04',
  description: 'Head Ache, Vomiting and breathing issues',
  sessionKey: 'e4946b494ba96b66b5ec1c613db40f043100b37de2e6b201761cc34164bf1743',
  userName: 'rahulparihar' }
"Appointment with the appointmentId 2f1vpf4rrn1i5clxj9ayx is updated in the global state"
POST /createAppointment 200 3042.995 ms - 7
OPTIONS /verifyPassword 204 0.163 ms - 0
true
the sessionKey is successfully stored 8etf564v9rh72wclo52ck
0987286f8db8a8326db2720fd3b73947c55e7f662b10338437dfd6d289531eb2
POST /verifyPassword 200 2972.106 ms - 75
OPTIONS /readAsset 204 0.229 ms - 0
{ registrationId: '964414396012',
  sessionKey: '0987286f8db8a8326db2720fd3b73947c55e7f662b10338437dfd6d289531eb2',
  type: 'Hospital' }
"{\"address\":\"Warangal , Telangana\",\"appointments\":[\"2f1vpf4rrn1i5clxj9ayx\"],\"bills\":[],\"doctors\":[\"964414396022\"],\"laboratories\":[],\"name\":\
ted\":[\"rahulparihar\"],\"pharmacies\":[],\"phone\":\"9644143960\",\"registrationId\":\"964414396012\",\"remainingPatientBills\":[\"rahulparihar\"],\"type\":
POST /readAsset 200 2828.716 ms - 412
```


gender: 'Male',
lastName: 'Sharma',
medicalRegistrationNo: '964414396022',
phone: '7828880145',
specialisation: 'General',
type: 'Doctor',
userName: 'amitsharma' }]
POST /readIndividualAsset 200 3078.674 ms - 258
{ listType: 'permissionedIds',
sessionKey: '6b87b157ab9ca3037239651becc4f393b84a9ad8a283fecd677ed4b51df08d88',
type: 'Patient',
userName: 'rahulparihar',
permissionedIds: [] }
{ listType: 'permissionedIds',
sessionKey: '6b87b157ab9ca3037239651becc4f393b84a9ad8a283fecd677ed4b51df08d88',
type: 'Patient',
userName: 'rahulparihar',
permissionedIds: [] }
here
asldnk
[]
[]
POST /readIndividualAsset 200 154.132 ms - 2
OPTIONS /grantAccess 204 0.114 ms - 0
{ sessionKey: '6b87b157ab9ca3037239651becc4f393b84a9ad8a283fecd677ed4b51df08d88',
patientId: 'rahulparihar',
requesterId: '964414396022',
requestType: 'channeled',
documentIds: ['dqabhl4elguol4j3ydqtx'] }
"Access has been provided to the requester with the id 964414396022"
POST /grantAccess 200 2977.826 ms - 7
OPTIONS /readAsset 204 0.144 ms - 0
{ userName: 'rahulparihar',
sessionKey: '6b87b157ab9ca3037239651becc4f393b84a9ad8a283fecd677ed4b51df08d88',
type: 'Patient' }
"{\"DOB\":\"1998-02-20\",\"aadhaar\":\"992623374411\",\"address\":\"Kavi Nagar Gwalior\",\"appointments\":[\"2f1vpf4rrn1i5clxj9ayx\"],\"bills\":[],\"bloodGroup\":\"AB+\",\"e
"emergencyContacts\":[],\"firstName\":\"Rahul\",\"gender\":\"Male\",\"labRecords\":[],\"lastName\":\"Parihar\",\"medicineReceipts\":[],\"permissionedIds\":{\"964414396022\"
e\":\"9644143960\",\"requesters\":[],\"type\":\"Patient\",\"userName\":\"rahulparihar\"}"
POST /readAsset 200 2991.472 ms - 505
OPTIONS /readAsset 204 0.367 ms - 0
{ medicalRegistrationNo: '964414396022',
6: TODO
8: Services
9: Version Control
Terminal
External file changes sync may be slow: The current inotify(7) watch limit is too low. More details. (today 11:19 AM)
1296:11

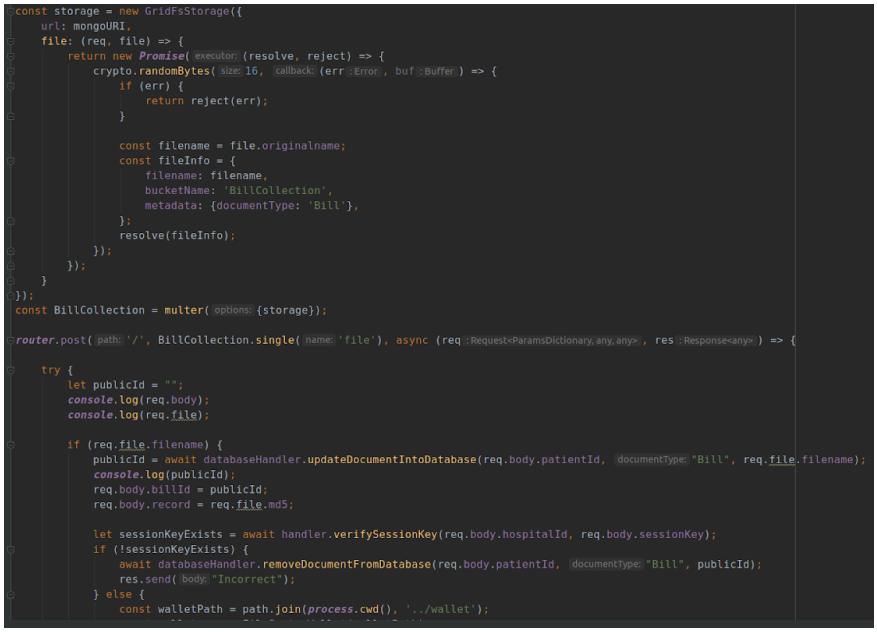

7915ixyvzpp9jblragecd
2020-06-15T11:43:09.848Z - warn: [DiscoveryEndorsementHandler]: _build_endorse_group_member >> G0:0 - endorsement failed - Error: transaction returned with failure: TypeErro
ndefined
2020-06-15T11:43:09.989Z - warn: [DiscoveryEndorsementHandler]: _build_endorse_group_member >> G0:0 - endorsement failed - Error: transaction returned with failure: TypeErro
ndefined
2020-06-15T11:43:09.990Z - error: [DiscoveryEndorsementHandler]: _endorse - endorsement failed::Error: Endorsement has failed
at DiscoveryEndorsementHandler._endorse (/home/rahul/EHR-Hyperledger-Fabric/web-app/server/node_modules/fabric-client/lib/impl/DiscoveryEndorsementHandler.js:185:19)
at <anonymous>
Failed to generate EHR by doctor undefined: Error: Endorsement has failed
POST /generateEHR 200 502.191 ms - 25
OPTIONS /readAsset 204 0.170 ms - 0
{ medicalRegistrationNo: '964414396022',
sessionKey: '5cd02a5e34efb24646bf586fa63bdf140f123d21d856cec620647c343f6435e1',
type: 'Doctor' }
"{\"DOB\":\"1993-01-01\",\"aadhaar\":\"782888014511\",\"address\":\"Warangal\",\"appointments\":[\"2f1vpf4rrn1i5clxj9ayx\"],\"currentHospital\":\"964414396012\",\"firstName\
stName\":\"Sharma\",\"medicalRegistrationNo\":\"964414396022\",\"patients\":[],\"patientsAttended\":[\"rahulparihar\"],\"phone\":\"7828880145\",\"specialisation\":\"General\
\"amitsharma\"}"
POST /readAsset 200 2838.588 ms - 432
OPTIONS /requestAccess 204 0.153 ms - 0
"the request to access the documents has been submitted with the patient"
POST /requestAccess 200 3290.217 ms - 7
OPTIONS /readAsset 204 0.134 ms - 0
{ userName: 'rahulparihar',
sessionKey: '6b87b157ab9ca3037239651becc4f393b84a9ad8a283fecd677ed4b51df08d88',
type: 'Patient' }
"{\"DOB\":\"1998-02-20\",\"aadhaar\":\"992623374411\",\"address\":\"Kavi Nagar Gwalior\",\"appointments\":[\"2f1vpf4rrn1i5clxj9ayx\"],\"bills\":[],\"bloodGroup\":\"AB+\",\"e
"emergencyContacts\":[],\"firstName\":\"Rahul\",\"gender\":\"Male\",\"labRecords\":[],\"lastName\":\"Parihar\",\"medicineReceipts\":[],\"permissionedIds\":{},\"phone\":\"964
96022\"],\"type\":\"Patient\",\"userName\":\"rahulparihar\"}"
POST /readAsset 200 3125.290 ms - 477
OPTIONS /readIndividualAsset 204 0.120 ms - 0
{ listType: 'requesters',
requesters: ['964414396022'],
sessionKey: '6b87b157ab9ca3037239651becc4f393b84a9ad8a283fecd677ed4b51df08d88',
type: 'Patient',
userName: 'rahulparihar' }
{ listType: 'requesters',
requesters: ['964414396022'],
sessionKey: '6b87b157ab9ca3037239651becc4f393b84a9ad8a283fecd677ed4b51df08d88',
type: 'Patient',
userName: 'rahulparihar' }
here
6: TODO
8: Services
9: Version Control
Terminal
External file changes sync may be slow: The current inotify(7) watch limit is too low. More details. (today 11:19 AM)
1296:11

```
        // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccpPath, options: {
            wallet,
            identity: req.body.userName,
            discovery: {enabled: true, asLocalhost: true}
        });

        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork( channelName: 'mychannel');

        // Get the contract from the network.
        const contract = network.getContract( chaincodeId: 'EHR');

        // Submit the specified transaction.
        console.log(req.body);
        let response = await contract.submitTransaction( name: 'createAppointment', JSON.stringify(req.body));
        response = JSON.stringify(response.toString());
        console.log(response);

        // Disconnect from the gateway.
        await gateway.disconnect();

        res.send( body: "Correct");
    }


} catch (error) {
    console.error(`Failed to create Appointment for the user  ${req.body.id}: ${error}`);
    res.send( body: "Failed to create Appointment");
}
});


module.exports = router;
```

```
async createEhr(ctx, args) {
    args = await JSON.parse(args);
    let hospitalExists = await this.assetExists(ctx, args.hospitalId);
    let patientExists = await this.assetExists(ctx, args.patientId);
    let doctorExists = await this.assetExists(ctx, args.doctorId);
    let appointmentExists = await this.assetExists(ctx, args.appointmentId);

    if (hospitalExists && doctorExists && patientExists && appointmentExists) {

        //create a new EHR and update it in the world state
        let newEHR = await new EHR(args.ehrId, args.patientId, args.doctorId, args.hospitalId, args.record, args.time);
        await ctx.stub.putState(newEHR.ehrId, Buffer.from(JSON.stringify(newEHR)));

        //update the EHR in the list of the ehrs for the patient and remove the appointment from the patient global state
        let patientAsBytes = await ctx.stub.getState(args.patientId);
        let patient = JSON.parse(patientAsBytes);
        let appointments = patient.appointments;
        let index = appointments.indexOf(args.appointmentId);
        if (index > -1) {
            appointments.splice(index, 1);
            patient.appointments = appointments;
        }
        let ehrs = patient.ehrs;
        ehrs.push(newEHR.ehrId);
        patient.ehrs = ehrs;
        await ctx.stub.putState(patient.userName, Buffer.from(JSON.stringify(patient)));

        //update the doctor with tha appointments and make the patient marked Attended
        let doctorAsBytes = await ctx.stub.getState(args.doctorId);
        let doctor = JSON.parse(doctorAsBytes);
        appointments = doctor.appointments;
        index = appointments.indexOf(args.appointmentId);
        if (index > -1) {
            appointments.splice(index, 1);
            patient.appointments = appointments;
        }

        let patientsAttended = doctor.patientsAttended;
        index = patientsAttended.indexOf(args.patientId);
```

```
async createPatient(ctx, args) {
    args = await JSON.parse(args);
    let patientExists = await this.assetExists(ctx, args.userName);
    if (!patientExists) {
        let permissionedIds = {};
        let emergencyContacts = [];
        let ehrs = [];
        let requesters = [];
        let bills = [];
        let labRecords = [];
        let medicineReceipts = [];
        let appointments = [];

        let newPatient = await new Patient(args.firstName, args.lastName, args.address, args.aadhaar, args.DOB, args.gender, args.bloodGroup,

        newPatient.permissionedIds = permissionedIds;
        newPatient.emergencyContacts = emergencyContacts;
        newPatient.ehrs = ehrs;
        newPatient.requesters = requesters;
        newPatient.bills = bills;
        newPatient.medicineReceipts = medicineReceipts;
        newPatient.labRecords = labRecords;
        newPatient.appointments = appointments;

        await ctx.stub.putState(newPatient.userName, Buffer.from(JSON.stringify(newPatient)));

        let response = `Patient with username ${newPatient.userName} is updated in the world state`;
        return response;
    } else {
        throw new Error(`Patient with username ${args.userName} already exists`);
    }

}
```

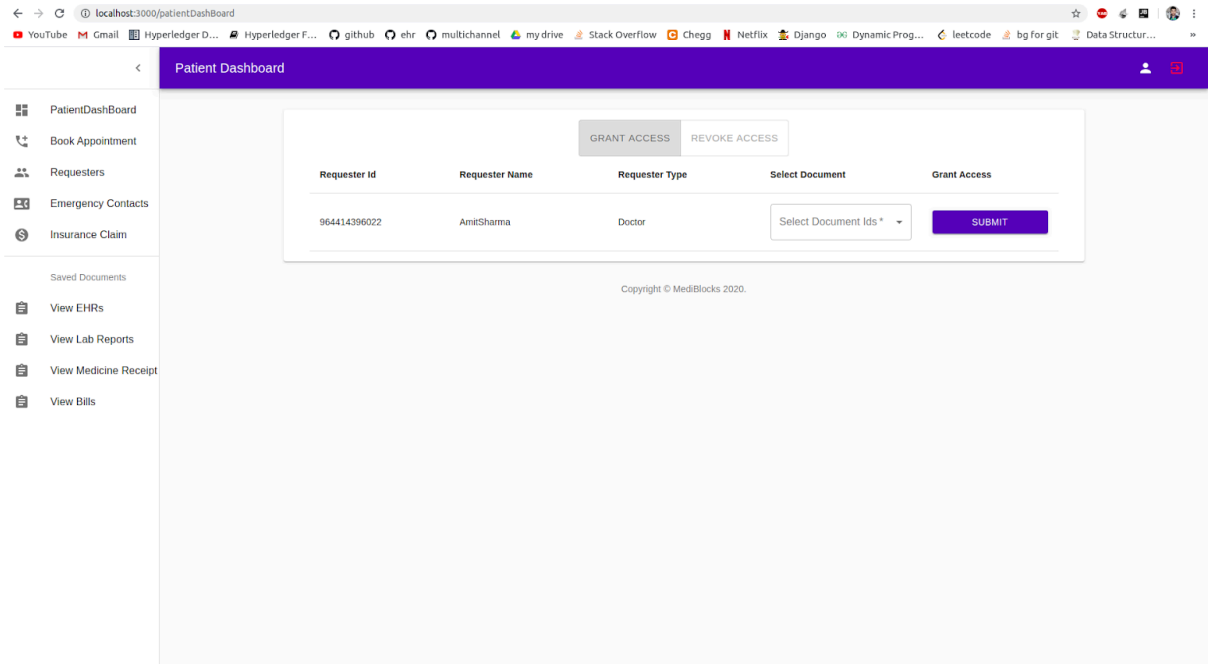
```
    * @param args
    * @returns {Promise<string>}
    */
    async generateBill(ctx, args) {
        args = await JSON.parse(args);
        let hospitalExists = await this.assetExists(ctx, args.hospitalId);
        let patientExists = await this.assetExists(ctx, args.patientId);
        let doctorExists = await this.assetExists(ctx, args.doctorId);
        let laboratoryExists = await this.assetExists(ctx, args.laboratoryId);
        let pharmacyExists = await this.assetExists(ctx, args.pharmacyId);

        if (hospitalExists && patientExists && doctorExists) {

            if (!laboratoryExists) {
                args.laboratoryId = '';
            }
            if (!pharmacyExists) {
                args.pharmacyId = '';
            }
            //generate a new bill with all the details
            let newBill = await new Bill(args.billId, args.hospitalId, args.patientId, args.doctorId, args.laboratoryId, args.pharmacyId, args.ti
            await ctx.stub.putState(newBill.billId, Buffer.from(JSON.stringify(newBill)));

            //update the patient with the bill
            let patientAsBytes = await ctx.stub.getState(args.patientId);
            let patient = JSON.parse(patientAsBytes);
            let bills = patient.bills;
            bills.push(newBill.billId);
            patient.bills = bills;
            await ctx.stub.putState(patient.userName, Buffer.from(JSON.stringify(patient)));

            //update the patient with the bill
            let hospitalAsBytes = await ctx.stub.getState(args.hospitalId);
            let hospital = JSON.parse(hospitalAsBytes);
            let index = hospital.remainingPatientBills.indexOf(args.patientId);
            if (index > -1) {
                hospital.remainingPatientBills.splice(index, 1);
            }
            bills = hospital.bills;
            bills.push(newBill.billId);
            hospital.bills = bills;
```

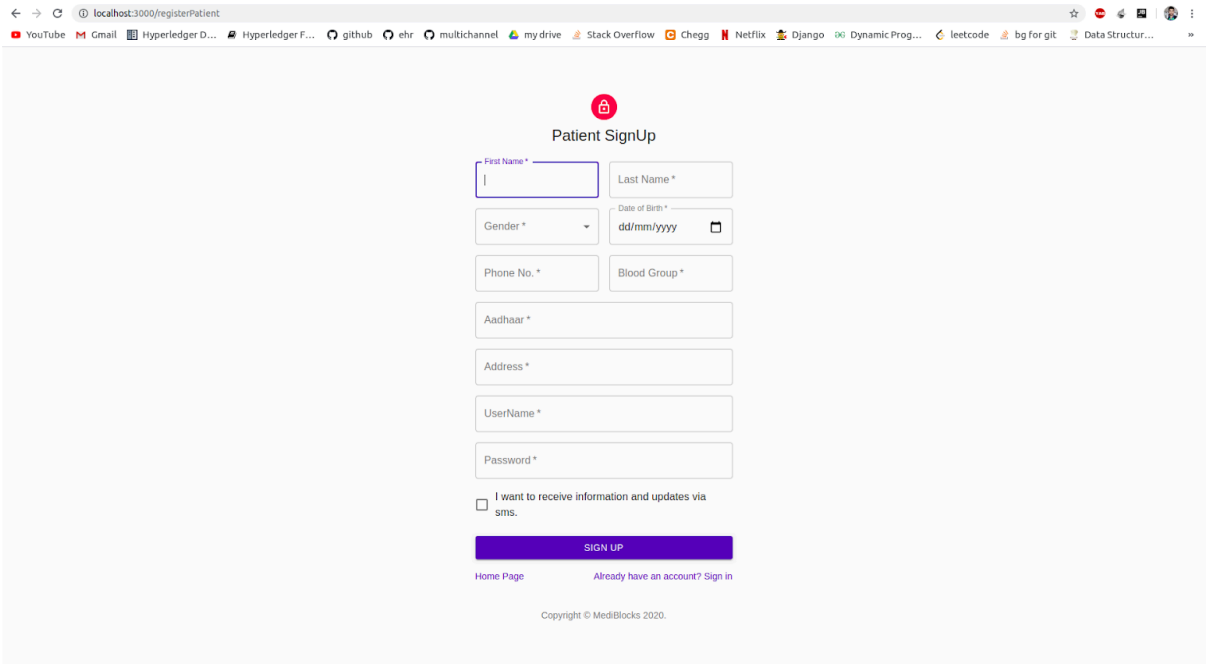


```
const storage = new GridFsStorage({
    url: mongoURI,
    file: (req, file) => {
        return new Promise( executor: (resolve, reject) => {
            crypto.randomBytes( size: 16,  callback: (err :Error , buf :Buffer ) => {
                if (err) {
                    return reject(err);
                }

                const filename = file.originalname;
                const fileInfo = {
                    filename: filename,
                    bucketName: 'BillCollection',
                    metadata: {documentType: 'Bill'},
                };
                resolve(fileInfo);
            });
        });
    }
});
const BillCollection = multer( options: {storage});

router.post( path: '/', BillCollection.single( name: 'file'), async (req :Request<ParamsDictionary, any, any> , res :Response<any> ) => {

    try {
        let publicId = "";
        console.log(req.body);
        console.log(req.file);

        if (req.file.filename) {
            publicId = await databaseHandler.updateDocumentIntoDatabase(req.body.patientId,  documentType: "Bill", req.file.filename);
            console.log(publicId);
            req.body.billId = publicId;
            req.body.record = req.file.md5;

            let sessionKeyExists = await handler.verifySessionKey(req.body.hospitalId, req.body.sessionKey);
            if (!sessionKeyExists) {
                await databaseHandler.removeDocumentFromDatabase(req.body.patientId,  documentType: "Bill", publicId);
                res.send( body: "Incorrect");
            } else {
                const walletPath = path.join(process.cwd(), '../wallet');
```

```
const storage = new GridFsStorage({
    url: mongoURI,
    file: (req, file) => {
        return new Promise( executor: (resolve, reject) => {
            crypto.randomBytes( size: 16,  callback: (err :Error , buf :Buffer ) => {
                if (err) {
                    return reject(err);
                }

                const filename = file.originalname;
                const fileInfo = {
                    filename: filename,
                    bucketName: 'EHRCollection',
                    metadata: {documentType: 'EHR'},
                };
                resolve(fileInfo);
            });
        });
    }
});
const EHRCollection = multer( options: {storage});

router.post( path: '/', EHRCollection.single( name: 'file'), async (req :Request<ParamsDictionary, any, any> , res :Response<any> ) => {

    try {
        let publicId = "";
        console.log(req.body);
        console.log(req.file);

        if (req.file.filename) {
            publicId = await databaseHandler.updateDocumentIntoDatabase(req.body.patientId,  documentType: "EHR", req.file.filename);
            console.log(publicId);
            req.body.ehrId = publicId;
            req.body.record = req.file.md5;

            let sessionKeyExists = await handler.verifySessionKey(req.body.doctorId, req.body.sessionKey);
            if (!sessionKeyExists) {
                await databaseHandler.removeDocumentFromDatabase(req.body.patientId,  documentType: "EHR", publicId);
                res.send( body: "Incorrect");
            } else {
```

```
    * @param args
    * @returns {Promise<string>}
    */
    async grantAccess(ctx, args) {
        args = await JSON.parse(args);
        let patientExists = await this.assetExists(ctx, args.patientId);
        let requesterExists = await this.assetExists(ctx, args.requesterId);
        if (patientExists && requesterExists) {
            let patientAsBytes = await ctx.stub.getState(args.patientId);
            let patient = JSON.parse(patientAsBytes);
            let requesters = patient.requesters;
            let index = requesters.indexOf(args.requesterId);
            if (index > -1) {
                //remove the requester from the requester array and put it into the permissioned Ids with the list of all the
                //document ids that particular can access
                requesters.splice(index, 1);
                patient.requesters = requesters;
                let permissionedIds = patient.permissionedIds;
                permissionedIds[args.requesterId] = args.documentIds;
                patient.permissionedIds = permissionedIds;
                await ctx.stub.putState(patient.userName, Buffer.from(JSON.stringify(patient)));

                //update the patient in the patient array for the requester
                let requesterAsBytes = await ctx.stub.getState(args.requesterId);
                let requester = JSON.parse(requesterAsBytes);
                let patients = requester.patients;
                index = patients.indexOf(args.patientId);
                if (index < 0) {
                    patients.push(args.patientId);
                }
                requester.patients = patients;
                await ctx.stub.putState(args.requesterId, Buffer.from(JSON.stringify(requester)));

                let response = `Access has been provided to the requester with the id ${args.requesterId}`;
                return response;
            } else {
                throw new Error(`No such requester with id ${args.requesterId}`);
            }
        } else {
            throw new Error(`patient with id ${args.patientId} or requester with id ${args.requesterId} doesn't exists`);
        }
```



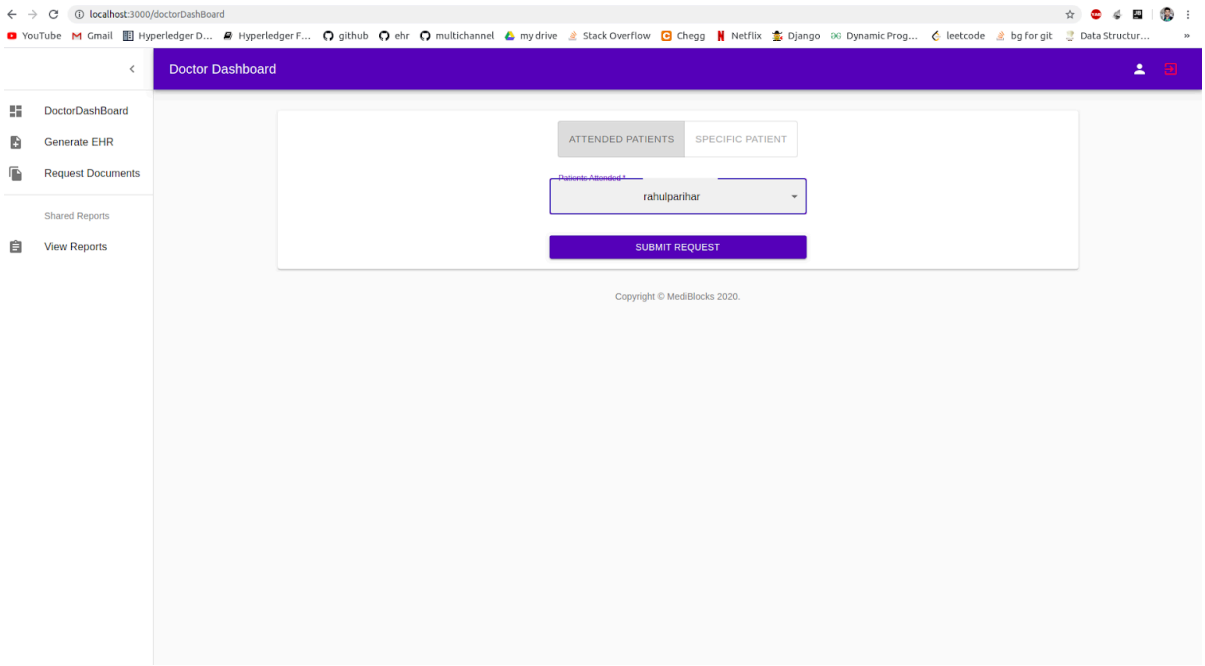



```
async readHospitalAssets(ctx, args) {
    args = await JSON.parse(args);
    let hospitalExists = await this.assetExists(ctx, args.hospitalId);
    if (hospitalExists) {
        let hospitalAsBytes = await ctx.stub.getState(args.hospitalId);
        let hospital = JSON.parse(hospitalAsBytes);

        let assetExits = await this.assetExists(ctx, args.assetId);
        if (assetExits) {
            let index = -1;
            if (args.listType === 'patients') {
                index = hospital.patients.indexOf(args.assetId);
                if (index > -1) {
                    return await this.readDocuments(ctx, args.assetId, hospital.registrationId);
                }
            } else {
                if (args.listType === 'patientsVisited') {
                    index = hospital.patientsVisited.indexOf(args.assetId);
                } else if (args.listType === 'doctors') {
                    index = hospital.doctors.indexOf(args.assetId);
                } else if (args.listType === 'pharmacies') {
                    index = hospital.pharmacies.indexOf(args.assetId);
                } else if (args.listType === 'laboratories') {
                    index = hospital.laboratories.indexOf(args.assetId);
                } else if (args.listType === 'bills') {
                    index = hospital.bills.indexOf(args.assetId);
                } else if (args.listType === 'appointments') {
                    index = hospital.appointments.indexOf(args.assetId);
                }
                if (index > -1) {
                    return await this.modifyAssetInfo(ctx, args);
                } else {
                    throw new Error(`asset not found`);
                }
            }
        } else {
            throw new Error(`the asset ${args.assetId} is not part of the hospital list`);
        }
    }
}
```

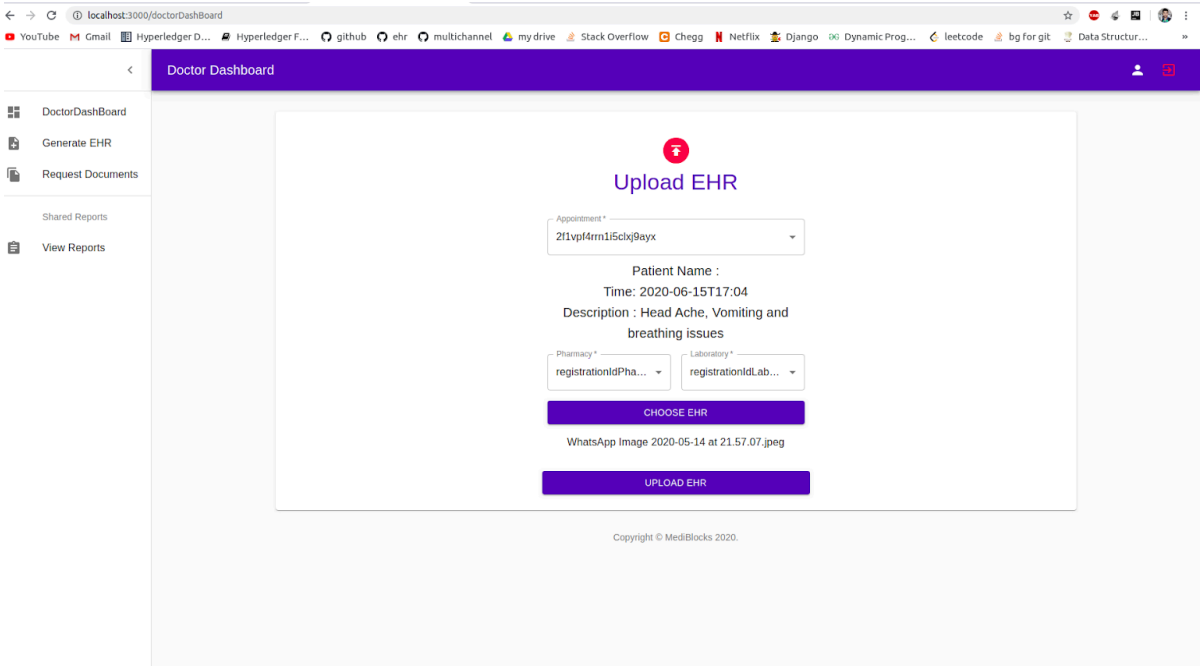
```
    const userExists = await wallet.exists(req.body.userName);
    if (userExists) {
        res.send( body: 'Candidate has been already registered ... ');
        return;
    }

    // Check to see if we've already enrolled the admin user.
    const adminExists = await wallet.exists( label: 'admin');
    if (!adminExists) {
        res.send( body: ' Admin is not currently enrolled. Please wait for sometime ...');
        console.log('Please run enrollAdmin.js file first ... ');
        return;
    }

    // Create a new gateway for connecting to our peer node.
    const gateway = new Gateway();
    await gateway.connect(ccpPath,  options: {wallet, identity: 'admin', discovery: {enabled: true, asLocalhost: true}});

    // Get the CA client object from the gateway for interacting with the CA.
    const ca = gateway.getClient().getCertificateAuthority();
    const adminIdentity = gateway.getCurrentIdentity();

    // console.log(JSON.parse(adminIdentity.toString()));
    // Register the user, enroll the user, and import the new identity into the wallet.
    const secret = await ca.register( req: {
        affiliation: 'org1.department1',
        enrollmentID: req.body.userName,
        role: 'client'
    }, adminIdentity);
    const enrollment = await ca.enroll( req: {enrollmentID: req.body.userName, enrollmentSecret: secret});

    // console.log(JSON.parse(enrollment.toString()));

    const userIdentity = X509WalletMixin.createIdentity( mspid: 'Org1MSP', enrollment.certificate, enrollment.key.toBytes());

    await wallet.import(req.body.userName, userIdentity);

    gateway.disconnect();

    let response = await registerInLedger(req);
```

```
        return response;

    }
    throw new Error(`Either the patient or hospital entity is not correct`);
}

/**
 *
 * @param ctx
 * @param args
 * @returns {Promise<string>}
 */
async requestAccess(ctx, args) {
    args = await JSON.parse(args);
    let requesterExists = await this.assetExists(ctx, args.requesterId);
    let patientExists = await this.assetExists(ctx, args.patientId);
    if (requesterExists && patientExists) {

        //get the patient and update the request for that patient
        let patientAsBytes = await ctx.stub.getState(args.patientId);
        let patient = JSON.parse(patientAsBytes);
        let requesters = patient.requesters;
        let index = requesters.indexOf(args.requesterId);
        if (index < 0) {
            requesters.push(args.requesterId);
        }
        patient.requesters = requesters;

        await ctx.stub.putState(patient.userName, Buffer.from(JSON.stringify(patient)));

        let response = `the request to access the documents has been submitted with the patient`;
        return response;
    } else {
        throw new Error(`this requester with id ${args.requesterId} or the patient with id ${args.patientId} doesn't exist`);
    }
}

/**
 *
 * @param ctx
 * @param args
```

```
    try {
        let sessionKeyExists = await handler.verifySessionKey(req.body.requesterId, req.body.sessionKey);
        if (!sessionKeyExists) {
            res.send( body: "Incorrect");
        } else {
            const walletPath = path.join(process.cwd(), '../wallet');
            const wallet = new FileSystemWallet(walletPath);

            // Create a new gateway for connecting to our peer node.
            const gateway = new Gateway();
            await gateway.connect(ccpPath, options: {
                wallet,
                identity: req.body.requesterId,
                discovery: {enabled: true, asLocalhost: true}
            });

            // Get the network (channel) our contract is deployed to.
            const network = await gateway.getNetwork( channelName: 'mychannel');

            // Get the contract from the network.
            const contract = network.getContract( chaincodeId: 'EHR');

            // Submit the specified transaction.
            let response = await contract.submitTransaction( name: 'requestAccess', JSON.stringify(req.body));
            response = JSON.stringify(response.toString());
            console.log(response);

            // Disconnect from the gateway.
            await gateway.disconnect();

            res.send( body: "Correct");
        }
    } catch (error) {
        console.error(`Failed to request for documents for patient ${req.body.patientId}: ${error}`);
        res.send( body: "Failed to request for access");
    }
});
```



```
/**
 *
 * @param ctx
 * @param args
 * @returns {Promise<string>}
 */
async revokeAccess(ctx, args) {
    args = await JSON.parse(args);
    let patientExists = await this.assetExists(ctx, args.patientId);
    let requesterExists = await this.assetExists(ctx, args.requesterId);
    if (patientExists && requesterExists) {
        let patientAsBytes = await ctx.stub.getState(args.patientId);
        let patient = JSON.parse(patientAsBytes);
        let permissionedIds = patient.permissionedIds;

        //check whether that id is present in the permissionedIds or not and if yes remove that entry
        if (args.requesterId in permissionedIds) {
            delete permissionedIds[args.requesterId];
            patient.permissionedIds = permissionedIds;
            await ctx.stub.putState(patient.userName, Buffer.from(JSON.stringify(patient)));

            //Delete the patient from the patient array for the requester
            let requesterAsBytes = await ctx.stub.getState(args.requesterId);
            let requester = JSON.parse(requesterAsBytes);
            let patients = requester.patients;
            let index = patients.indexOf(args.patientId);
            if (index > -1) {
                patients.splice(index, 1);
                requester.patients = patients;
                await ctx.stub.putState(args.requesterId, Buffer.from(JSON.stringify(requester)));
            }

            let response = `Access has been revoked for the requester with the id ${args.requesterId}`;
            return response;
        } else {
            throw new Error(`No such Permissioned id with id ${args.requesterId}`);
        }
    } else {
        throw new Error(`patient with id ${args.patientId} or requester with id ${args.requesterId} doesn't exists`);
    }
}
```



CONCLUSIONS

This work exploits the main healthcare scenario of managing health information and proposed an architecture for exchanging data in a secured manner.In our proposal,all healthcare data is owned by the patients.It depends on blockchain technology and offline storage services that are well distributed to obtain security, fault tolerance and improved trust. The architecture that we provided covers the flow of the subsequent work that will be carried out and furthermore all the internal functioning of the blockchain as well.

Implementation of the EHR system with proposed architecture using blockchain technology and performance testing of the system ensures that all the features mentioned are being provided. The web based application exploits the blockchain network technology to provide all the mentioned features while storing the electronic health records in a decentralized manner.
The complete supply chain for the medical treatment can be established through this application. From booking an appointment in the hospital to going to a doctor for treatment and getting your medical documents uploaded. The access control of the documents and then sharing them to your trusted contacts.
The app brings all the participants in the medical supply chain together in order to create a secure environment for sharing your documents as well as protecting your private health data from going into the wrong hands. The trusted contacts in the network ensures that the security, privacy can be established easily.

REFERENCES

- Setting up the Evironment : https://www.srcmake.com/home/fabric
- Official documentation of hyperledger fabric https://www.hyperledger.org/projects/fabric
- Secure and Trustable Electronic Medical Records Sharing using Blockchain https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5977675/
- H. Kim, H. Song, S. Lee, H. Kim, and I. Song, "A simple approach to share users' own healthcare data with a mobile phone," in Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on. IEEE, 2016, pp. 453–455.
- A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in Open and Big Data (OBD), International Conference on. IEEE, 2016, pp. 25–30.
- Anastasia Theodouli, Stelios Arakliotis, Konstantinos Moschou, Konstantinos Votis, Dimitrios Tzovaras, 2018, 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications
- RBAC-SC: Role-based Access Control Using Smart Contracts https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8307397
- Eletronic Health Records using Blockchain Technology https://arxiv.org/pdf/1804.10078.pdf
- Electronic Record Storage using Blockchain github.com/amtj/EHR-Storage-using-Blockchain/blob/master/Detailed%20report.pdf
- Medical Chain Whitepaper https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf
- On the design of a Blockchain-based system to facilitate Healthcare Data Sharing https://ieeexplore.ieee.org/document/8456059
- Metrics for Assessing Blockchain-based Healthcare Decentralized Apps https://www.dre.vanderbilt.edu/~schmidt/PDF/IEEE-Healthcom-2017.pdf
- Karalee Close, Emily Serazin, Alexander Aboshiha, Amy Hurwitz, Lise Lørup, and Nayel Hakim, Architecture of the Hyperledger Blockchain Fabric Christian Cachin IBM Research - Zurich CH-8803 Rüschlikon, Switzerland cca@zurich.ibm.com July 2016
- A PRESCRIPTION FOR BLOCKCHAIN IN HEALTH CARE
- MediChainTM: A Secure Decentralized Medical Data Asset Management System https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8689026
- Integrating Blockchain for Data Sharing and Collaboration in Mobile Healthcare Applications IEEE Research paper https://ieeexplore.ieee.org/document/8292361
- Applications of Blockchain in Healthcare: Current Landscape & Challenges https://www.researchgate.net/publication/329525760_Applications_of_Blockchain_in_Healthcare_Current_Landscape_Challenges