Frankfurt University of Applied Science

–Dept. of Computer Science and Engineering–

Patient Data Management System Using Hyper Ledger Fabric

High Integrity Systems Project Final Report
Master of Science (M.Sc.)

Presented By, **Team Titanium**

Shubham Girdhar - **Matriculation Number**: 1323003
Vineeth Bhat - **Matriculation Number**: 1322415
Faraz Shamim - **Matriculation Number**: 1294679
Md Towfic Aziz - **Matriculation Number**: 1205859

Referent : Prof. Dr. Martin Kappas

DECLARATION

We hereby confirm that we have written the present work independently and have not used any other sources than those given in the bibliography.

All passages that are taken verbatim or correspondingly from published or not yet published sources are marked as such.

The drawings or images in this work were created by myself or provided with a corresponding source reference.

This work has not been submitted to any other examination authority in the same or a similar form.

Frankfurt, 31. March 2021

Team Titanium

ABSTRACT

A blockchain is an ever growing list of records that are linked to each other in a distributed network. These linked records called ledgers are immutable in nature providing resistance to change. Blockchain provides a secure way of processing the data in a distributed environment. It was widely involved in cryptocurrencies in the earlier days and however its application in bitcoin motivated and inspired other applications to adapt its concepts. Its application in healthcare requires blockchain to be highly secure, provide a more trusted environment than the traditional blockchain, that is by design should be an enterprise level blockchain by restricting access to the public. Hyperledger Fabric caters to all these requirements in providing a secure and distributed environment for healthcare systems. In healthcare there are a lot of fields where Hyperledger Fabric can be adopted, but the focus here is given to management of patient's medical records. Traditionally the medical records are either stored centrally in a database that is accessible to only the hospitals owning it, this creates a various of problems for patients. Here adopting Hyperledger Fabric could pave way to much smoother transactions for patients and the hospitals involved. The aim is to consider the actual scenario of how the records are handled, how the patient will interact in the real world and design a system using Hyperledger Fabric to tackle major problems if not all. By using the proposed architecture, a simulation is carried out with the scenarios defined and make the comparison with a traditional database system and discuss the merits and demerits of using Hyperledger fabric. This will provide a clear picture on whether the Hyperledger Fabric is suitable for maintaining patients medical records, or if there are any shortcomings in the technologies that prevent adapting Hyperledger Fabric completely in the aforementioned case.

ACKNOWLEDGMENTS

We would like to thank Prof. Dr. Matrin Kappas for providing this opportunity to work on one of the most exciting and interesting technologies in the the field of Computer Science. Also, like to thank him for the support and guidance throughout the project to make it successful. Special thanks to Mr. Johannes Bouche and Mr. Lukas Atkinson for their constant support and feedback which helped us in achieving our goal and providing valuable insight into the technology. Many thanks to all our colleagues who worked with us and participated in this project along with us for their valuable feedback and also presenting their findings on the technology to help us gain more insight into how the technology can be adopted in various fields.

CONTENTS

LIST OF FIGURES

LISTINGS

ACRONYMS

HLF	Hyperledger Fabric
PDM	Patient Data Management
PoW	Proof of Work
PoS	Proof of Stake
MSP	Membership Service Provider
EHR	Electronic Health Record
SDK	Software Development Kit
API	Application Programming Interface
JWT	JSON Web Tokens
BFT	Byzantine Fault Tolerance

Part I

REPORT

1 INTRODUCTION

Blockchain is gaining popularity for its decentralized design, change-resistant database and secure data processing and proving to be a much better fail-safe system. Staring with the idea of having cryptographically secured blocks for maintaining documents having immutable timestamps to implementing it as a base for cryptocurrencies in bitcoin, to having enterprise graded blockchains, it is growing rapidly. Traditionally it was widely used in the financial domain, following Bitcoin's glory progress some other Blockchain technologies came up to increase its use besides financial perspective and started to spread its root to various other fields such as healthcare, supply chain and other industry 4.0 concepts. Due to its secure data processing and decentralized network, it provides an efficient way to handle data. As was the case with Bitcoin from the very beginning of this technology, the principal goal of Bitcoin was to give access to anonymous, reliable, trusted, securable, and auditable financial movements. Moreover, blockchain creates trusted and secure data storage with the use of cryptography techniques with combinations of distributed consensus procedures. In the case of Bitcoin makes sure of preventing these third-party involvements in the transaction. Blockchain mostly uses the distributed ledger concept, the ultimate representation of an ordered and correlative chain of some financial transaction. These transactions take place over an untrusted network utilizing the proof of task to complete the agreement among parties. Even though Blockchain is getting beloved by all, it is expected to have a great impact throughout the recent upcoming years and will be used significantly in such areas, such as supply chain management, IoT- Internet of things, healthcare and more on the way. Blockchain can be used in healthcare for the proper flow of secure data transfer to ensure confidentiality of sensitive data, to track patients between medicals as well to allow transparency on the treatment provided to the patients.

Managing patient's medical records is always a tedious task as it is extremely sensitive data that needs careful handling. Electronic Health Record (EHR) makes easy traceability of a patient's medical history providing more information for doctors as it helps them to explore patients' health databanks to make a suitable decision in providing the best treatment, but it comes with its own problems. When it comes to EHR, additional attention must be taken to provide security, easy accessibility and auditable. For example, whenever we think about information, especially our personal data stored online or transferred online at first it's a great concern about data manipulation, loss, or stolen other what leads to data unavailability on the required period. Making reliable and secure transfer of data among organizations over an unsecured network. Having a single EHR for each person alone will generate a large amount of data that must be handled in centralized systems. "To put this into perspective, in the financial domain, the total financial transactions made by Bitcoin have reached 400 million transactions in ten years. In this context, in Brazil's health domain, there were 1.4 billion patient visits just in 2018 carried out by its Unified Health System. In China, there were approximately 7 billion patient visits in 2017."[4] Which in-turn creates various problems in terms of maintainability, accessibility, security, reliability and interoperability. To move away from such problematic situations, blockchain can be adapted to handle the patient's data.

Adopting blockchain technology in the healthcare domain can transform the existing system by providing more reliability and easy accessibility through its distributed network, security by using cryptographic methods and auditability through immutable records. To achieve all these requirements a blockchain must have the aforementioned features along with an enterprise graded network to restrict public access without authorization. Hyperledger Fabric fulfills this by providing exactly the same kind of blockchain. Hyperledger Fabric offers an enterprise-grade blockchain network that utilizes the concept of smart contracts to carry out the transactions in the network. This could prove as an excellent solution in tackling the problems posed by traditional database systems in the healthcare domain along with the problems involved in the healthcare domain itself.

In this paper, the main discussion is all about patient data management which will be a flexible secure data transfer between health professionals who belong to different institutions and providing easy accessibility to the patient all through the use of Hyperledger Fabric.

1.1 MOTIVATION

Healthcare is one of the most important industries out there. In this era of technology and innovation, the Healthcare industry sometimes gives an illusion of an outlier. If we think about it, a lot of work in Hospitals, like storing and managing data of patients, is done the same way it was done a decade ago. On the other side, we can see statistics such as life expectancies and mortality rates suggesting that there have been major innovations in the healthcare industry. To decode these contradictory statements, we need to understand the difference between Vertical and Horizontal innovation. Vertical innovation talks about the innovation specific to a field or industry whereas horizontal innovations are the ones that cut across various verticals or industries [2]. Although there has been plenty of vertical innovations in healthcare, the horizontal innovations have not yet interfaced well with this industry. Blockchain is a great example of horizontal innovation, one that can be very beneficial for the healthcare industry with regards to storing and sharing patient's data electronically.

To understand the use of blockchain in patient data management, we need to dive deep into some of the requirements in this domain[1]:

1. Interoperability is probably the biggest requirement because access to a patient's medical history is essential to correctly prescribe medication. Ideally, the patient data should be usable across different healthcare institutions by various stakeholders like the patient, doctor, etc. But in reality, there exist issues like information blocking where an institution unethically blocks the data and denies sharing it with another healthcare institution.

2. Privacy and security are other important aspects that need to be taken into account. Health records may contain personal information, whose confidentiality should not be compromised.

3. Health records should be auditable and resistant to tampering. There is a need for a transparent yet secure system where any changes in the state of the database can be logged and traced back to the entity which made the change.

4. A patient-centric system is needed where the patient is in control of his/her data.

With the requirements mentioned above, it becomes quite hard to use a centralized system approach. The centralized approach makes it difficult to fulfill the interoperability requirement as the data is stored in the central database of one institution and information blocking is still a possibility. Also, there needs to be a high level of trust that needs to go in such a system as there is no transparency and the admin for the centralized database can manipulate the data (tampering) without being tracked. Another disadvantage is that there is a single point of failure.

A permissioned blockchain seems like a good fit for this situation. A blockchain is a distributed ledger that is immutable. A permissioned (private) blockchain is not open to all but only trusted participants are chosen to be part of such a system. It is favourable for interoperability as it is a distributed system. It is a secure and trusted medium that satisfies privacy and security requirements. It provides an auditable and tamper-proof system as blockchains are immutable and the history of transactions is recorded.

To sum it up, the primary motivation for this project is to use a permissioned blockchain to design a system for managing patient data (in the form of electronic health records) that is patient-centric and empowers the patient to control his/her data better. This system takes into account factors like interoperability, privacy, security, transparency and auditability of electronic health records.

1.2 INTRODUCTION TO HYPERLEDGER FABRIC

Blockchain has been around for the last two decades but as this technology involves many parties or entities and these entities can be trusted and/or untrusted, Blockchain has not been considered for enterprise applications. But with the concept of permissioned blockchain many technologies coming to the surface for trusted blockchain networks. Few examples of permissioned blockchains are NEM, Quorum, Chain core [6], and Hyperledger Fabric.

The Linux Foundation started the Hyperledger Fabric project which is a permissioned and private blockchain. It provides a permissioned immutable distributed ledger network for enterprises to carry out public as well as confidential transactions within the same network. It provides a trusted subnet within a shared network which allows only trusted parties to participate in this trusted subnet. It provides a network within a network that allows participants to communicate both non-confidential and confidential transactions with other network entities as per the requirement.

As blockchain has immutable ledgers and each ledger is appended after approval of the transaction. Different algorithms have been used in public blockchains, Proof of Work(PoW) and Proof of Stake (PoS) are examples of such algorithms used in transaction approval. Such algorithms are based on consensus and require several participants to participate in consensus to approve any transaction. This approach is not in compliance when it comes to enterprise transactions, as not all participants' approval is always required, only two participants or even a single participant's approval would be enough for certain transactions, and in consensus with PoW and PoS is not possible. To overcome this shortcoming, endorsement policies have been introduced in Hyperledger Fabric for transaction approval and each participant has been assigned their roles. Hyperledger Fabric network involved multiple role and components and it would be easier to explain further after defining these roles and components,

- Membership service provider (MSP): all network entities in Hyperledger Fabric enrolled themselves using a membership service provider. Every organization in Hyperledger Fabric can have its own MSP and then a general MSP would be required to perform a transaction across the network.

- Distributed ledger: Hyperledger Fabric provides an immutable ledger with each committing peer and gets appended after each successful transaction. Each ledger has further two components namely world state and transaction log. World state has the current state of ledger whereas the transaction log keeps the records of all transactions which resulted in the world state.

- Consensus: It happens in Hyperledger Fabric only through certain peers. Endorsement policy defines at least how many of those peers must approve the transaction before ledgers are appended.

- Smart contracts: These are program codes that provide Hyperledger Fabric participants controlled access to the ledger. Every task or function is performed in Hyperledger Fabric through pre-defined smart contracts. These smart contracts are being written in general programming such as Go, Javascript, Java, etc, and not in any domain-specific language.

- Chaincode: It is a bundle of smart contract and endorsement policy and has been defined on a channel at the time of the creation of the channel and all participants of the channel must approve chaincode to participate in any transaction through this channel. Chaincode contains all the tasks that can be performed across the network.
- Channel: A permissioned blockchain network in which only allowed participants can access. All transactions in Hyperledger Fabric are executed through a channel.
- Peer: Network entities contain ledger and involve directly in the transaction. Organizations involved in Hyperledger Fabric connect to the network using these peers. There are two types of peers: Endorsing peers responsible to endorse transactions as per the defined policies. While Committing peers commit transactions once it is endorsed by the minimum number of endorsing peers as per the endorsement policy.
- Orderer: A network entity responsible to order the transactions in which transactions should be appended in all ledgers after endorsement. There can be multiple orderers in a network and would form an ordering service.

After defining components and roles, Hyperledger Fabric can be explained as a blockchain in which organizations are allowed by administrators, access channels through peers. The role of these peers can be recognized using MSP. Organizations may have multiple peers and these peers can act as both committing as well as endorsing peers. An application or SDK provides a gateway for these organizations to access the channel. Through this application, allowed organizations can request a transaction. Endorsing peers endorse transactions after validation and submit them to Orderer. Orderer orders the sequence of transactions and then sends the sequence to committing peers. Committing peers after verifying that these transactions have been endorsed by endorsing peers commit the transactions. In Hyperledger Fabric, all committing peers contain a ledger and update all ledgers being simultaneous.

1.3 HYPERLEDGER FABRIC AS HEALTHCARE SYSTEM

Hyperledger Fabric provides a viable solution to the problems posed by handling Electronic Health Records (EHR). Being a permissioned blockchain, Hyperledger Fabric allows only identifiable participants (health institutions) to be involved in transactions, hence a system of ‘potentially’ trustworthy participants can be created. As explained in the previous section, the distributed ledger of Hyperledger Fabric is used to decentralize the EHR, which helps in tackling the issue of centralization of records, that is to avoid the single point of failure and maintain high availability and also reduces the overhead of transferring of EHR to another health institute in case of need by eliminating the middleman. Due to this, the data can be made more transparent to the patient. The immutable ledger offers the advantage of maintaining the integrity of the EHR and also the medical history of a patient can be tracked easily, thereby achieving auditability.

EHR demands a high-level of security, which is limited access to the records by authorized personnel and also a high-level of encryption. Hyperledger Fabric provides various pluggable features that aid in this. One such notable feature is the smart contact through which the ledger can be accessed, using which the core functionalities of this medical system can be built with the smart contract. For example, defining how the EHR is being created, who has access to it, who can modify the contents of the EHR and various other functionalities. There will also be a case where the patient might want to revoke access to a specific doctor or health institution and grant access to another which can be achieved with the help of smart contracts. Due to these smart contracts where the core functionalities of the system are defined, these functionalities can be separated into multiple smart contacts thereby providing maintainability of the system. In case of any change in the behaviour of the system for example a new functionality is to be provided for the doctors, which is defined in the smart contract can be easily replaced with another smart contract without having to affect the entire healthcare system.

Another major advantage of using Hyperledger Fabric is the concept of private data collection, this allows the participating organization to keep their data private from other organizations all while maintaining the decentralized nature of the Hyperledger Fabric. This is achieved by storing the data in a private collection where the collection can be owned by one or more organizations and the hashed value of that data is being distributed across the network. If an organization outside of the collection wants to check the integrity of the data it can request the hash value of the collection and check that value against the value it has at any point in time. Also, the actual data is never transmitted in the network and it never leaves the collection. In the healthcare system, this private data collection serves its purpose by providing much-needed security and encryption to the EHRs.

Then there are additional components that facilitate smooth functioning of the healthcare system such as Certificate Authority, using which the patients, doctors, and the hospital itself can be digitally authorized. Similarly, there are various modular components, that solve the challenges in healthcare systems. The healthcare system using Hyperledger Fabric can be broken down into several components. First the network itself with several hospitals as organizations with their peer nodes to host the smart contracts that define how the healthcare system can operate and the distributed ledger that stores the EHRs. Then the channel in the network where the communication between the hospitals takes place. Followed by the functionalities of the healthcare system, how it is operated is defined in the smart contract. The communication between the user that is the patient/doctor takes place through the APIs provided in the Software Development Kit (SDK) which then invokes the functionalities in the smart contract to carry out the transactions.

2 DESIGN

2.1 ARCHITECTURE

The architecture of the patient data management system consists of the following components: One channel for communication, a consortium containing two organizations (hospitals), a Membership Service Provider (MSP) and a Certificate Authority for each hospital, an individual peer for each hospital, a ledger with a copy of world state on each hospital peer, chaincode deployed on each hospital peer, an ordering service with one orderer node, an endorsement policy, actors, and a web application. Figure 2.1 can be referred, to understand the proposed architecture better.

To keep the architecture simple and scalable, there is only one channel (channel C) in the system. All the hospital organizations access this channel to carry out any transactions. Initially, two hospitals (organizations) are part of this architecture but there is flexibility and as many hospitals can be added as needed. Each hospital has its own Membership Service Provider (MSP) and Certificate Authority (CA) which are responsible for generating the public-private key pairs for the organizations and signing the certificates. The organizations cannot directly interact with the channel in a Hyperledger Fabric network, so they need peers to access it. Here we have one peer each for every hospital. Each peer has a copy of the state database consisting of the current world state of the network. When any create, read, update or delete transaction is carried out, the world state is updated for all the peer ledgers. Smart contracts are written for every type of interaction that is needed with the Hyperledger Fabric network. These smart contracts are packed into a chaincode and this chaincode is deployed on every peer node for hospitals.

Any transaction that is validated by a peer goes to the ordering service where further actions are decided according to the active endorsing policy. In our architecture, there is one orderer node, but in realistic implementation, more than one orderer could be used which would make the system more reliable and fault-tolerant. With regards to endorsement policy, which specifies organizations that need to execute chaincode to validate transactions, we have persisted with the default endorsement policy wherein both the hospital organizations need to approve a transaction. There are three actors in our system namely admin, doctor and patient. Each organization has an admin who is responsible to enrol doctors and patients. Doctors are medical practitioners and patients are the users for whom the medical records will be created in the distributed ledger. A web application consisting of a frontend and backend acts as the interface between actors and the Hyper-

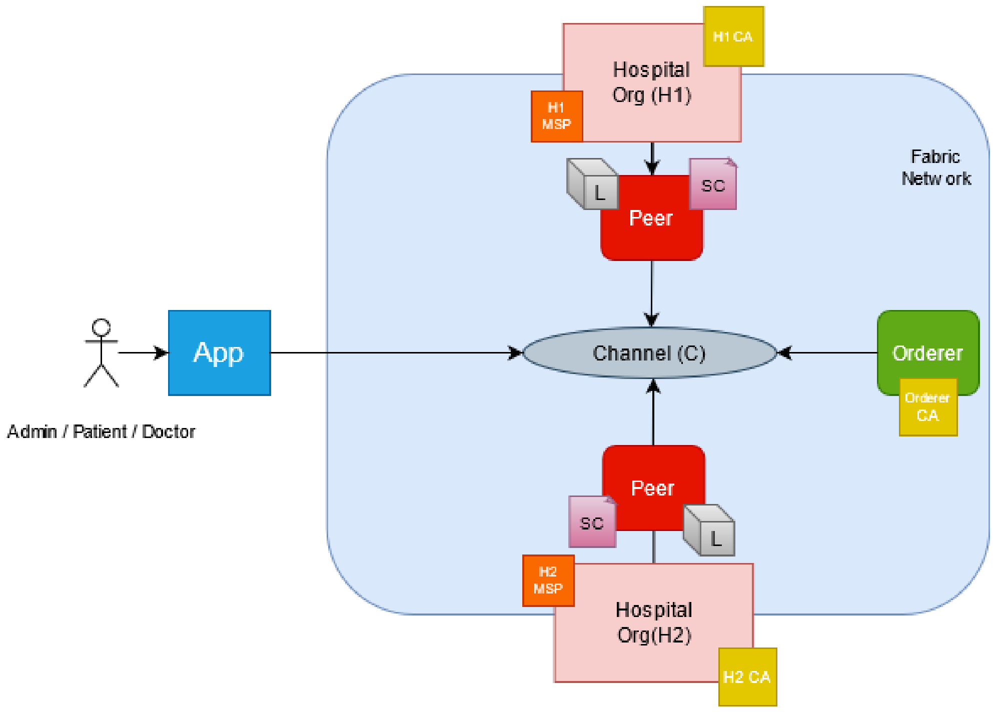


Figure 2.1: Proposed Architecture

ledger Fabric network. Only authorized users can log in to the system and perform their assigned tasks.

The architecture described till now is the one that is implemented for the prototype. It has many advantages like it is simple yet secure and easily scalable, but it is a refined version. In the early phase of this project, many ideas were brainstormed and one architecture is particularly worth mentioning. The idea for that architecture was that every hospital organization that joined the network would have its own channel. There would be two peers (a doctor peer and a patient peer) for every organization. Each peer would have a copy of the world state and chaincode specific to that organization. The patients and doctors would interact with the blockchain network through different web interfaces. The benefits of this design were security and privacy. Since the patient's data was only present in the particular channel whose hospital he/she visits, other doctors and patients would not be able to access this data. Significant drawbacks were the complexity and poor scalability. The number of channels in such a design would grow linearly with respect to the number of hospitals added in the blockchain network.

It was later realised that the privacy and security features could be achieved, without increasing the complexity by introducing a new channel for each organization, using a feature of Hyperledger Fabric called Private data collection. Hence this architecture was not pursued further. Private data collection and its use for the patient data management use case will be discussed in some more detail in further sections.

2.2 WORKFLOW

Each hospital has an admin who is responsible for registering the doctors in that hospital and also any new patient that visits that hospital. The Patient Data Management system registers a patient and gives all authority of his/her health record to the patient; Every hospital would have the patient's health record in their ledger. The patient has the authority to allow any doctor to update his/her health record or restrict any doctor. This enables patients to move easily from one doctor to another doctor or one hospital to another hospital within the Hyperledger Fabric network.

At the time of patient registration, a doctor is also assigned to the patient. The patient's health record contains two parts; personal data and medical data. Personal data contain fields like name, id, address. The patient has rights to read and update personal data while he/she can only read medical data and cannot do any changes in it. Other than that, the patient also has the right to give access to his/her health record to any other doctor and revoke the access from any doctor whenever he/she wants.

Once a doctor is assigned to any patient, he/she gets access to the patient's health record. Doctors can read patient health records and update only medical data of the patient's health record when needed. Once the patient revokes access for a doctor, the doctor can no longer access the patient's health record.

Many scenarios are possible for this system when considering the interaction between the involved actors. In the following subsections, we discuss some of the common scenarios.

2.2.1 *Scenario 1: Patient visits doctor1 in Hospital1 for the first time*

In this scenario, the patient is visiting a hospital in the blockchain network for the first time. So, till this point, there will be no EHR created for this particular patient. The admin registers the patient in the system which creates a new EHR for the patient with personal details as well medical data. The doctor whom the patient visits (doctor1) is set as the doctor for this record and this doctor by default has the access to read or update this EHR. Of course, the prerequisite is that the doctor should already be registered by the admin before he/she can be assigned to a patient.

2.2.2 *Scenario 2: Patient visits doctor2 in Hospital2*

Since the EHR for patient1 is already present, there is no action on the admin related to the patient. The doctor though needs to be registered to the system if he/she is not already registered. The existing EHR can then be updated in such a way by the system that this new doctor (doctor2) is included in the list of doctors authorized to access EHR for this patient (patient1). The patient can do this by granting access to the new doctor. Doctor2 can now read or update the EHR and the rights for Doctor1 are not affected by this.

2.2.3 *Scenario 3: Patient has completed his/her treatment with Doctor1*

In this case, the patient can decide that since the treatment is complete, doctor1 should not be able to access the data for the patient anymore. The patient can revoke the access of doctor1 to his EHR. This will remove doctor1 from the list of authorized doctors and he will no longer be able to read or update the record for patient1. As expected, there will be no change in the access for doctor2. The patient and doctor2 will of course be able to read the complete data in EHR, including the data generated by doctor1.

2.3 SECURITY

Security of EHR is of utmost importance in healthcare systems. To provide security to EHR, Private Data Collection is used. Private Data Collection provides a way to keep the data handled by an organization in the network private while maintaining the distributed ledger for that data. This way the organizations participating in the network can see the transactions being carried out but not the actual data instead a hash value of that data will be available. This is due to the definition of collections, to which the data is associated. This data resides in a private database with the organization that is authorized to access it. If in case any other user/organization that does not have access to that collection tries to access it, the smart contract will verify the mspID and other details and will deny access. This can be visualized from Figure 2.2. Access to this collection can be defined in the collection configuration. Only the hash value of the data is endorsed by the peers, then ordered and written to the ledgers of other peers in the network. Even while sending the proposal response back to the client, only the hash values are sent back along with the read/write sets and public data. In the healthcare system, this can be incorporated by defining a set of collections for each hospital involved in the network. This has to be defined before deploying the chaincode to the network as the configuration will be part of the chaincode. There can be multiple cases considering the actual possible scenario involved with a patient and hospital.

Case 1: Firstly a collection for each of the hospitals in the network has to be defined considering a user/patient will visit at least one hospital. Thereby ensuring the EHRs of each hospital are private from the rest.

Case 2: To facilitate the use case of a patient visiting another hospital, that is a patient is registered with two hospitals at the same time or the patient registers himself/herself with a second hospital. Here the EHR of that patient has to be shared between both the hospitals that the patient is registered with. Hence a common collection has to be defined with access provided to those two hospitals involved and the EHR from the first collection where

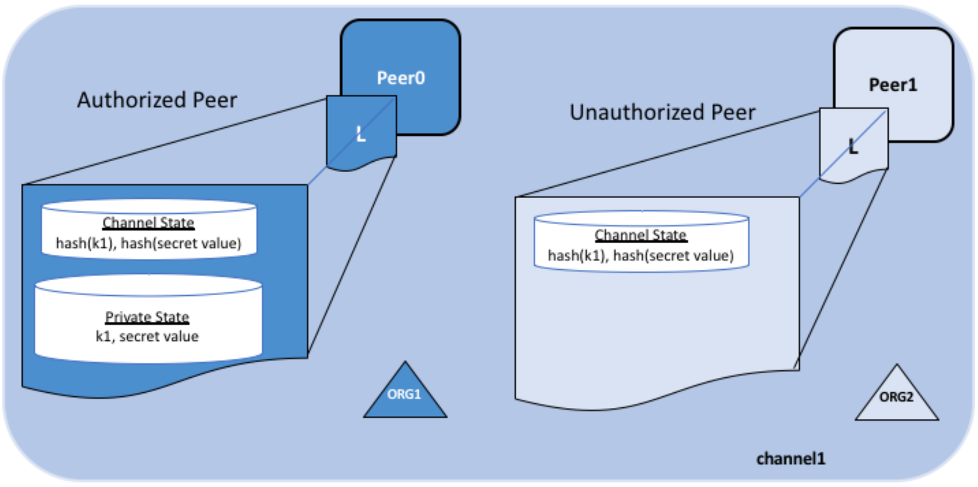


Figure 2.2: Private Data Collection

the access was with the first hospital has to be moved to this new common collection. This can be repeated if the patient decides to register with the third hospital and so on.

Case 3: If a patient decides to deregister from one hospital and move to another hospital, then the EHR of that patient has to be added to the collection associated with the new hospital and the old hospital will just have the hash values of that data.

By considering all the scenarios, there has to be 'n! + 1' combinations of collections where each hospital has a collection where the access is shared with each of other hospitals and the combination of others. For example, consider Hospital A, Hospital B and Hospital C. The collections for these hospitals would have Hospital A, Hospital B, Hospital C, Hospital AB, Hospital AC, Hospital BC and Hospital ABC. This way no matter which hospital/combination of hospitals the patient visits his/her EHR will be secure and the data will be not available outside of that collection.

Also, in addition to this security, Private Data Collection allows the definition of endorsement policy in its configuration. Defining the endorsement policy with the Private Data Collection caters to the requirement of the system, that the transaction of EHR related to any hospital has to be done by that hospital alone. In case if the private data is shared between multiple hospitals then the hospital that is carrying out the transaction in that collection has to sign the transaction.

3 IMPLEMENTATION

3.1 TECH STACK

The implementation of the complete system can be broken down into various components namely the Hyperledger Fabric network and distributed ledger, smart contracts, application SDK and application frontend. The network is the actual blockchain part of the application. The smart contracts are implemented in JavaScript. The SDK developed using Node.js and the frontend of the application is built using Angular. In this section, we have a look at these components in some more detail.

3.2 HYPERLEDGER FABRIC NETWORK

Hyperledger Fabric network is the base of the system. This consists of the hospitals with their peers participating in the channel with the distributed ledger. As explained in the architecture all the hospitals are connected to one single channel. For ease of use, the test-network available from the Hyperledger Fabric[11] was made use of in setting up the Patient Data Management System. Using the existing network the organizations are changed to represent the hospitals by modifying the docker files, configuration files and the corresponding certificates for it. The changes mainly included modifying the organization names to include the hospital in the configtx.yaml file and the associated Certificate Authorities in the docker-compose file. Then the files that reference these files are updated. Once the modified network is brought up with two hospitals and a channel as displayed in the architecture. All the necessary certificates for the organization and peers involved would be created.

3.3 DATABASE (DISTRIBUTED LEDGER)

As per the official documents, hyperledger fabric supports 2 peer databases; LevelDB and CouchDB. LevelDB is the default database for hyperledger fabric which stores data as key-value pairs, whereas couchDB is an alternative which records data as JSON documents. CouchDB allows rich queries over JSON documents compared to LevelDB which allows only composite key queries.

We have used CouchDB as a peer database in this project and modeled each patient health record as a JSON document as shown in the Listing 3.1. Each electronic health record contains multiple fields and a patientId field is used to identify the owner of the health record. Other fields are for patient's personal detail as well as medical data. Another field is an array of list of doctors allowed to access the record only doctors mentioned in the list can access this record otherwise Hyperledger Fabric would deny the access.

```
{
   "Record":{
      "PatientId":"patient1",
      "Address":"Address XX, 123 Street, City",
      "Telephone":17615945896,
      "Diagnosis":"Common cold",
      "Medication":"paracetamol",
      "DoctorAuthorizationList":["doctor1"]
   }
}
```

Listing 3.1: Data Structure in CouchDB

The reason to use couchDB is to be able to execute grouped queries using couchDB indexing which allows to group JSON document (in our case health record) as per any field present in JSON document.For now not any indexing or design document (ddoc) feature of couchDB has been used but for future implication can be considered.

Also Docker image of couchDB runs on the same server as the peer and the number of images depends on the number of peers. Each peer has one ledger so one couchdb image for each peer would be required for a hyperledger fabric network.

3.4 SMART CONTRACTS AND CHAINCODE

All the executable business logic for the application is implemented using smart contracts. This means that any Create, Read, Update or Delete operations to the distributed ledger are carried out via smart contracts. Smart contracts can be different functions or even different files (or classes) depending on the architecture and programming language used. We have used Javascript to implement our smart contracts. To keep the architecture simple and modular, we have written one function for every functionality that our system needed to interact with the HLF network.

Smart contracts are generally developed around the entity upon which the transactions are supposed to take place in the network [8]. In our case, this is the EHR and we have developed smart contracts around that. CreateRecord() helps create a new EHR in the distributed ledger when an admin registers a new patient. UpdatePatientInfo() and UpdateRecord() are contracts used to update the personal information and medical information for the patient respectively. DoctorReadRecord() and PatientReadRecord() contracts are triggered when a doctor or patient tries to read an EHR. GetRecordHistory() contract is used to fetch the history of a particular EHR. GetHistory is a feature of the HLF blockchain network which helps to fetch the history of transactions that have occurred on a particular entity. This can be helpful since the world state only stores the updated or latest state of a record and with history, any previous transactions can be tracked.

Grant and revoke access is one of the key features of the Patient Data Management System and implemented using GrantAccess() and RevokeAccess() functions. This allows the patient to grant or revoke access to a doctor to his/her EHR. It is implemented in the smart contract as the access to the data can be controlled at the time of data retrieval so that the data is within the fabric network before the check is completed. This is achieved by having an access control list in the EHR called DoctorAuthorizationList, which contains the list of IDs of doctors who are authorized to access that EHR. Patients can specify a doctor to whom the access has to be granted from the UI and that doctor's ID will be added to the patient's EHR. Similarly to revoke the access of a doctor, a patient can choose the doctor and that doctor's ID will be removed from the DoctorAuthorizationList. Only the patient will have access to this list and the method to grant or revoke access and the doctor does not have access to either. And also this list will not be visible to the doctor.

When a doctor tries to access or modify the EHR of a patient, the system checks the user ID of the doctor against the DoctorAuthorizationList in that EHR and if that ID is unavailable in the list, then that doctor is not authorized to view or modify the EHR and the system will return 'Access Denied' to that doctor. Similarly, if the iD is available then the Doctor has the authorization to view or modify the EHR. Figure 3.1 can be referred, to understand the interaction of all the smart contracts with the rest of the application.

Smart contracts are generally packaged into Chaincode and deployed on the blockchain network. So, a chaincode may have multiple smart contracts and once this chaincode is deployed on the network, all the contracts become available to the application. The deployment of chaincode to the HLF network can be broadly divided into 4 steps [3]: 1) Packaging the chaincode, 2) Installing chaincode on peers, 3) Approving chaincode definition for an organization, 4) Commitig chaincode definition to channel. All these steps can be done at once when executing the command deployCC once the HLF network is up. We need to pass the correct flags as the path of our chaincode and the language in which it is written.

3.5 software development kit (sdk)

Connection to the Hyperledger Fabric network, to invoke smart contracts and carry out transactions is carried out with the help of APIs provided by the Hyperledger Fabric client Software Development Kit (SDK). "Hyperledger Fabric SDK provides various configurable components like cryptographic algorithms for signatures, logging through an extensible standard interface.[9]. There are various SDKs available but for the development of this prototype, Node.js is considered. In this Patient Data Management System, SDK takes care of the registration of users to the fabric network, updating the patient's record, granting/revoking access to a doctor, any actions that the user triggers from the UI will be carried out in the SDK and if the action is to invoke the smart contract then SDK will establish a connection with the Fabric network and invoke the smart contract. The connection between the SDK and the Fabric network is created with the help of the Gateway class provided by the Hyperledger Fabric. By taking the user details from the wallet and the channel name that needs to be connected to access the right peer and its ledger, a network connection is established and an instance of this network is used to get the appropriate chaincode and invoke the smart contracts. This network connection is valid to that user, whose identity was used to create the connection and there can be multiple network connections established. Whenever an API is triggered in the SDK, it will establish a network connection with the channel and then fetch the chaincode. The connection will be terminated after the transaction is carried by invoking the termination method available in the Gateway class.

3.5.1 *Wallet*

As mentioned earlier the wallet is used as an identifier and it is an important part of the Hyperledger Fabric SDK. It stores the Fabric metadata, public key contained in an X.509 certificate and private key in an identity file certified by a Certificate Authority. There can be different types of wallet, file, in-memory and database wallets. In the development of the Patient Data Management System, the file type is used. During the connection establishment, the Gateway class uses the mspID and the type available in the wallet for that user for connecting to the network and determining if the user has the access rights to the channel mentioned[12].

3.5.2 *Application Programming Interface (API)*

The basic functionalities of handling an EHR are implemented as shown in the Figure 3.1. When the Register Doctor/Patient action is carried out, the registerUser() method is invoked in the SDK and then it takes the user details like the ID, Username and hospital the user will be affiliated to. Then by getting the network configurations for that hospital, a wallet is created and an identity file is included for that user in the wallet. This serves as an identifier for the user to have access to the Fabric network. This process has to be carried out by the admin of the hospital and the registration of admin is done similarly but it is a one-time-setup carried out before the user registration. Incase of Patient's registration this will invoke the CreateRecord() method from the smart contract which will then create a new record for that Patient with the personal information. Once the user has been registered he/she will have access to the APIs provided by the SDK. Certain APIs are exposed only to certain users so that the data cannot be manipulated by each user. That is Register Doctor/Patient action is limited only to the hospital admin.

Whereas Update Patient Health Record is only accessible to Doctors as this API modifies the diagnosis and medication of EHR. Access to Update Patient Personal Information, Grant/Revoke Access to Doctor is only provided to patients as the doctors are not allowed to add personal details nor modify the DoctorAuthorizationList. Reading a patient's data and Read Patient's Medical History is provided for both the Doctors and Patients as this is used to access and read the EHR. Each user is assigned a role (admin, doctor and patient) using which certain actions are restricted or based on the role certain smart contracts are invoked. For example, reading a patient's data can be performed both by a Doctor or a Patient. Irrespective of the user the action triggers the same API, i.e readPatientData(). Then in the SDK based on the role, different smart contract methods are invoked.

3.5.3 *JSON Web Token (JWT)*

To handle the authorization to APIs and to maintain the session of the user, JSON Web Tokens (JWT) have been incorporated. This provides a way to ensure that the user who logged in is the user who is accessing the API. When the user logs in to the application, a token is generated using the username and the user's password and signed off and encrypted and sent back and stored on the client-side so the server does not have to store the session details itself. Next time when the user accesses the API with the same token, the JWT verifies the token with the key (user's password) and if it's not been tampered with or if it's the same user then it authorizes the user to access the API. For the user login, the credentials can be encrypted and stored in a separate database ideally with the application provider, but to keep the focus on Hyperledger Fabric rather than application development, the credentials are stored in a file as JSON which is validated against. The credential validation takes place along with token validation as shown in the Figure 3.1. for the Login action in the application.

3.6 APPLICATION FRONTEND

An application consists of more than one layer. One is the presentation layer-what a user can view and using this can interect with the system. Another one is the presentation logic layer that works behind the presentation layer and it is invisible to the users. This layer mainly controls all the activities and data transformation between front and backend. AngularJS builds on MVC architecture and it carries API requests towards the server to expose data from the server. Therefore, view associates with controller to collect data from server information and shows to the user as it is requested through the system function.[5]

Front end of the application is designed and developed using the Angular framework and this application is compatible with all modern web browsers (not tested with old web browsers). We tried to create separate views and

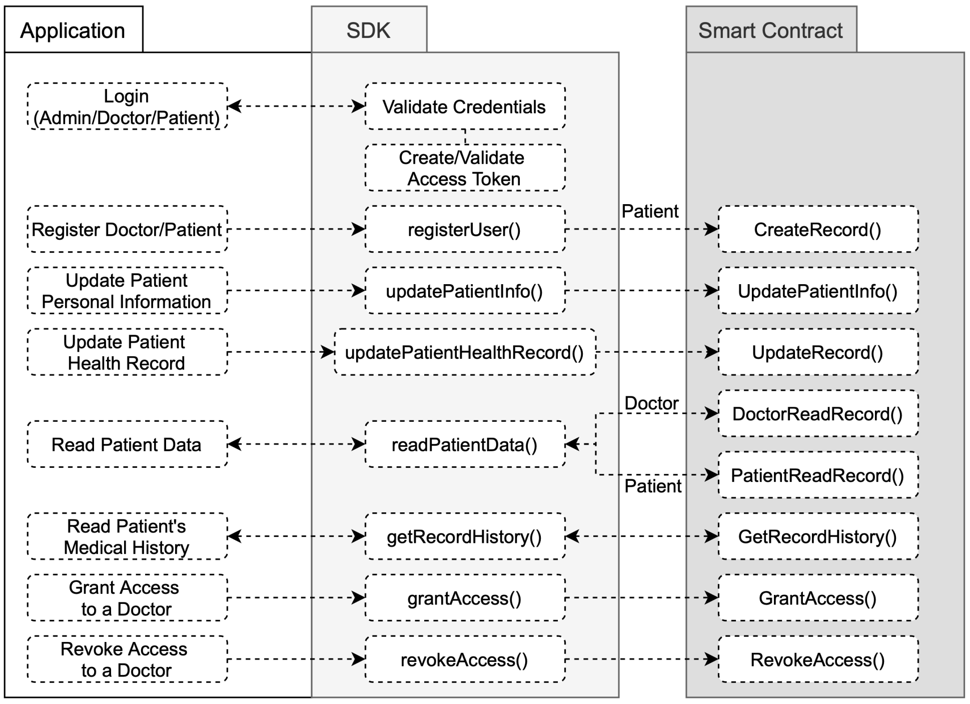


Figure 3.1: API connection between SDK and Smart Contract

restrict the access and tasks as per the roles (admin, doctor, patients). e.g only admin has the option to register any user in PDM systems and any other user cannot get this option.

As the backend server sends user information along with jwt (java web token), the front end application stores this information along with jwt for the entire session in local browser history. Every time a logged in user tries to access the Hyperledger fabric network using the PDM front end, the back-end application requires jwt of the user before allowing the user to access HLF network. Stored information contains what is the username and role assigned to the user and the user gets the view and controls in the front end application on that basis.

There are four views in application in total; Login, admin, doctor and patient. Login page contains two fields for username and password when no one is logged in. Once a user is logged in this page shows that someone is logged in and allows the user logged out by enabling the log out button, as shown in Figures 3.2 and 3.3

Admin view is enabled when the browser receives the admin role from the backend along after the user logged in. This admin view contains only the "Register" option in it as admin is only supposed to register patients or doctors. Further tasks can be considered in future implementation such as suspend some user as warning or delete the user once user is not the part of the PDM system. Though deletion can still be done by manually by deleting user details from HLF wallet. This is shown in figure 3.4

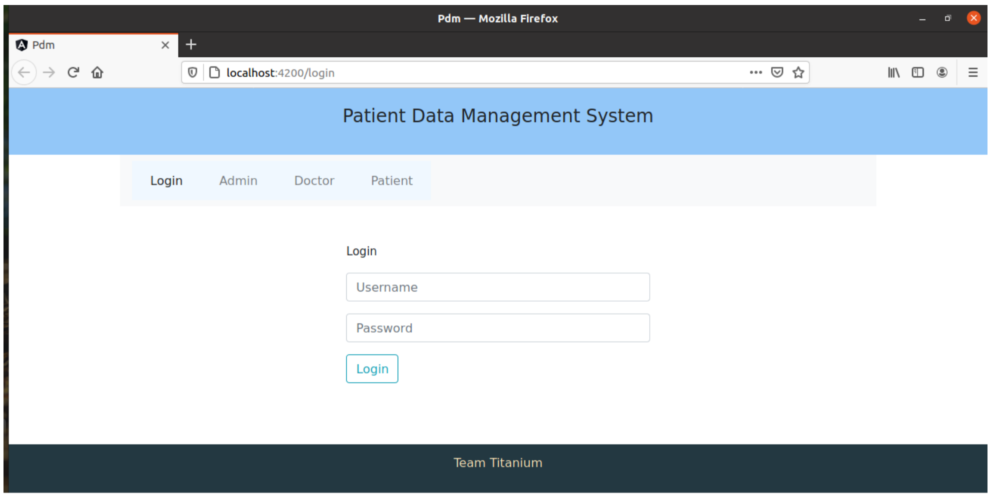


Figure 3.2: Login Screen

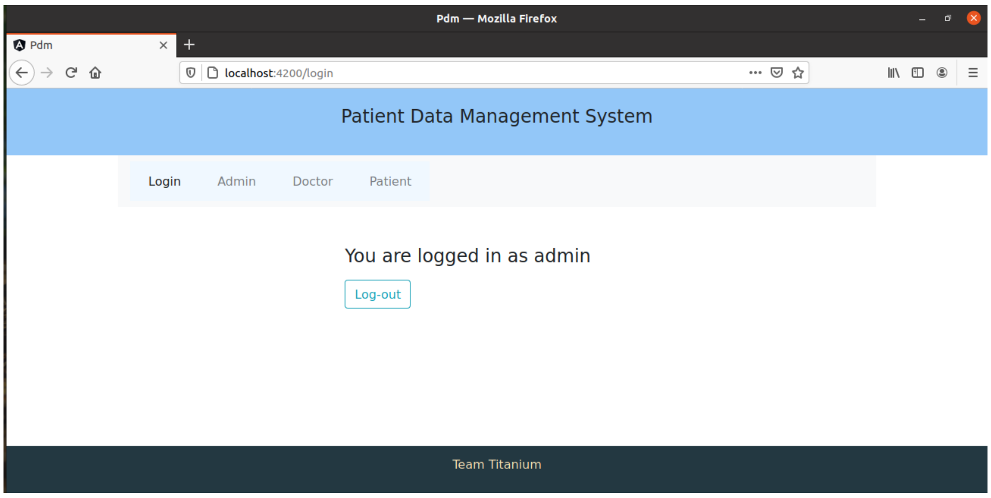


Figure 3.3: Logged in view

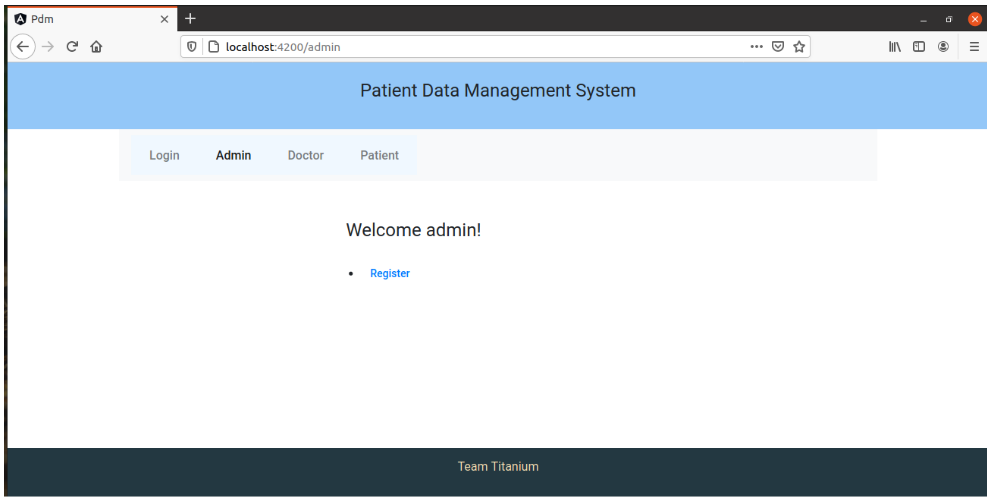


Figure 3.4: Admin view

To register a doctor, the admin needs to enter a few details like doctorid and the organization. To register a patient, some additional data needs to be provided like the medical details (diagnosis, medication) because at the time of patient registration a new electronic health record is created in the blockchain network.

Doctor view has two options in it. Read patient data and update patient data. These two functionalities allow doctor to perform basic functions now such as looking at patient data and updating the patient data where it is needed as shown in figure 3.5

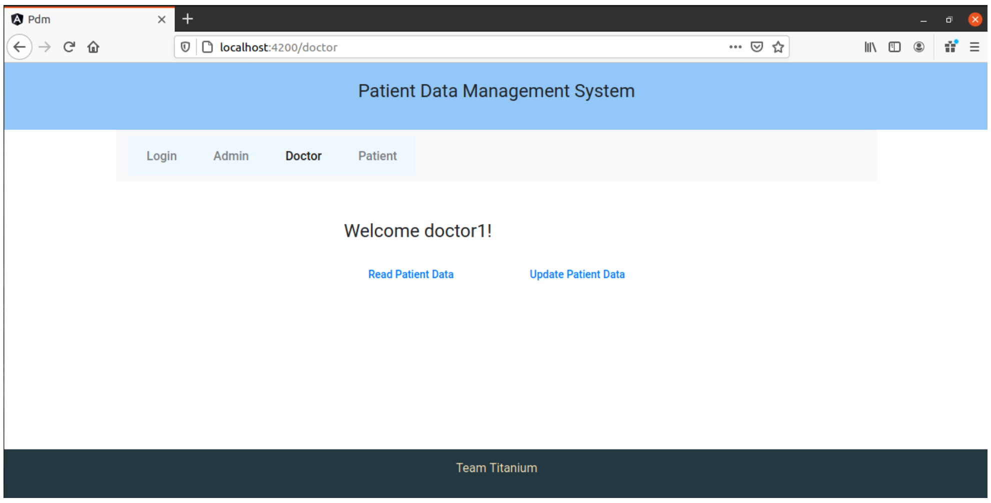
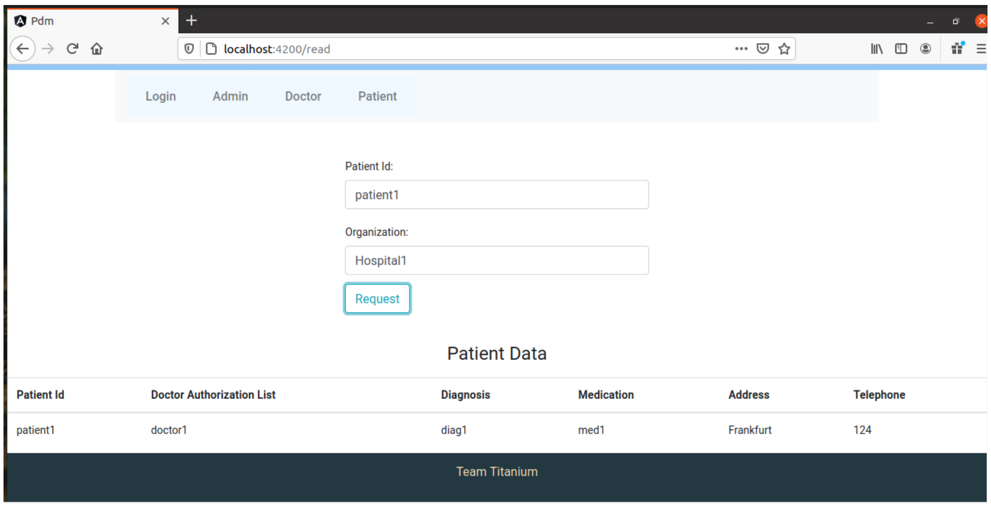

Pdm
localhost:4200/doctor
Patient Data Management System
Login Admin Doctor Patient
Welcome doctor1!
Read Patient Data
Update Patient Data
Team Titanium

Figure 3.5: Doctor view

Read patient data requires two entries from a doctor, Patient ID and Organization (Hospital in which patient is registered). If the patient exists and the patient already added the doctor in his/her doctor authorization list then the doctor will see patients details in a table at the bottom of the page as displayed in Figure 3.6


Pdm
localhost:4200/read
Login Admin Doctor Patient

Patient Id: patient1

Organization: Hospital1

Request

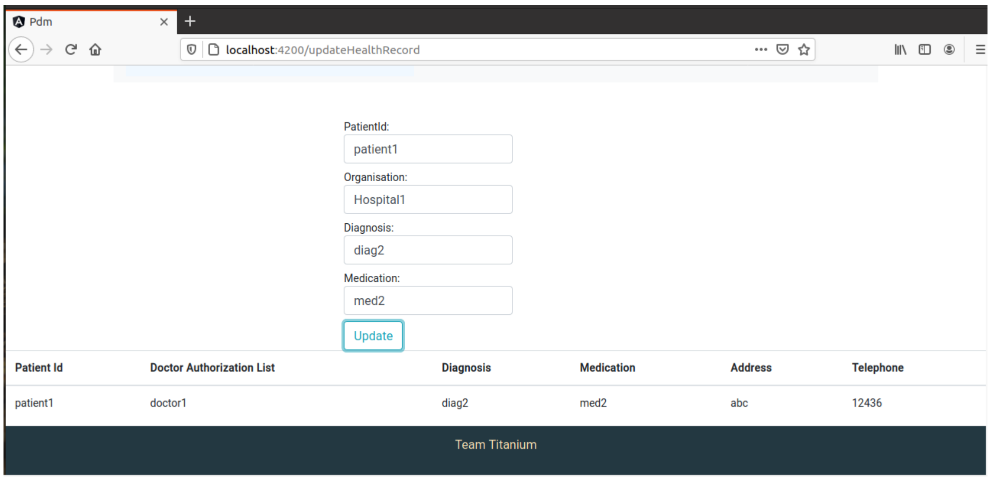
Patient Data

Patient Id	Doctor Authorization List	Diagnosis	Medication	Address	Telephone
patient1	doctor1	diag1	med1	Frankfurt	124

Team Titanium

Figure 3.6: Patient data read access for Doctor

Update patient data page has four entries: Patient ID, Organization, Diagnosis and Medications. As in the PDM system doctors can update only medical data of the patient. If the doctor is allowed by the patient, the doctor can enter new data in medication and diagnosis fields and after getting updated in peers databases, the table at the bottom of the page will show updated health record. These are basic information for any electronic health record might have but for real life implementation these table might have a lot more information as shown in figure3.7


PatientId: patient1
Organisation: Hospital1
Diagnosis: diag2
Medication: med2

Patient Id	Doctor Authorization List	Diagnosis	Medication	Address	Telephone
patient1	doctor1	diag2	med2	abc	12436

Figure 3.7: Patient data update by Doctor

Patient view has four options in total: Read patient data, Update personal info, Grant Access to doctor, Revoke access for a doctor.Which are basic functionalities require in any electronic health record This is displayed in figure 3.8

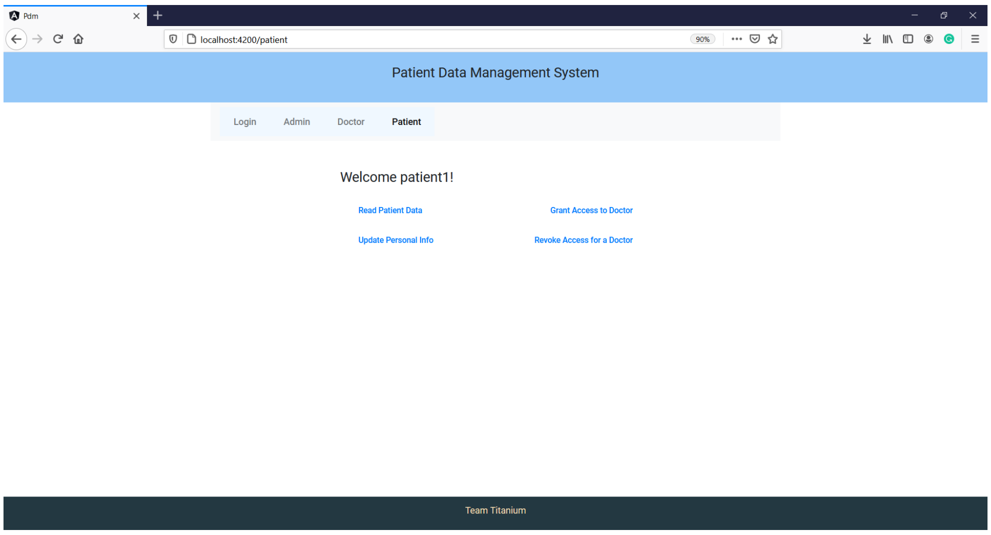
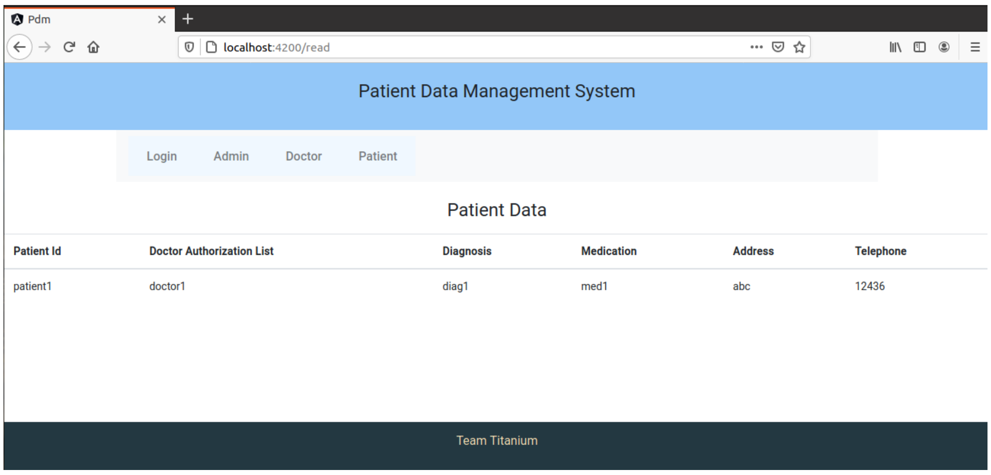


Figure 3.8: Patient view

Read patient data is the same as in doctor view but here the patient has no field to enter data to restrict the patient to update any unnecessary information. As medical information is critical and only medical professionals must write these information for any patient.For future development reference to multimedia (X-ray report, CT scan reports etc) can also be added in this health record. Front end application uses stored user account information and jwt from local storage and retrieves patient's data automatically and shows it in a table. Figure 3.9 shows health record view in PDM System

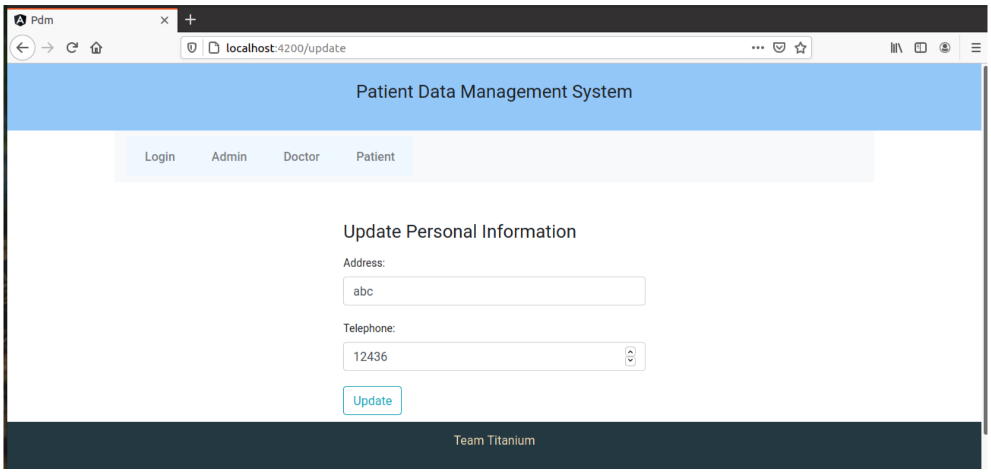


Patient Data

Patient Id	Doctor Authorization List	Diagnosis	Medication	Address	Telephone
patient1	doctor1	diag1	med1	abc	12436

Figure 3.9: Read access for patient of patient data

Update personal info shows the already existing information in two entry fields; address and telephone. Patient can edit these fields if he/she wants to change the current information. One point to be noted is that the patient has not been given any write access over his medical data as that can only be done by certified medical professional and patient can misuse that access if given as shown in figure 3.10



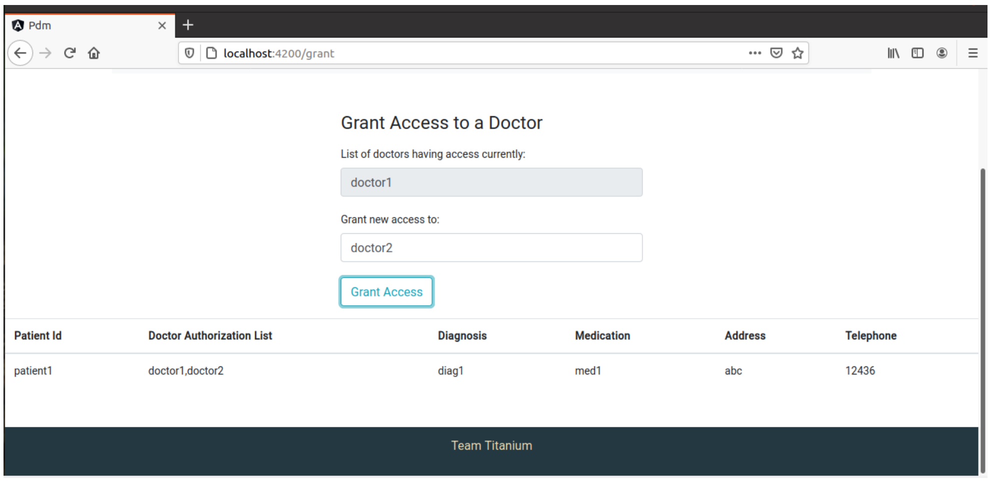
Update Personal Information

Address: abc

Telephone: 12436

Figure 3.10: Read and update patient personal data

Grant Access and Revoke access to doctors in this application give patient all the authority in PDM system over his/her medical record that which doctor can access his/her data and which doctor must not have any more access to the patient data. First field shows all doctors given access to this health record and second entry field allows the patient to give the new doctor access to his/ her medical record. After entering new doctor details and hitting Grant access button, the new doctor id is added in doctor authorization list in all peers databases and the doctor now access the patient health record and update it when it is required as shown in figure 3.11



Grant Access to a Doctor

List of doctors having access currently: doctor1

Grant new access to: doctor2

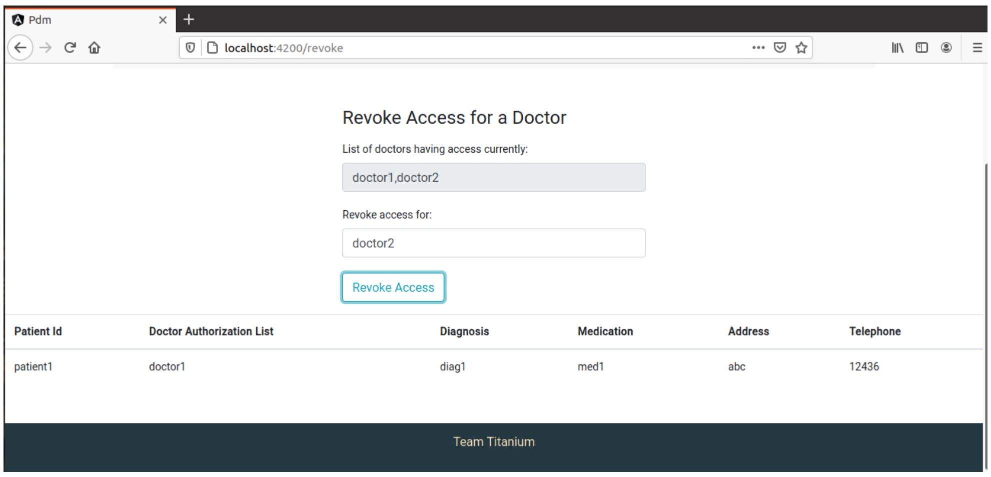
Grant Access

Patient Id	Doctor Authorization List	Diagnosis	Medication	Address	Telephone
patient1	doctor1,doctor2	diag1	med1	abc	12436

Team Titanium

Figure 3.11: Grant access to doctor by the patient

This option also has two fields, first containing a list of all doctors allowed to access this health record and second entry field can be used by patient to revoke access of any doctor who should not access this health record any more. After entering doctor id and clicking on remove access patient remove the doctor access and the doctor id is removed from peer databases for this health record. Figure 3.12 displays the revocation of access from patient health record.



Revoke Access for a Doctor

List of doctors having access currently: doctor1,doctor2

Revoke access for: doctor2

Revoke Access

Patient Id	Doctor Authorization List	Diagnosis	Medication	Address	Telephone
patient1	doctor1	diag1	med1	abc	12436

Team Titanium

Figure 3.12: Revoke doctor from patient health record

3.7 SECURITY

As mentioned in the design section2.3, the much-needed security can be provided by the Private Data Collection. This can be implemented by including a "collections_config.json" along with the chaincode. This file will contain the configurations needed for creating the collections as shown in Listing 3.2. Here the name of the collection, minimum peer that the data is disseminated to before signing, maximum peer to which the data is distributed to and endorsement policy also can be defined along with other parameters. This has to be pre-configured and deployed along with the chaincode. These parameters allow further restriction to the data to increase the security for the data.

```
"name": "Hospital1Collection",
"policy": "OR('Hospital1MSP.member')",
"requiredPeerCount": 0,
"maxPeerCount": 1,
"blockToLive":3,
"memberOnlyRead": true,
"memberOnlyWrite": false,
"endorsementPolicy": {
  "signaturePolicy": "OR('Hospital1MSP.member')"
}
```

Listing 3.2: Private Data Collection Configuration

Since Hyperledger Fabric is still a developing blockchain, unfortunately, the Private Data Collection is limited in its features. As it lacks the ability to provide the transaction history of the private data which is the EHR in the Patient Data Management System. In this system, a patient's medical history is one of the important parts of EHR and a core feature, as the Hyperledger Fabric provides easy access to the transaction history. This issue has been under development but not yet available to the user at the time of development of this system. The Hyperledger Fabric records the transaction in private data, but there are no APIs to extract that information. A workaround is suggested by the Hyperledger Fabric community, which is available in the issue description of their bug-report [7]. Unfortunately, the provided workaround did not yield any result. So the Patient Data Management System lacks the Private Data Collection feature to provide security.

3.7.1 *Alternate Solution to Private Data Collection*

Considering the drawback of the Private Data Collection, the proposed alternate solution could solve the problem of retrieving the transaction history of private data. The proposed solution involves modifying how the data will be stored in the ledger. As shown in the Listing 3.3, as opposed to the designed structure where diagnosis and medication are considered as a simple string array. Here anything related to a patient's health is stored as an array of EHR objects.

```
    {
        "PatientID": "Patient1",
        "Address": "Address XX, 12345, City A",
        "Telephone": "123456789",
        "DoctorAuthorizationList": ["Doctor1","Doctor2"],
        "EHR":[{
                "RecordNo": "EHR1",
                "Diagnosis": "Diagnosis1",
                "Medication": "Medication1",
                "Timestamp": "DD.MM.YYYY",
                "DoctorId": "Doctor1"
        },{
                "RecordNo": "EHR2",
                "Diagnosis": "Diagnosis1",
                "Medication": "Medication2",
                "Timestamp": "DD.MM.YYYY",
                "DoctorId": "Doctor2"
        }]
}
```

Listing 3.3: Alternate Solution for Data Structure

The advantage in restructuring allows the system to track the history at the state database level and not rely on ledger transactions to get the history. This is achieved by creating a new EHR object every time a patient visits the doctor, this object is added into the EHR array with a unique record number which is incremented along with the corresponding details provided by the doctor. Doctors can only insert new EHR objects into a patient's data and they cannot modify the previously inserted records. By using the private data for this structure this data is secured, other hospitals will have the hash value of this data. While retrieving the history the system can fetch this data as it would with a normal state database and the history can be easily extracted from the EHR array and can be presented to the user. This can be visualized as a folder dedicated to a patient with his name on the folder and the folder contains medical reports from every visit to the doctor.

One of the drawbacks of this proposed approach would be it would create a large amount of data as an EHR record in the state database. Which in turn may cause latency due to retrieval and processing of such large data. This will however not break the existing flow designed for the system but in turn, addresses one of the concerns from the existing structure. That is if the doctor's id is available in the DoctorAuthorizationList[ ] the doctor who wants to view the data can also modify the current state of the patient's EHR as it has just a string array and the history is tracked separately. But this is eliminated as the doctors are allowed to just add a new EHR to the existing record rather than modify it.

4 DISCUSSION

4.1 CHALLENGES

Designing the system itself proved to be a challenge. Considering the requirements by keeping patients at the heart of the design, including multiple channels to provide secure communication between hospitals to maintain the security of EHR. That is having a channel for each hospital. This created a problem of linearly growing channels which would create a load on the system not to mention the complexity it involves in implementing as well as maintaining. To avoid this scenario Private Data Collection is used as it is made as a simple alternative for multiple channels.

Implementation of Private Data Collection would create various challenges in different areas of the Hyperledger Fabric network, the First challenge would be in changing the patient's wallet information when he/she switches the hospital and the EHR is added to a new collection of a different hospital. As the old wallet information would be invalid as it would contain the mspID of the old hospital and the user is unauthorized to access the new collection. So the solution would be to dynamically switch the patient's wallet in the backend along with moving the EHR to a new collection. Then another challenge would be the implementation of a workaround for extracting transaction history as mentioned earlier due to the unavailability of APIs to extract history.

4.2 DRAWBACKS

One of the major drawbacks of using Hyperledger Fabric as a healthcare system is the database associated with it. As of now, the state database includes just CouchDB and LevelDB which store the data in JSON format. Usually, medical reports might contain lots of data including images of scans and other media files which would then result in large amounts of data and that causes various issues with CouchDB or LevelDB. This can be avoided by saving the actual data in cloud storage or any other database that supports easy maintenance of large data and then save the metadata like URL to the actual data, key values in the Couchdb. This certainly would take the data out of the distributed system and put it in a more centralized database which again renders the point of using Hyperledger Fabric unnecessary but the system can still provide that easy access to patients to their EHR[4].

This approach of patient data management assumes that the involved parties include the patients and the hospitals operate to help the patients and assume that the smart contracts deployed in each hospital follow this assumption. Also, in an ideal situation, other organizations like the insurance companies are involved in healthcare even if they require access to the EHR. It is assumed that no such organizations are involved in this prototype. However, including external organizations would hugely alter the way the system operates.

Scalability might require great deal of effort as the amount of data involved is large that is as mentioned in introduction "In Brazil's health domain, there were 1.4 billion patient visits just in 2018 carried out by its Unified Health System. In China, there were approximately 7 billion patient visits in 2017."[4]. Maintainability, security, infrastructure might lead to huge investment not just in terms of effort but financially as well.

In terms of Private Data Collection there still needs to be futher development in Hyperledger Fabric as explained in challenges it is of the area that lacks the ability to fully adopt the requirements of healthcare domain

4.3 COMPARISON WITH A TRADITIONAL DATABASE SYSTEM

Databases are usually centralized, controlled by database administrators, and provide a single point of failure. As a database administrator has all the rights over the database that means if database administrator security is compromised then whole database data can be manipulated Whereas blockchain decentralizes the data storage and creates multiple copies in form of multiple immutable distributed ledgers. Every participant in the network has its ledger and can see any modification in the ledgers, in this way it removes the single point of failure. In the case of conventional databases, database administrators might manipulate data whereas in blockchain if one point or one ledger is corrupted it will get corrected automatically using a smart contract (business logic of the network) and consensus. If a new participant joins the network and shares data with other participants, all the data will be immediately appended to every ledger and available simultaneously to every participant of the network.

Ledgers in blockchain are immutable means any data once stored in a blockchain ledger will remain there forever and any minor change is recorded in all distributed ledgers. CRUD (Create, Read, Update, Delete) operations are part of conventional databases in which one cannot only write new data and read all data but can update and delete already existing data as well. Whereas in blockchain only read and write operations are available due to the immutable nature of ledgers.

Blockchain provides integrity and transparency which means every participant knows data retrieved on his ledger is valid because copies of only valid data are appended in blockchain and every participant can see how the current state of data is achieved over time by looking at appending history.

5 CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Considering the complex scenario of the patient data management system as the main motivation and providing a solution through Hyperledger Fabric, to the involved problems in the current data management systems and methodologies, the proposed and implemented architecture using Hyperledger Fabric serves as a viable solution for the existing system. The components involved in the system can be tailored to meet the design requirements. This provided a lot of information on the complexity involved in managing a patient's health record, the challenges and drawbacks in implementing the system itself in terms of ease of use, security, scalability and maintainability, by modeling the scenarios with regards to how the actual application would behave. Apart from what was achieved through the prototype, discussions were also held on the theoretical concepts which, when implemented, could improve the system. A comparison was drawn between a traditional centralized database approach and a decentralized blockchain approach, to visualize the improvements the blockchain approach would provide and also to check the necessity of blockchain.

However, it can be strongly argued that managing medical data electronically using a system based on Hyperledger Fabric is indeed an effective solution. An EHR should be private, sensitive and should not be alterable. These features can be directly achieved using Hyperledger Fabric as it is distributed, trusted and immutable. To add another layer of security, accessing the medical history is maintained on the blockchain through transaction history recorded for the ledger. So a patient-centric system can be fabricated that grants better control of the data to the concerned person generating the data.

Such a system could solve some of the major problems in healthcare today like interoperability, but only when such systems are implemented at a full scale i.e. state or even on a national level. Unfortunately, there remain some open issues that need to be addressed before this can be achieved comprehensively. Since Hyperledger Fabric is a relatively new and growing strength, it lacks some capabilities such as getting transaction history if Private Data Collection is implemented. Requiring additional databases apart from CouchDB/LevelDB outside of Hyperledger Fabric in maintaining and complex data. And some other performance issues like the low maximum number of transactions possible per unit time are also present. In conclusion, it can be said that the Patient Data Management System is a feasible and indeed useful application in Hyperledger Fabric but there is some room for development within Hyperledger Fabric in this direction before we can have a finished product and there is a need for additional components or technologies outside of Hyperledger Fabric to make it successful.

5.2 FUTURE WORK

- Development on metadata concept: As of now, we are storing the actual patient data on the blockchain network. This may become infeasible if such a system is to be used as a National or a Global solution as the blockchain will become bulky. A better approach could be to store only the metadata related to the health records and accesses on the blockchain and the actual data can be stored in a traditional database. This hybrid approach may help us utilize the benefits of the decentralized system while at the same time also leveraging some advantages of traditional database systems.

- Implementation of Private Data Collection: The workaround for the private using the modified data structure can be implemented in the future to ensure security for the EHR. Also, if the next version of Hyperledger Fabric includes the retrieval of transaction history in private data then it could prove to be an efficient way to solve the problem of getting transaction history.

- Scaling the application: This is a prototype but we have designed the system considering the scalability of the application in mind. In a real implementation of such a system, a huge number of hospitals and other medical institutions will become part of the blockchain network, one organization could have multiple peers, there could be separate sub-channels and endorsement policies.

- Development of fail safe system by using multiple orderers: In Hyperledger Fabric v2.x, multiple orderers would use Raft service that follows the "Leader and Follower" model and provide a crash tolerant system, which means if there are three ordering nodes in a system and one node fails the ordering service would still perform using other two ordering nodes and by electing a new leader. As per Hyperledger Fabric official documents "Raft is the first step toward Fabric's development of a byzantine fault tolerant (BFT) ordering service"[10]

- Considering real-world data: Future work can be improved on giving input of real patient data and doing further testing on the process.

Part II

APPENDIX

A
AUTHORS

Abstract - Shubham Girdhar and Vineeth Bhat

1. **Introduction** - Md Towfic Aziz
    a) **Motivation** - Shubham Girdhar
    b) **Introduction to Hyperledger Fabric** - Faraz Shamim
    c) **Hyperledger Fabric as Healthcare System** - Vineeth Bhat
2. **Design**
    a) **Architecture** - Shubham Girdhar
    b) **Workflow** Faraz Shamim and Shubham Girdhar
        i. **Scenario 1: Patient visits doctor1in Hospital1 for the first time** - Faraz Shamim and Shubham Girdhar
        ii. **Scenario 2: Patient visits doctor2 in Hospital2** - Faraz Shamim and Shubham Girdhar
        iii. **Scenario 3: Patient has completed his/her treatment with Doctor1** - Faraz Shamim and Shubham Girdhar
    c) **Security** - Vineeth Bhat
3. **Implementation**
    a) **Tech stack** - Shubham Girdhar
    b) **Hyperledger Fabric network** - Vineeth Bhat
    c) **Database (Distributed Ledger)** - Faraz Shamim
    d) **Smart contracts and Chaincode** - Shubham Girdhar
        i. **Grant and Revoke Access** - Vineeth Bhat
    e) **Software Development Kit (SDK)** - Vineeth Bhat
        i. **Wallet** - Vineeth Bhat
        ii. **Application Programming Interface (API)** - Vineeth Bhat
        iii. **JSON Web Token (JWT)** - Vineeth Bhat
    f) **Application Frontend** - Faraz Shamim, Md Towfic Aziz and Shubham Girdhar
    g) **Security** - Vineeth Bhat
        i. **Alternate Solution to Private Data Collection** - Vineeth Bhat
4. **Discussion**
    a) **Challenges** - Vineeth Bhat

   b) **Drawbacks** - Vineeth Bhat
   c) **Comparison with a traditional Database system** - Faraz Shamim

5. **Conclusion and Future Work**
   a) **Conclusion** - Shubham Girdhar and Vineeth Bhat
   b) **Future Work**
      - **Development on metadata concept** - Shubham Girdhar
      - **Implementation of Private Data Collection** - Vineeth Bhat
      - **Scaling the application** - Shubham Girdhar
      - **Development of fail safe system by using multiple orderers** - Faraz Shamim
      - **Considering real-world data** - Md Towfic Aziz

B SOURCE CODE

The source code for this prototype can be found in the following github repository[13]
Instructions on how to setup the network, trouble-shooting, setting up the SDK, frontend can be found on the Readme file in the repository.

BIBLIOGRAPHY

[1] *ACTION-EHR: Patient-Centric Blockchain-Based Healthcare Data Management for... Alevtina Dubovitskaya*. URL: https://www.youtube.com/watch?v=mH5jUNaiejs. Accessed: 29.03.2020).

[2] *Blockchain in healthcare: The Ultimate use case?* URL: https://blockgeeks.com/guides/blockchain-in-healthcare/. Accessed: 29.03.2020).

[3] *Fabric chaincode lifecycle*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/chaincode_lifecycle.html. Accessed: 30.03.2020).

[4] Andressa Fernandes, Vladimir Rocha, Arlindo Conceicao, and Flávio Horita. "Scalable Architecture for sharing EHR using the Hyperledger Blockchain." In: Mar. 2020, pp. 130–138. DOI: 10.1109/ICSA-C50368.2020.00032.

[5] T. Mikula and R. H. Jacobsen. "Identity and Access Management with Blockchain in Electronic Healthcare Records." In: *2018 21st Euromicro Conference on Digital System Design (DSD)*. 2018, pp. 699–706. DOI: 10.1109/DSD.2018.00008.

[6] Harman Puri. *A Complete List of Blockchain Platforms — 2020*. URL: https://medium.com/wikidlt/a-complete-list-of-blockchain-platforms-2020-49cf01ee6688. (accessed: 14.03.2021).

[7] *Query history of private data - by key, block, or transaction id*. URL: https://jira.hyperledger.org/browse/FAB-5094. Accessed: 29.03.2020).

[8] *Smart Contracts and Chaincode*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/smartcontract/smartcontract.html. Accessed: 30.03.2020).

[9] *Software Development Kit*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/glossary.html?highlight=Software\%20development\%20kit\#software-development-kit-sdk. Accessed: 29.03.2020).

[10] *The Ordering Service*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/orderer/ordering_service.html. Accessed: 30.03.2020).

[11] *Using the Fabric test network*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/test_network.html. Accessed: 30.03.2020).

[12] *Wallet*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/wallet.html#scenario. Accessed: 29.03.2020).

[13] *hyperledger-fabric-patient-data-management*. URL: https://github.com/Shubham-Girdhar/hyperledger-fabric-patient-data-management.git. Accessed: 31.03.2020).