Using Blockchain for Electronic Health Records

Ayesha Shahnaz

MS-17 CSE

205513

Supervisor: Dr. Usman Qamar

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

2019

Using Blockchain for Electronic Health Records

Author

Ayesha Shahnaz

2017-NUST-MS PhD-CSE-205513

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Computer Software Engineering

Thesis Supervisor:

Dr. Usman Qamar

Thesis Supervisor's Signature: _____

DEPARTMENT OF COMPUTER AND SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,

ISLAMABAD

October 2019

Declaration

I certify that this research work titled "*Using Blockchain for Electronic Health Records"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Ayesha Shahnaz

2017-NUST-MsPhD-CSE-205513

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Ayesha Shahnaz

2017-NUST-MsPhD-CSE-205513

Signature of Supervisor

Dr. Usman Qamar

Copyright Statement

Acknowledgements

In the name of Allah, Full of Compassion, Ever Compassionate

All praises be to Allah the Lord of the Universe. Let His regards and salutations be upon the Holy Prophet Muhammad and his family and companions.

I would like to express my deep and sincere gratitude to my thesis supervisor, Dr. Usman Qamar for providing invaluable guidance throughout this thesis work. It was a great privilege and honor to work and study under his guidance.

I am thankful to Dr. Saad Rehman and Dr. Wasi Haider for being on my thesis guidance and evaluation committee, and also for their support and cooperation. In addition, I would also like to express my special thanks to Dr. Ayesha Khalid for her help throughout my thesis this work would not have been possible without her guidance and support at every step.

I am profusely thankful to my beloved parents who are the reason of what I become today and for their great support and continuous care. I am also grateful to my sisters who have been my inspiration and strength.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment

Abstract

Blockchain have been an interesting research area for a long time and the benefits it provides have been used by a number of various industries. Similarly, the healthcare sector stands to benefit immensely from the blockchain technology due to security, privacy, confidentiality and decentralization. Nevertheless, the Electronic Health Record (EHR) systems face problems regarding data security, integrity and management. In this paper, we discuss how the blockchain technology can be used to transform the EHR systems and could be a solution of these issues. We present a framework that could be used for the implementation of blockchain technology in healthcare sector for EHR. The aim of our proposed framework is firstly to implement blockchain technology for EHR and secondly to provide secure storage of electronic records by defining granular access rules for the users of the proposed framework. Moreover, this framework also discusses the scalability problem faced by the blockchain technology in general via use of off-chain storage of the records. This framework provides the EHR system with the benefits of having a scalable, secure and integral blockchain-based solution.

Keywords: blockchain, health records, electronic health records, decentralization, scalability

Table of Contents

Chapter 3 Literature Review

Chapter 4 System Design and Architecture

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

The advent in technology is affecting all parts of human life and is changing the way we use and perceive things previously. Just like the changes that technology has made in various other fields of life, it is also finding new ways for improvement of healthcare sector. The main focus of such revolutionary changes done by the technologies is the improvement of security, user experience and many other aspects of healthcare sector. Recently there has been much attention on making health-care sector to be patient-centric. For this many systems have been developed and are widely used in some institutions, such as Electronic Health Record (EHR) and Electronic Medical Record (EMR) systems. But these systems also face some problems related to security, user ownership of data, data integrity etc. The solution to them could be the use of blockchain technology in healthcare sector. This technology is helpful in not only making the health care sector to be enable ownership of patient medical data but will also make it secure.

1.1. Background and Motivation

Before the advent of modern technology, healthcare sector used to store patient's medical record using handwritten paper-based mechanism. This paper-based medical record system was inefficient, not secure, unorganized and was not temper-proof. It also faced the issue of data-duplication as all the institutions that patient visited had various copy of patient's medical records thus, creating the problem of data duplication. Also the right of ownership of medical data was only given to providers and not the patients.

1.1.1. Electronic Health Records (EHR) System

Recent innovation in healthcare shifted this paper-based medical record system to Electronic Health Record (EHR) systems. EHR is a system that was designed to combine both paper-based and electronic medical records (EMR) in order to improve the quality of healthcare sector. This system includes all the necessary information of patient such as demographics, progress notes, medications, medical history, laboratory reports and results etc [1]. The main goal of EHR systems was not only to store patients' medical records electronically or to replace the previously used paper based system but this system was also intended to offer quality and security to the healthcare sector [1].

The EHR systems have been implemented in a number of hospitals around the world due the benefits it provides, mainly the improvement in security and its cost-effectiveness. They are considered a vital part of healthcare sector as it provides much functionality to the healthcare [2]. The basic functionalities these system offers are electronic storage of medical records, patients' appointment management, billing and accounts and lab tests. These basic functions are available in many of the EHR system being used in the healthcare sector.

The EHR systems are designed while keeping in mind that the traditional patient-doctor relationship has now changed as there is involvement of some other stakeholders in it, such as insurance companies etc. The functionalities of EHR systems are utilized by many individuals in a healthcare organization such as doctors, patients, nursing staff, administrative staff etc. These functionalities utilized call for the need to make data sharing, a primary aspect of the EHR systems and this is also the main reason why paper-based records were no longer useful as they cannot be used for effective data sharing among health care providers or other concerned individuals in the healthcare organization.

The data sharing feature is actually known as Health Information Exchange (HIE) in broader terms. It is labeled as the process of sharing or exchanging the healthcare related data of patient electronically among various healthcare providers or hospitals [3]. The HIE aspect of EHR systems would greatly benefit the healthcare sector as it would avoid data redundancy, inconsistency and integrity problems associated with the medical data being stored.

Besides data sharing, EHR systems provide many other benefits to the healthcare sector the most prominent of it being the minimization of medical error rate. As now the information of patient is stored electronically, and with the decision support systems provided along with EHR system the doctors' have a lower chance of making any medical mistake. They can use these systems for controlling the dose level of a patient thus preventing a medication mistake. It would also help the healthcare sector to ensure patients' security.

1.1.2. History and Evolution of Electronic Health Records (EHR) System

As mentioned earlier before the EHR systems were introduced in healthcare sector, the handwritten notes and records were the means by which the patient's records were maintained. This manual process was used for quite some time in the healthcare sector but it was replaced when the idea of computer-based record saving system was introduced in the year 1991.

1.1.2.1. State of EHR System in 1991

With the help and sponsorship provided by the Institute of Medicine (IOM) in the beginning of year 1991, the concept of current electronic health record system was introduced [4]. The name Electronic Health Records (EHR) was assigned by IOM to this concept in 2003. And the ultimate goal of this system was provide safety and quality to the healthcare domain which was missing in the paper based system. A report containing the core functions of this idea were presented by IOM, which greatly influenced the EHR systems that are being used now days by many healthcare organizations. These core functions are as follows [4],

1. Health information and data
2. Result Management
3. Order Management
4. Decision Support

5. Electronic communication and connectivity
6. Patient Support
7. Administrative processing and reporting
8. Reporting and population health

These were the 8 core functions proposed in IOM report and these functions also hold an importance in today's EHR systems as well. These functions are comprehensive and are able to fulfill the basic operations that are required of a system that is made purely with the intention of serving the patient with the quality healthcare systems.

1.1.2.2. State of EHR System in 1992

With the introduction of computer based record saving system by IOM in 1991 a growing need of using the hardware needed for this system also arises. In this case the hardware during early 1980s and 1990s became quite affordable which helped in this system being considered as an option for solving healthcare problems. During these years when at one side the hardware was becoming affordable and the personal computers were also being commonly used in the public sector. This also helped the EHR system as the personal computers were relatively easier to use and understand which would help in the adaptation of this system in healthcare organizations or in general hospitals mainly.

Despite of all of these benefits which mainly include security and quality, provided by the EHR systems these systems were not widespread in the healthcare due to certain problems and issues they faced. These issues were as follows [5]:

- Physician's resistance towards adaptation and acceptance of the EHR systems
- High costs associated with these systems as the already prevalent systems in hospitals were paper-based manual systems so a huge initial cost was needed for converting it completely into the computer-based system
- Errors occurring during the data entry from converting the paper-based records to the computer-based records
- Lack of any incentives that could be considered as real and possible for these systems

These issues were more concerning for healthcare sector as they were already facing inadequacies in the paper-based manual system of records saving. So, to adapt a system that would cause more problems than the already existing system's problem was not considered a feasible choice. Despite of all these problems the development was going on for making the healthcare to be more computerized in order to ease the problems caused by the manual systems. Although these developments and solutions were not being widely used but still they were known to some public sectors.

With the boom of technology when the personal computers started being accepted by the general public in other sectors of life the healthcare sector also started their use for some tasks such as

viewing the laboratory reports, patient care and measurements, physician and nursing notes, keeping the record of consultations [5]. Along with these features and functions many other were also helpful in normalizing the use of such a computer based for nursing staff and physicians these were graphical representations of patient medical records, electronic documentation of medical records etc. [5]

With the new development and technologies being introduced in the EHR systems the hospitals started adopting them along with the data management systems which were specifically focused for patients' records. These systems were used for storing the medical records of the patient along with the records that were gained by the monitoring systems attached by the patient's bedside while they were in the hospital [5]. These systems were mainly used for recording the real time monitoring of patients' records while they were undergoing a treatment and this eventually helped the physicians as now this task was automated and they won't have to waste time adding this manual to their file based records.

These EHR systems were developed by trusted parties and funded by government as in the case of IOM it was a division of National Academic Sciences, Engineering and Medicine who sponsored the idea of computer based records. But soon third party software or ideas started getting included in the EHR systems which called for the need of some standards to be set for the healthcare sector on its whole while using these systems.

1.1.2.3. Health Insurance Portability and Accountability Act (HIPAA)

The *Health Insurance Portability and Accountability Act (HIPAA)* is an act which was defined for maintaining some security rules for healthcare organizations. Its purpose was not for defining the EHR systems and its usage rights but instead the purpose of this act was to explain the rules healthcare organizations would follow while using EHR systems which broadened the definition of EHR systems [4].

This act was introduced in 1996 due to the requirement of protection of privacy and security of healthcare related information. The role of the Secretary of the U.S. Department of Health and Human Services (HHS) was to develop these security rules that would protect the healthcare sector against any possible security concerns. These rules are of two kinds that are HIPPA privacy rules and HIPAA security rules. The HIPAA privacy rules and HIPAA security rules would be enforced by the Office of Civil Rights (OCR) [6].

The HIPAA privacy rules were for protection of individual patients' medical and personal health information records. These privacy rules were also addressed for certain other entities of the healthcare sector as it sets limits for the protection of the personal healthcare information of the patients. This rule also provided patients the right of getting a copy of their medical records from the healthcare organization.

The HIPAA security rules were defined as the national security rules for protection of computerized healthcare information and records. These rules were intended for all of the healthcare related stakeholders and entities, the basic aim was to secure the healthcare information from any unauthorized and malicious use. As the data was now in electronic form so it should be made secure from any such access which would cause security concerns for the electronically saved medical records.

Before the HIPAA act was defined by HHS there were no defined standards that would help to secure the healthcare information. This was a problem frequently faced by the healthcare organizations as they were adopting the EHR systems which made their records and general structure of management to be automated and technical. This caused them to face certain problems which included the decisions of providing the patients personal medical records, sharing of medical and healthcare information, rules and regulations for security of this information and maintaining the privacy of the patients medical records. These problems were important and were also being faced by many healthcare organizations, which called for the need of such a standard that would help them in maintaining the quality of EHR systems and in general the healthcare services provided by them.

Also as the new technologies were being developed evolved and adapted in the healthcare organizations the security and privacy rules were needed more than ever for enforcement of proper authorization rules for these records. There was also a need that these standards should be flexible so that any new technologies could be introduced in the hospitals i.e. enforcement of these standards should not halt the process of development in the healthcare sector.

1.1.2.3.1. Brief overview of HIPAA Rules and Regulations

The HIPPA act aims to tackle the three most important aspect of healthcare records and information [6] i.e.

- Integrity
- Confidentiality
- Availability

It is quite understandable that why these aspects are important to be kept in HIPAA act as these make up the basis of security and privacy of any information being stored or shared electronically. The following section would explain these in detail.

- *Integrity* is the process of maintaining the information or data in a way that it becomes undivided or intact. The quality of that information should not change by it being shared to other entities it should remain unchanged. In literal terms integrity refers to the honesty i.e. anything should be what it shows or claims to be. So, in case of electronically saved records and information of patients it should also remain intact and should not change without any proper reasoning and justification. So, this is the first aspect discussed by the

HIPAA in their rules and regulations that patient medical records or in general the healthcare information should be kept intact and these systems should maintain the integrity of these records.

- *Confidentiality* is the process of keeping any data or information secure from any unrelated party or entity. This term is used for defining that information of an individual should be kept secure from any third party access. It is a purely security and privacy related aspect of the HIPAA which was again defined for electronic records as the automated electronic records system being replaced by the computerized records saving systems i.e. EHRs were storing data on the databases or any other platforms. This information or record needed to be made secure from any third party unauthorized access.
- *Availability* is the process of making the information or record to be available whenever it is needed by the user. In automated or computerized systems mainly this is an important aspect which must be fulfilled to keep the system running and functional. In case of HIPAA act this was an important factor due to reasons that medical or healthcare information should be made available to the entities (physicians, nursing staff, administrative staff, patients etc.) whenever needed. While making the system secure by applying the aspects of integrity and confidentiality it must be kept in mind that these records that are being made secure must also be readily available when needed. As these records hold importance in making certain decisions regarding the patients' health.

The above mentioned aspects were in broader term used for defining the rules and regulations of HIPAA act. There are also some separate roles and regulations that are specifically defining how these aspects could be maintained in greater detail. These rules were helpful or intended for administration, technical and physical concerns related to security [6].

There were certain measures defined for related entities in these rules and regulations for helping them in understanding that what measures could be taken while implementing the rules and regulations. These measures could also be considered as considerations these entities must have while implementing any new security measures in their systems. They were [6],

- The infrastructure needed for implementing these security updates in the system or any other measures in the existing framework i.e. hardware, technical and software needs should be kept under review while implementing them.
- The complexity, size and costs associated with including those measures in the existing working body of the system.
- Any risks that implementation of these security measures would cause to our system or its users.

The safeguards defined by the HIPAA act were that of administrative, technical and physical which we have mentioned before as well. These make up an important part of the healthcare organization as the adaption of EHR systems in particular need the support and help from these 3 entities i.e. administration staff, technical staff and physical requirements. They help in defining

the way a system could be used in any organization. The safeguarding measures for these entities could be seen in the following table.

Table 1: Measures to secure the HIPAA aspects

	Measures
Administrative Safeguards	• To reduce potential risks associated with the implementation of security measures • A personnel who is responsible for maintaining the security, and implementation of defined policies and procedures • To train the staff or working body of the organization for the defined policies and procedures regarding security • Authorization and defining the access available to the administrative staff • To evaluate the policies and procedures related to security in order to meet the security rules requirement
Technical Safeguards	• Policies and procedures should be implemented to make sure that any information related to healthcare is not changed or altered i.e. to ensure integrity of information • Only authorized individual should be allowed to access the healthcare information and proper rules and procedures should be implemented to control the access rights • Over an electronic network some technical measures to secure against the unauthorized access to the system must be implemented
Physical Safeguards	• Physical measures should be taken to forbid the unauthorized parties from accessing the healthcare information • To physically secure the devices and other platforms where the healthcare information is stored, updated or viewed. These platforms holding the information should be made secure by properly following the defined rules, regulations and procedures

The reason due to which these standards were introduced and enforced in the healthcare organizations using the EHR systems was also due to the fact that there were no universally defined standards already prevalent for such systems. When the information stored in these EHR systems was communicated or shared to other systems certain problems were faced as the formats for saving these records were not similar due to not defined standard. This called for the

need of defining the universally agreed upon semantics for EHR system [5]. The definitions of these semantics were monitored by the National Library of Medicine [5] and these were then enforced in all of the EHR systems being used.

Another important aspect of these electronic health record systems was ONC certification which is explained in the following section.

1.1.2.4. ONC Certification

The term ONC stands for Office of National Coordinator for Health Information Technology which is used for Health IT Certification Program for modules related to healthcare in Information Technology (IT) [7]. There are certain standards that are enforced using the ONC based certification program which mainly include the Health IT standards and Implementation specifications etc. The ONC is also a part of United States Department of Health and Human Services (HHS). Its purpose was to promote the development of healthcare information technology (HIT) by following the certification and requirements defined by the ONC.

The ONC was designed mainly for defining standards and policies for HIT, for enforcing a common a national standard in the healthcare sector. The mission of ONC is explained as follows [8],

- Reducing costs related to healthcare sector with the improvement of quality service provided by this sector
- To improve the information sharing between hospitals, laboratories and any other healthcare organization
- To secure the healthcare information and records

The ONC provides the EHR systems with certifications using which they could be considered certified for use in the healthcare organizations. In order to get this certification certain standards should be met along with the EHR systems holding certain capabilities [8].

1.1.2.4.1. Current State of EHR System

With the advance in technology and with standards being enforced for improving the quality, security of EHR systems they are being used in a number of healthcare organizations. Obviously these standards could not fully enforce that no issues arise for the EHR systems but they helped in the adaptation and acceptance of these systems. The EHR systems or vendors of such systems that are used by the healthcare sector in these years are listed as follows,

- Epic
- Cerner
- MEDITECH
- Evident

- Allscripts
- MEDHOST
- Netsmart Technologies
- athenahealth
- Harris Healthcare
- Indian Health Service

Table 2: EHR Systems / Vendors Detail

EHR System / Vendor	Description	Market Share
Epic	*Used in:* Retail clinics, community hospitals, academic medical centers, rehab centers etc.	30.9%
Cerner	*Used in:* its EHR systems are widely known in public sector	25.1%
MEDITECH	*Used for:* recently offered solution for Web Ambulatory EHR	14.7%
Evident	*Used in:* rural healthcare organizations	8.1%
Allscripts	*Used in:* health systems, healthcare facilities, health management solutions	5.7%
MEDHOST	*Used by:* more than 1000 healthcare facilities	5.5%
Netsmart Technologies	*Used in:* behavioral health, social services, post-acute care facilities	1.5%
athenahealth	*Used in:* community and rural hospitals	1.3%
Harris Healthcare	*Used for:* provides services for improvement of quality, security of patient care in its EHR systems	0.8%
Indian Health Service	*Used for:* its EHR systems offer to capture the public and clinical health related data	0.5%

1.1.3. Current Gaps in EHR System

Many countries have taken measures to implement the EHR systems in order to bring efficiency and data sharing with ease in their healthcare sectors. The basic focus is to provide medical record whenever they are needed and they must be secure, temper-proof and could also be shared

across different platforms. But there are also some problems associated with these systems which are being discussed in the following sections.

Interoperability

Interoperability is the way for different information systems to exchange information between them. The information should be exchangeable and must be usable for further purposes. As explained before, Health Information Exchange (HIE) or data sharing is an important aspect of EHR systems. There are a number of EHR systems deployed in various hospitals all of them having different terminologies, also having varying level of technical and functional capabilities. Due to these contrasting ways there is no single universally defined standard that acts as interoperable format for data exchange and sharing [9]. The interoperability aspect is missing in many EHR systems as this problem stands on the technical and cultural level. At technical level the information or medical records being exchanged should be interpretable, and that interpreted piece of information could be further used [9].

Another important division in interoperability is the institution-centered interoperability and patient-centered interoperability. Institution-centered interoperability has hospital, care providers acting as the business entities that are responsible for information exchange [10]. While patient-centered interoperability offers that patient is the entity that has the control for information exchange. It places patient at the center of authority and let him take the exchange decisions. This interoperability feature is missing in many EHR systems that raise concerns as the patient's data won't be exchangeable and interpretable.

Information Asymmetry

Today the greatest problem in healthcare sector defined by the critics is information asymmetry [11], i.e. the one party having better access to information than the other party. In case of EHR systems, or in general healthcare sector is suffering from this problem as doctors or hospitals have access to the patient's records, thus making it central.

A patient who intends to access his medical records would first need to request the administrative staff to grant access to view his personal medical records. Also, in the cases where the patient's medical record is saved in a hospital's EHR system and patient visits another hospital for consultation he would need that previously stored record. For that he would have to follow the whole process of requesting access to the records from the administration, followed by verification and authentication process. This whole process is long and tiring for the patient to follow. And this happens because patient does not have access to his own records due to information asymmetry in the healthcare sector. The information is centralized to only a single healthcare organization and its control is only provided to the hospitals or organizations.

Data Breaches

Patient medical data is a vital part of healthcare sector that needs to be secure and intact under all circumstances. Any attack on this information makes the whole healthcare organization vulnerable and patient's personal records are also at stake. So, EHR systems are developed under the security requirements that must be fulfilled to make it secure for storage and exchange of sensitive medical information. But recent data breaches in healthcare sector calls for the need of a better platform and some measures to be taken, as reported by HIPAA journal in the year 2018 there were 2,456 data breaches of healthcare and it involved 500 records in it [12]. Also as reported by Forbes, the numbers of data breaches have increased to 70% in the past seven years and the reasons behind most of these breaches are related to IT incidents or in general hacking [13].

The privacy, security concerns that arise with the data breaches in electronic record storage causes to be a hindrance in adaptation of EHR systems [14]. The research has been done to tackle these breaches and include various safety measures to ensure security. But these systems still lack the way they secure these records as the data breaches statistics keep on rising.

Table 3: Gaps in EHR System

Current Gaps in Electronic Health Record (EHR) System	
Interoperability	A number of EHR systems devised across various hospitals and all follow different terminologies, technical and functional requirements. Due to this EHR systems suffer from lack of interoperability standards.
Information Asymmetry	The current system devised in hospital or healthcare organizations have an asymmetric information control. Care providers and doctors have better access privileges to the patient's data. While patient needs to follow a long process to access his personal record.
Data Breaches	Security and privacy concerns related to EHR system make it problematic for it to be adapted across various hospitals. The growing number of data breaches makes the sensitive patient record to be vulnerable to unauthorized access.

These problems make it reasonable to find a platform that would be helpful in transformation of healthcare sector i.e. Blockchain. A platform which would be secure transparent and would also provide data integrity to the medical records of the patients. Blockchain technology is explained in detail in the other sections.

1.1.4. Objective and Contribution

The aim of this thesis endeavor is to improve healthcare sector by storing records on the blockchain. We intend to create such a decentralized platform that would store patient's medical records and give access of those records to providers or concerned individuals i.e. patient. We also intend to solve the scalability problem of blockchain, as it is not in the design of blockchain to store huge volumes of data on it. So, we would use off-chain scaling method that makes use of

the underlying medium to solve the scalability problem by storing the data on that medium. In our case, we intend to use InterPlanatery File System (IPFS) this would help to solve the scalability problem as now the patient's record would not be stored on the main blockchain but on the IPFS. As explained before, IPFS is a distributed file system, which uses a peer-to-peer network for file storage and sharing.

So, our contribution is two-core:

1. Storing medical records on the blockchain: a secure and temper-proof platform that would only be accessible to trusted individual. That would be secure and would also provide the advantage of usability, confidentiality, privacy etc.
2. Solving scalability issue of blockchain: using off-chain scaling solution of IPFS protocol.

The challenges faced by EHR systems that were discussed in previous section are addressed by the Blockchain technology. The information is not centralized and it does not have one medium that controls its access privileges. Using this technology it can be ensured to provide granular level access to all the participating entities i.e. doctors, patients, nursing staff, care providers, pharmaceutical companies and insurance companies.

1.1.5. Outline

The second chapter of this thesis document contains the introduction to blockchain technology and third chapter contains the literature review and comparisons to the previous work or studies conducted in this domain.

CHAPTER 2

INTRODUCTION TO BLOCKCHAIN TECHNOLOGY

2.1. Introduction

This technology was introduced by Satoshi Nakamoto [15], for his popular work of digital currency or crypto-currency i.e. bitcoin. Nakamoto used blockchain technology to solve the double spending problem of bitcoin but soon this novel technology was being used in many other applications.

Blockchain is a chain of blocks that are connected together and are continuously growing by storing transactions on the blocks. This platform uses a decentralized approach that allows the information to be distributed and that each piece of distributed information or commonly known as data have shared ownership. Blockchains holds batches of transactions that are hashed thus providing them security and they are managed by peer-to-peer networks. A blockchain has certain benefits such as security, anonymity, and integrity of data with no third party intervention. These benefits make it a reasonable choice to store patient's medical records on it, because the innovation of technology in the healthcare industry has made the security of patient's medical data a top priority. Using this technology makes it feasible for patient's records to be encrypted and accessible across different platforms with the consent of patient. A number of researchers have also identified that using blockchain technology in healthcare would be a feasible solution [10] [16] [17].

2.1.1. A World without Middleman

The basic benefits of blockchain technology in general are explained in the above discussion but the most important benefit provided by the blockchain technology is the end of having an intermediary between transactions of two parties. This is known as removing the middleman between your agreements and transactions. Blockchain technology provides such a platform that would allow your agreements to not have a middle party controlling all of the transactions, validating these transactions. This middle party is responsible for any sort of services needed and required by its clients.

The process followed by the applications not using the blockchain technology is that of client server where the central party is in this use case is being referred to as the middleman who is controlling this whole process. The middleman has the authority over all the connected nodes in a computing network.

If considered in general terms, let us consider an example where two parties are signing an agreement with the help of a middleman. In this scenario the important task would be the

agreement of the parties to the terms and conditions of the agreement. This agreement would be handled by the middleman who would initiate the agreement process, make the two parties understand the terms and conditions of the agreement and validate the agreement. The middleman would also be responsible to get the two parties sign the agreement. This middleman could be an agent, or any other individual responsible for performing this important task. In the case of banking scenario this middleman would be the bank whose resources are being used for transfer of money from one party to another.

This mechanism of using a middleman for such an important task is not supported due to the obvious reasons the authority becomes central and if this middleman is vulnerable to any attack or worse if it is exposed to any attack that would corrupt the whole process of transaction or agreement occurring between parties or entities. Using a process that has no middleman would ensure an effective and efficient environment. As it would help the whole process to be done in a better way with effective costs associated with the whole process.

Blockchain technology uses a peer-to-peer network that does not allow any middleman to act as the authority in the network. There are defined consensus algorithms used in this technology that would help with the authentic way this technology operates the transactions or any other task done on them.

2.1.2. Blockchain Architecture

The basic architecture of blockchain can be understood as that it is a sequence of blocks connected together to form a system that is used to store transactions just like a traditional ledger [18]. Once some data is stored inside the blockchain it becomes difficult to change it as blockchain uses peer-to-peer network, and it consists of many computers that are responsible for the transactions (data) on the blockchain. All of this is managed by a consensus between all the involved parties on the blockchain. There are many consensus algorithm that are used in various kind of blockchain but the most common of all is proof of work (PoW) consensus algorithm.

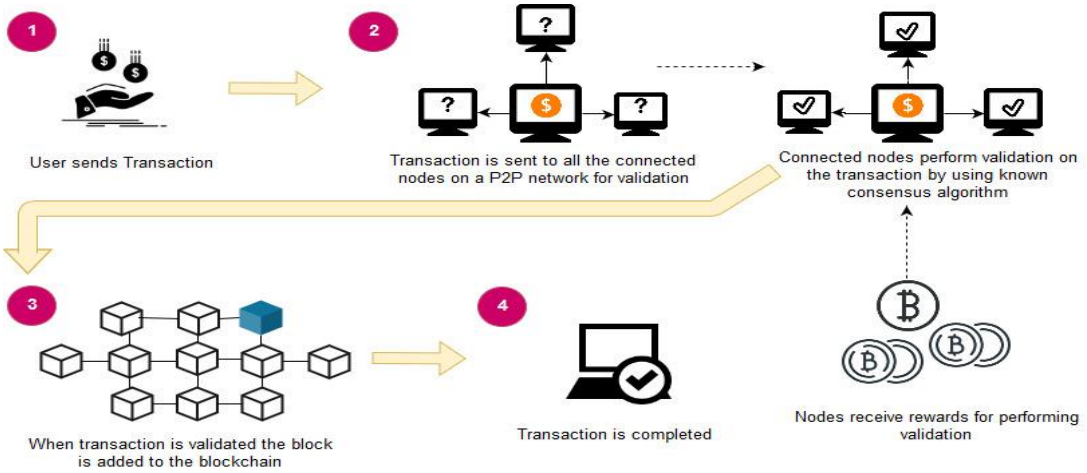

Figure 1: Blockchain Architecture

The blockchain architecture can be more easily understood by a simple scenario where a user on the blockchain network sends a transaction. The transaction of the user is broadcasted to all of the connected nodes in the peer-to-peer network of blockchain. The purpose of this broadcasting of transaction to all the nodes is to validate that transaction. This validation is performed by the connected nodes using some known algorithms to verify the transaction and to ensure that sender is an authenticated part of the network. When a node succeeds in performing the validation that node is rewarded with bitcoin. This process of validating the transaction is known as mining and the node performing this validation is known as miner. This concept would be explained in later sections of this chapter. After, the transaction is validated that block is added to the blockchain and transaction gets completed. This whole process is defined in Figure 1 above. Some basic concepts of blockchain technology can be understood in the following descriptions.

2.1.3. Peer to Peer Network

As mentioned in the architecture of blockchain, this technology uses the peer-to-peer (P2P) network of computers for validation and transformation of transactions being sent on it. As the idea was to have a distributed technology that would not be controlled by any central entity due to this the blockchain makes use of P2P network where the connected nodes in a network act as peers to each other. No node connected in this network has the authority to lead or control the whole network but in actual all of them have the same level of control in the network.

These peers connected in a network enjoy the same control or in more specific terms equally privileged [19]. These nodes also share the same resources i.e. these nodes that are acting as peers to each other in a network make a part of their resources available to the other nodes in this network due to which they can have access to them without the need of a central authoritative party [19]. By resources here we refer to processing power and disk storage etc. which is made available for other nodes in this network.

The other model used in contrast of this P2P network model is client-server model where a server is responsible for managing and defining the network rules, whereas the client requests server for using any resource or performing any task. In this network the server has the control over network and fulfills the client's requests. If the server becomes busy or gets crashed the whole network gets affected by this, although there are many methods and mechanisms to control the server from getting damaged but still this networking model could not be used in blockchain technology which is being built on the idea of decentralization and no entity acting as the controller in any scenario. Moreover, in a client-server networking model the resources are divided among the nodes connected in it, which gave the nodes complete control over them and these cannot be shared easily with the other connected nodes.

The following figure represents the way these networks have nodes connected inside them. The P2P network has nodes connected in a manner where every node is connected to each other and

there is no central authority controlling the whole network. In contrast to P2P network the client-server network has a server that exists at the center of the network and is controlling all the client nodes connected in the network. The server node has the authority over the whole network which does not allow decentralization in the network. The client-server network model supports the idea of centralization whereas the P2P network model supports the idea of decentralization.

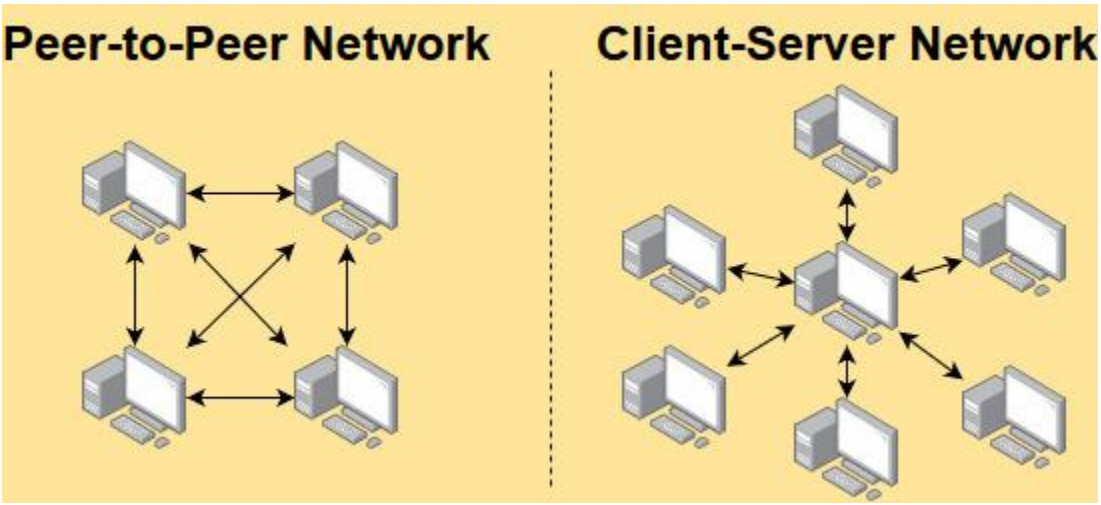

Figure 2: Network Models

Before moving on to the details of P2P network model let us first understand the centralization and decentralization, their differences and benefits offered by them and also the possible applications of them in various domains.

Centralization

The concept of centralization could be understood as the scenario where there is a central authority controlling all of the resources and important decisions of that network. In terms of an organization centralization can be taken as when the higher level stakeholders have the right of making decisions. In case of computing, centralization occurs when one node has higher level of control then the other nodes connected in the network. This node having the control is known as the central node and it owns all the resources of the computer network. This whole concept is known as centralization. The client-server network uses the centralization concept where a server is responsible for making any control decisions in the networks and the client nodes request for the services and resources from the server node.

The problem faced by the centralization is that the other nodes in the network are dependent on the central node. The reliance of the connected nodes on the central node could cause the nodes to lose resources and services in case of central node being non-functional due to any reason. In simple terms, if the central node is attacked by any third party the connected nodes would also become vulnerable to the attack caused by third party.

Decentralization

With decentralization there is no single node connected in the network that can control the whole network and its resources. In simpler terms there is not point of control which is central in the network or in any other scenario. It is the exact opposite of centralization where a node exists at the center and is responsible for all of client nodes connected in the network. But all of the nodes present have their own roles instead with no central node having the complete authority over the network and its resources. Each node in the network has equal number of resources available to their workstations.

The P2P network follows the same model of decentralization where all peers have equally allocated resources to them. There is no authority existing that would control these peers and their services.

2.1.3.1.Peer to Peer Architecture

Now let us understand the architecture of P2P network which is an important aspect of blockchain technology in general. The following section explains this architecture.

As explained above P2P networks have peers that are connected to each other to form this network. These peers are actually nodes existing in the network and these are acting as both client and server in the P2P network. But the P2P network has two different kind of networks in it which differentiate by the way the nodes acting as peers are connected to each other i.e. the order in which these nodes are located in the network and how the resources are shared [19]. These two networks are,

- Unstructured networks
- Structured networks

Unstructured networks or unstructured P2P networks do not follow a particular structure to arrange the nodes connected in it. This kind of network has nodes connected in a random manner with no defined structure i.e. the nodes are connected in a random way which could be optimized by the network participants [19]. For sharing data or looking up any data throughout the network a node would send a query to all of the connected nodes or peers in the network. This request would be sent to all of the randomly connected nodes in the network and would be send back to the requesting node after looking it up throughout the unstructured P2P networks.

Structured networks have the nodes connected in the way where a topology is followed to organize these nodes. These organized nodes are acting as peers where a protocol is used for looking up other nodes in the network [19]. In this network a hashing table is maintained by nodes that allow these nodes to be responsible for a part of the content of the network. This hashing table is known as Distributed Hash Table (DHT). These hashes are assigned to every peer connected and they are stored in the hashing table i.e. DHT for using it later on to look up

for assigning any content to these peer. For this assigning of contents it uses a protocol which helps in determining that which node should be responsible for which task or content [19].

2.1.3.2.Other Network Models

Along with the above mentioned two kinds of networks in P2P architecture there is also another model known as Hybrid Model. This is the combination of both peer-to-peer and client-server networking models. It also allows the nodes to act as peers in the network and also have a central node controlling those peers. This allows for an efficient networking of the connected nodes in the network. This model is considered to perform better than the above mentioned unstructured and structured networks. Another advantage offered by this hybrid model is that it can be used for centralization and decentralization purposes [19].

2.1.4. Block

As explained earlier blockchain are formed together by a number of blocks connected together in a peer-to-peer network thus making a decentralized application. These blocks contain the hashes in their header of the previous blocks on the chain. These blocks contain three things in them:

- Data
- Hash of current block
- Hash of previous block

The data could be anything as it depends on the type of blockchain. As in case of bitcoin, the data consists of coins that are actually electronic cash [15]. The hash that is stored in these blocks contains a SHA 256 cryptographic algorithm. This hash is used for unique identification of a block on the chain and due to this reason they are stored on the previous block so they can be connected to each other.

2.1.5. Consensus Algorithm

Each block that is added on the chain would need to follow some consensus rules for it to be added on the blockchain. For this purpose blockchain technology uses consensus algorithms. The most common consensus algorithm used is Proof of Work (PoW) algorithm and it was used by Nakamoto [15], in bitcoin network. The basic working of this algorithm is that there are number of nodes or participants on a blockchain network so when a transaction is requested to be added on the network by any participating node it needs to be calculated. This process is called mining and the nodes that are performing these calculations are miners [20]. There are number of other consensus algorithms being used in other platforms for blockchain technology but the aim of these are same that they need to protect the blockchain network from being compromised and keep it transparent and secure.

The process of mining in blockchain is the way this technology ensures a decentralized secure network of blockchain. As mentioned before, mining is the process of validating transactions on

the blockchain using consensus algorithm to ensure security of peer-to-peer (P2P) network. This process not only validates the transaction but also prevents a user on blockchain from double-spending [21]. The nodes performing this validation are known as miners. Miners perform validation on the transactions and these transactions stored in a block are added on the global blockchain. The process of mining is performed by miners solving difficult mathematical problems. The most commonly used consensus algorithms Proof of Work (PoW), contains the mining process as miners solving a cryptographic algorithm using SHA-256 hash algorithm to find the target hash of the block [22]. When a miner confirms a transaction and adds it to the blockchain the miner receives a reward. This reward could be of two forms i.e. bitcoins or transaction fees for the mining of blocks.

There are various types of consensus algorithms used by vendors of blockchain technology some of these are discussed as follows:

1. **Proof of Work**

   The most common consensus algorithm used by the blockchain technology is Proof of Work (PoW) algorithm and it was first used by Nakamoto [15], in bitcoin network. It is considered as the original algorithm of blockchain technology because it was said to work for blockchain in its true sense.

   The basic working of this algorithm is that there are number of nodes or participants on a blockchain network so when a transaction is requested to be added on the network by any participating node it needs to be calculated. This process is called mining and the nodes that are performing these calculations are miners [20]. This could also be understood that this algorithm is used for securing the transactions and blockchain network electronically [23]. In PoW algorithm as mentioned earlier the miners compete with each other to earn the rewards for adding the transactions on the block. This ensures that miners would not cheat while mining the transactions onto the network because they would not get any reward in case of them cheating.

   The reason due to which the miners would not cheat is because they need to solve a complex mathematical calculation for adding the transaction onto the block. To solve this problem they would also need computational power which explains the fact that this algorithm electronically secures the network. The complex mathematical calculation when solved by the miner would result in the form of *hash* which is actually the target hash of the block. This hash is an important value that must be found by the miners in order to claim the reward for mining.

   Proof of work is such a piece of data which is difficult and is also costly to be solved as it needs a lot of computational power and time to solve it [24]. The main target is for the network participants to accept block in it and that should cover all of the data of the block. To stop from many numbers of blocks to be simultaneously being added on the network the *difficulty* level should be set. This difficulty level would help in limiting the

blocks to be added on the network. Also this would also ensure that it is not able to predict which miner has added the block [24].

The difficulty level should be set of the kind that it is not too easy to solve the mathematical problem and it should also not be too difficult to solve it. If it is too easy it becomes vulnerable to any attacks on the network [25]. And if takes too much time to solve it i.e. it is too difficult the problem the executions would be stuck [25]. Also the hash generated by the miners if not similar should be less than it. The blocks in the blockchain as defined before are connected in a chain and each block has the hash of the previous block in its header.

So, if any attacker wants to change any block on the blockchain by tampering with any block it would require a lot of work as the blocks can only be tampered or changed by performing work on them which is essence of PoW algorithm. This mechanism helps the blockchain technology to be protected from any possible tampering and reduces its vulnerabilities.

This consensus algorithm is used by many crypto-currencies such as Bitcoin, Ethereum etc. These are popular crypto-currencies and usage of PoW in these depict the importance and reliability of this algorithm. The main advantages of this algorithm are as follows,

- It protects the system from any attacks caused by the third party who is trying to add a block to the chain.
- It helps the network from not being spammed by the attacker as the difficulty level is set adequately

This algorithm also has some problems or flaws in it which are mentioned as follows,

- The PoW algorithm consumes too much energy which is mainly used for performing extra calculations. This is the problem or flaw of Bitcoin which was criticized the most and is considered to be an issue with PoW algorithm. This problem basically occurred because miners were making use of more and more of electricity power to make their high processing systems to compute the complex mathematical problem and gather more rewards by adding the block to the main chain.
- An important issue with PoW algorithm is that it does not provide security in smaller networks of nodes. Because if there is a huge network with a number of miner nodes it becomes difficult for an attacker to compromise the system and cause its connected nodes to lose their resources. In a smaller network the possibility of getting an attacker to have an entry point in the system is higher thus it is not a feasible choice for applications having smaller network of nodes as it would not provide security.

2. Proof of Stake

After PoW consensus algorithm this algorithm is second most popular algorithm used by blockchain. This uses the putting something on 'stake' mechanism, in place of the solving the complex mathematical problem method used in PoW algorithm. Also as evident from the name there is something at stake and they would also need some proof for that stake that they have defined for generating a block of the blockchain.

The main reason of this algorithm being used was to avoid or more likely to solve the power consumption problem of PoW algorithm. This was achieved by changing the part of algorithm where the miners are mining the block by consuming energy to the nodes who want to generate the block must hold some stacks for doing it. The nodes that intend to generate the blocks must prove that they own a certain amount of crypto-currency that they are putting on stake for generating that block [23]. The higher the stake set the higher the chance of being the node that will generate the block [23].

This whole process starts when a transaction is sent and a block is to be created for adding it to the blockchain. At that point the nodes who intend to participate in the consensus (previously in PoW known as miners), these nodes would set the crypto-currency they own at stake. The higher the stake is the higher is the chance that the node would get to generate the block and get reward for doing so. In general, the reward in proof of stake algorithm is the transaction fees which is provided to these nodes when they successfully create a block. These nodes in PoS are known as forgers [26].

In PoS the basic requirement needed by a forger to generate the block is to have something they own at stake and this should be higher to all the other forgers connected in the network. They are then randomly selected on the basis of their stakes and assigned the task to generate the block [26]. This method clearly differs from the one used in PoW consensus algorithm where the miners compete with each other to earn the rewards for adding the transactions on the block. That would ensure that miners would not cheat while mining the transactions onto the network because they would not get any reward in case of them cheating.
One of the important aspects of PoS algorithm is to find a feasible and effective way of selecting the forgers for generating a block. For this a number of methods are proposed and are being used by various retailers / crypto-currencies of blockchain technology. Here we would only discuss the two popular selection algorithms [26] used for this purpose.

i. **Coin Age Based**

   As suggested by the name this selection method makes use of the coin age while selecting a forger from the network. In order to complete a block the coins that are put on stake must be older, this ensures their chance of being assigned the task to forge the block. The limit for the coin to be older is 30 days minimum and once a forger has completed the task the coins would again set to 0. This ensures that a single forger would not be able to dominate the whole network.

To calculate the age of these coins being put on stake the

$$Age\ of\ Coin = No.of\ days\ cryptocurrency\ (coins)\ \ is\ held\ as\ stake \times No.of\ coins\ being\ staked$$

This is a secure method of assigning the task to a forger to perform consensus as it prevents the nodes from having the authority over the network resources. This also allows for the blockchain in holding the basic property of decentralization.

ii. Randomized Block

This selection algorithm makes use of random selection of nodes on the basis of the method that looks for both the lowest hash value and their stake size for assigning the next forger of the network. These sizes of stakes are public which means the connected nodes in the network might be able to predict the possibility of a node being assigned the role of forger.

3. Delegated Proof of Stake

As it is understandable the terms and names assigned to the blockchain technology and its dependencies are mostly suggestive of the task it would be performing specifically. Similarly, the Delegated Proof of Stakes (DPoS) makes use of a trusted delegate to perform the proof of stake based consensus on the block. The term 'delegate' refers to an individual who is assigned the duty to do some specific task. This duty is entrusted to this individual due to the trust that it maintains in the network. In the same way, the DPoS uses the voting system to assign this duty to the trusted nodes in the network. They are responsible for block creation in the network but with defined access to them with having no access to the transaction details of the block.

This algorithm functions on the trust between the connected nodes as they would be responsible for some important tasks and resources of the network they are functioning in. So, for these delegates to be involved in the consensus they must be trusted [23] and selected by a proper mechanism to be assigned this important duty. This mechanism is 'voting' which is conducted to select the nodes responsible for this duty or task.

The connected users or nodes are the one who vote to select the trusted users who should be allowed to validate the transactions [27]. These users who are voted to be selected as the trusted users are known as 'witnesses'. Unlike PoS algorithm, this algorithm does not consider that node as the forger who has the larger stake than other nodes. But instead it relies on the voting the node having higher number of votes is assigned the task of forger.

After the voting is completed and the witnesses are selected the next task is to let them validate the transactions. They exist at the top level and are assigned the tasks of transactions validation, blocks creation etc. [27] They have the control to manage the

transactions being added in the block and can also stop a transaction being added to the chain in case the nodes consider that transaction to have any issues. The witnesses do not have the control over viewing the details and contents of the transaction as to ensure the security of the data or any asset being transmitted by that transaction. With this it should also be noted that witnesses once elected are not permanent to stay at top level all the time instead they could be replaced by any other elected witnesses at any time. This also ensures a transparent process which prevents the network from being dominated by specific nodes.

Just like the users of the network can have the right to vote in order to select the witnesses in the same way if any witnesses is responsible for any malicious intent in the network would be exposed to the public and would also be not allowed to be part of the top level anymore i.e. won't have the right to validate transactions and perform other tasks.

4. Practical Byzantine Fault Tolerance

This is another important consensus algorithm used by the blockchain technology. In theory, this algorithm is derived from the known Byzantine's problem where all of the connected parties in a distributed environment must hold a strategy that would be made use of incase of any system failure (catastrophic). The purpose of holding this strategy or defining the strategy to handle such high level of failure is to achieve the desired tasks with the surety that it would be completed and if not completed the risk associated with the system failure might be mitigated in some way [28].

The exact same phenomenon of the Byzantine's problem is made use of in Practical Byzantine Fault Tolerance (pBFT) where the main aim is to have a solution to the problems that would occur in case of malicious nodes being connected in the network [28]. Miguel Castro and Barbara Liskov introduced this algorithm in year 1999 and the aim was to improve the original Byzantine Fault Tolerance (BFT) algorithm.

The basic working of this algorithm is that of all the nodes are connected in an order/sequence and one node is assigned as the leader node with all of the other nodes as the secondary nodes. These secondary nodes would act as backup in case of any possible system failure. These nodes are communicating with each other and it is made sure that transactions being conducted are not malicious and are not tempered during the whole process [28].

The process followed by the pBFT is that the leader node accepts the service requests invoked, this request is multicast to the secondary nodes who are acting as the backup which would ensure that incase of any failure the transactions would be validated. The request is executed and its results are transmitted back. This whole process is known as one view and at end of each view the nodes connected should have the same state [28].

In a mathematical perspective the pBFT can be explained as follows,

$$n = 3f + 1$$ Here f stands for the faulty or byzantine nodes [29]

The pBFT algorithm offers two basic properties of safety and liveness [29], these properties are the reason that would be helpful for the network. The network being safe would allow for the nodes to be atomic in execution of its processes. The liveness would help in ensuring that network would not be stop if the byzantine nodes are not more than the defined.

5. Federated Byzantine Fault Tolerance

This algorithm also makes use of BFT where the nodes are responsible for their own blockchain. Unlike the pBFT consensus, the federated BFT does not predefines its network validators the nodes make this decision by themselves. It is also known as Federated Byzantine Agreement (FBA). The reason for this shift towards federated BFT was that pBFT did not provide the scalability that was needed for blockchain.

The nodes needed for performing consensus using this algorithm are free to join the network and do not require a membership or permission to join the network [30]. These nodes can join and leave the system whenever needed so it has a permissionless system that it follows for these nodes. The federated BFT makes use of quorum and quorum slices for these nodes participating in the network. These would be responsible for reaching on an agreement or consensus in the network. Let us now understand the quorum and quorum slices,

Quorum: In general terms, quorum is the minimum set of individuals or members who must be present at a meeting for making its decisions or proceedings to be valid. In terms of this consensus algorithm these are the set of the nodes who are existing in a network and they must all agree together to reach on an agreement or consensus [30]. The nodes in order to reach upon an agreement must be communicating with each other. This communication is also done on the basis of trust that exists between these nodes. For this agreement to be reached upon the maximum number of nodes must agree for it to be valid.

Quorum Slices: These are subsets of quorum that are also used for helping or assisting the other specific nodes in reaching an agreement [30]. These quorum slices make up the network and are responsible for management of other nodes in the network. In short, quorum slice can make other nodes to agree and is also a subset of quorum.

An individual node connected in the network is responsible for selecting their quorum slices by themselves. This ensures decentralization in the network and also the decisions following this mechanism becomes federated. Another important concept apart from

quorum and quorum slices is that of 'voting'. The voting being conducted in it has three states associated with the messages that are communicated with it which are [30]:

- Unknown
- Accepted
- Confirmed

These are the messages that the nodes communicate to each other when communicating with each other. These are the agreement states that must be met for making it easier for the nodes to understand that whether an agreement has been reached upon or not. Also the voting process has 4 basic stages which are explained briefly as follows [30]:

- *Initial Voting* contains the preliminary votes that are given to the nodes and statements that they consider to be valid. The votes that the nodes give in this stage are also dependent on the other trusted nodes, if that other node votes against or for any statement the node in discussion might also do the same.
- *Acceptance* is the stage where the nodes accept the statement on the basis of the trusted other participating nodes in a quorum slices are accepting the statements or not.
- *Ratification* is when the quorum nodes accept a statement before this stage the individual nodes were involved in the voting process but now the quorum nodes are the primary actor of this stage of federated voting. Ratification is basically the act of accepting any agreement in an official manner and also making that agreement to be valid.
- *Confirmation* is the last and an important stage in federated voting process as the result of voting process would be communicated across the connected nodes. So, when a specific defined number of nodes agree on a statement then they reach upon an agreement and that is communicated across the network to all the connected nodes. This communication is again done in the form of messages that are cryptographically secured for helping nodes communicate with each other in a easy manner. On the basis of these messages the nodes can change their voting decisions as well by following the voting choices of the other nodes. But this stage is the final and last stage of voting process so it holds its due importance.

Table 4: Federated BFT Voting Stages

Stages	Description
Initial Voting	Initial stages where the voting is yet not final and nodes are only giving votes for valid statements and also the factor of following the choice of the vote of trusted nodes is involved here as well.
Acceptance	The statement is accepted in this stage as the nodes vote and when the threshold of quorum gets

	completed the acceptance of the statement is done as well.
Ratification	The official agreement on the statement by the quorum slices is depicted in this stage.
Confirmation	When the agreement is made valid by all of the concerned nodes reaching upon an agreement and the threshold is also fulfilled then the confirmation stage commences.

6. Delegated Byzantine Fault Tolerance

The delegated byzantine fault tolerance (dBFT) works in a method that is similar to the way a country works where there are leaders who govern the whole system with the help of delegates who keep its very parts functional. The mechanism to achieve consensus in this algorithm is that of state machine [29] where there are different states assigned at different point of time in the ongoing process. The transitions of states are dependent on the round-robin method [29] where the nodes change their states in a round robin way.

As mentioned earlier the state machine mechanism is followed by the dBFT algorithm so the state machines contain states that get updated whenever a process is initiated or updated. These states mainly include the start state, processing states, and end state. But these states are followed or present in general systems where the state machines are used. In dBFT these states are named in a different way, they might still perform the same task but are known differently in dBFT algorithm. The following are some of those main states included in dBFT [29],

- *Initial* is the first state of this consensus algorithm and is the initial state of machine. At this state no work is being done the machine would be in its initial phase with no processing started.
- *Primary* is another state of dBFT algorithm that is depending on the height of the block along with its view number. All of the states have an input and they always generate an output to that input after performing the specified processing on the input. In this case the height of the block and its view number would be output of this state.
- *Backup* state has the output as true if the node is primary and it would be false otherwise.

Table 5: Consensus Algorithms and platforms

Algorithm	Applications/Platforms
Proof of Work	Bitcoin, Ethereum
Proof of Stake	Ethereum
Practical Byzantine Fault Tolerance	Zilliqa, Hyperledger
Federated Byzantine Fault Tolerance	Stellar
Delegated Byzantine Fault Tolerance	NEO

2.1.6. Types of Blockchain

This technology also offers different kinds of blockchains which could be used by the developers for developing the application that could benefit the society. These different types of blockchains have different applications and advantages provided by them. They are explained as follows:

2.1.6.1. Public Blockchain

As its name suggests this blockchain is public which means that it could be accessed by anyone existing on the blockchain network. By accessing here it means that anyone on it can send transactions and receive transactions from this kind of blockchain [31]. Also the data stored on it would be visible to anyone connected to it from any part of the world. Moreover the consensus on this kind of blockchain is also using these publically connected users to have a right of decision making. These blockchains do not require for any permission from the already connected nodes in the network anyone can connect them without any issue and can perform their desired task on them. So following are the main points of these types of blockchains [23],

- Anyone can connect to this blockchain and send transactions on it, and would also expect these transactions to be added on the block (if they are valid).
- Anyone can participate in the consensus while validating transactions and adding them on to the chain.
- The transactions being performed on the blockchain would be visible to everyone without the need of any permission.

2.1.6.2.Federated Blockchain

Unlike the public blockchains these operate under an authoritative group of nodes or in simple terms under a federation. This federation would not allow any other entity to join this blockchain. These types of blockchain are also referred to as *Consortium Blockchain* as they select a number of nodes which are responsible for performing and managing the consensus process being done in the blockchain network [31]. These blockchains could be customized to allow a node to have access to read the transactions being done on the blockchain network or to make these transactions to be public i.e. visible by everyone. One of the greatest advantages of

these types of blockchains is providing privacy to the transactions being conducted on it. This was not possible with the public blockchains because they would never be private and were accessible by everyone.

2.1.6.3.Private Blockchain

As understandable by its name these blockchains are fully private with no access provided to the any party existing outside of the network. The permissions to write in these types of blockchains is central to one party and the read permission is also assigned to the authorized parties existing on the blockchain network [23]. Their greatest advantage is considered to be the security provided by the tight permission-based model it uses in its network. The transactions being done are secure and are not visible or accessible to any third party. They are mostly desirable for use in small companies which do not need their data to exist outside their company's network.

2.1.7. Key Features of Blockchain

The key features or the benefits offered by blockchain are discussed in this section. They are as follows:

Decentralization: With blockchain the information is distributed across the network rather than at one central point. This also makes the control of information to be distributed and handled by consensus reached upon by shared input from the nodes connected on the network. The data which was before concentrated at one control point is now handled by many trusted entities.

Data transparency and confidentiality: Achieving data transparency in any technology is to have a trust based relationship between entities. The data or record at stake should be secured and temper proof. Any data being stored on the blockchain is not concentrated at one place and is not controlled by one node but is instead distributed across the network. The ownership of data is now shared and this makes it to be transparent and confidentially secure from any third party intervention.

Security and Privacy: Blockchain technology uses cryptographic functions to provide security to the nodes connected on its network. It uses SHA-256 cryptographic algorithm on the hashes that are stored on the blocks. SHA stands for *Secure Hashing Algorithm,* these hashes provide security to the blockchain as data integrity is ensured by them. A hash can be generated by any data being stored digitally but data cannot be extracted from that hash [32]. This makes blockchains to be secure. And as the blockchain technology is decentralized and secured by the cryptographic approaches this makes it to be a good option for privacy protection of certain applications.

2.1.8. Challenges Faced by Blockchain Technology

Scalability and storage capacity: Storing data on the blockchain creates two main problems i.e. confidentiality and scalability [33]. The data on the blockchain is visible to everyone that is

present on the chain this makes the data vulnerable and is not a desired outcome for a decentralized platform. As, the purpose behind using blockchain was to get data security that was somewhat compromised in EHR systems. The healthcare data that could be stored on the blockchain would contain patient records as data on the block and would also include current and previous block's hash. The data stored can contain the patient medical history, records, lab results, X-rays reports, MRI results and many other related results and reports, all of this voluminous data is to be stored on the blockchain that would highly affect the storage capacity of blockchain[34]. Because blockchain can store data on it but its protocol was not designed for this purpose.

Lack of social skills: The blockchain technology is understandable by very few people as they don't know how it works and it also faces these issues as this technology is still in the phase of evolving. Also, the shift from trusted EHR systems to the blockchain technology would take time as hospitals, or any other healthcare institutes need to completely shift their systems to blockchain.

Lack of universally defined standards: As this technology is still in the initial phases and is constantly evolving so there is no defined standard for it. Due to this the implementation of this technology in healthcare sector would also take some time and effort. As it would require certified standards from international authorities that overlook the standardization process of any technology [35]. These universal standards would benefit in deciding upon the data size, data format and type of data that could be stored on the blockchain. Also, the adaptation of this technology would become easier due to the defined standards, as they could be easily enforced in the organizations.

Table 6: Benefits and Barriers of Blockchain Technology

Benefits and Barriers of Blockchain Technology		
Benefits	Decentralized	The data or information stored on the blockchain is distributed across the network
	Data Transparency, Confidentiality	Data stored on the blockchain is temper-proof and has shared ownership of that data
	Security and Privacy	Blockchain uses SHA-256 cryptography algorithm to secure any information being stored on it
Barriers	Scalability, Storage Capacity	Storing huge volumes of data on the blockchain would cause storage and scalability problems
	Lack of social skills	As blockchain is an evolving technology and it is not a well understood technology so, it is quite challenging to shift the previously used systems on this technology
	Lack of universally defined standards	There are no defined standards and principles for blockchain technology that are universally applied which makes it difficult to enforce it throughout a specific domain

2.1.9. Solutions to the Challenges Faced by the Blockchain Technology

The solution to one of the most commonly faced challenge i.e. *scalability* and *data storage* is discussed in this section. As scalability is concerned storage of data on-chain is not a preferred solution as it would not only affect the confidentiality of data but also increases data size on the chain. So, the solution to this problem could be store the data off the blockchain. Also, the stored data should be encrypted to ensure security of data. One such solution was provided by Sahoo and Baruah [36], they proposed storing the data on a database system and using it as an off-chain storage system. They proposed to use the scalability provided by the underlying Hadoop database along with the decentralization provided by the blockchain technology. They used the method to store blocks on the Hadoop database, the blockchain on top this framework include all of the needed dependencies of blockchain but the blocks are stored on Hadoop database to improve scalability of the blockchain technology. It was a good solution but the problem with it was again that data was being stored on a centralized storage which kills the purpose of blockchain i.e. decentralization. So, we needed such a platform that would store data in a decentralized manner off-chain. One such platform is InterPlanatery File System (IPFS); it is a protocol that uses peer-to-peer solution of storing data [37].

IPFS protocol uses a cryptographic mechanism to protect the data stored on it from being tempered or altered. The data that is stored on IPFS enjoys following benefits:

- Assigning unique cryptographic hash to the stored files
- File duplication not allowed
- Storage of files by decentralized naming system i.e. IPNS

These benefits make it a feasible solution for data storage and scalability issue faced by the blockchain technology.

Chapter 3

LITERATURE REVIEW

Blockchain technology was designed by Satoshi Nakamoto [15] , the basic idea was to have a cryptographically secured and a decentralized currency that would be helpful for financial transactions. Eventually, this idea of blockchain was being used in various other fields of life; healthcare sector also being one of them intends to use it.

As discussed earlier, healthcare sector is constantly evolving and is using various technologies to improve it and make its resources more secure and cost effective. Also, as reported by IBM in year 2017 [38], a total of 16 percent healthcare sectors aim to have a commercial release of blockchain solution. As of year 2018, it is reported that 90% healthcare companies are planning to launch pilot blockchain healthcare related projects [39]. This shift of healthcare sector towards this novel technology is due to the many benefits that it offers, the most important of it being the data exchange option. Also, previously the data being stored was concentrated at a central medium serving as the control location of that data, any attack on that central medium makes the data to be not secure. Using blockchain here provides the benefit of storing that data without any central medium in a distributed manner, and now the data is not stored at one place but rather at many secure locations making it safe.

A number of researchers have carried out the research on this area, these research works focus on the fact that whether the idea of using blockchain for healthcare sector is feasible or not. They also identify the advantages, threats, problems or challenges associated by the usage of this technology. Some researchers also discussed the challenges that would be faced while actually implementing this on a larger scale. One of those issues was identified to be the data storage problem on blockchain. As considering the healthcare sector, data is huge and it is constantly growing so if a company or an organization selects to shift its records on the blockchain, the main problem that it would face will be data storage of those records on the blockchain. As, storing data on the blockchain costs the user some digital currency it differs from platform to platform but it follows the universal principle that storing a large file would cost a larger amount of digital currency. In terms of blockchain, storing a piece of information (data), leads to a transaction being done on the blockchain and this transaction has some cost associated with it and it varies with complexity of transactions [40].

3.1. Theoretical / Analytical Blockchain-Based Research

Gordon and Catalini [10], conducted a study that focused on the methods by which blockchain technology would facilitate the healthcare sector. They identified, that healthcare sector is controlled by hospitals, pharmaceutical companies and other involved third parties. The patients are not given the power to have access to their medical records. They also specified that key

reason why blockchains should be used in healthcare is because of its data sharing property. There are a number of trusted parties that have to agree about the data sharing to define the data sharing rules. They also identified four factors or approaches due to which healthcare sector needs to transform for usage of blockchain technology. These include way for dealing of digital access rights, data availability, and faster access to clinical records and patient identity.

This study also focuses that the entity having ownership of data would define the permission and access rules for the data being stored on the blockchains. And these records being stored on the blockchain would have to be accessible at one point, so it is accessible to respected parties when needed. Here, there was a need to define that whether the data should be stored on-chain or off-chain, if it is on-chain it is easier for parties to interchange data. For later one, there needs to be some other mechanisms to have access to data. Another important identified factor was patient identity as blockchain uses keys to identify users on the chain or more specifically public keys, which act as the medium used for identification of a user on the blockchain platform.

Along with the helpful features of blockchain the authors also included the challenges faced by this novel technology. The identified challenges or barriers were huge volume of clinical records, security and privacy, patient engagement and the incentives by which healthcare sector would adopt this novel technology i.e. blockchain. The identification of these factors would be helpful, as now we know the potential challenges posed by blockchains and it also directs that how we can solve these problems in near future to create a secure and fully-functional system for healthcare sector on blockchain.

Eberhardt et al. [41], conducted a study to understand possible approaches to solve the scalability problem of blockchain and also to find such projects that intend to solve this problem. They define blockchain as composition of various computational and economical concepts based on peer-to-peer system. The aim of this study was to find which data should be stored on-chain and what could be stored off-chain. On-chain data is any data that is stored on the blockchain by performing transactions on it. Off-chain data storage is to place data elsewhere on any other storage medium but not on-chain and it also would not include any transactions.

They presented five patterns for off-chain storage of data and also include the basic ideas and implementation framework of these patterns. First pattern discusses to compute state transitions off-chain as checking of final state on-chain is expensive. Using this pattern smart contracts now act as the state machines, it decreases the computational cost of state transitions. Second pattern discusses how can two participants, existing on the same network and are intending to perform transactions later on can reduce cost of these transactions. To do so these two participating entities create a smart contract containing a function that includes a signature to ensure that both entities agree on any state transitions. The smart contract is deployed but the transactions are performed off-chain to save computational costs. Third pattern discusses to store data off-chain on a storage system and smart contract only contains the reference to that off-chain stored data. To access the data now the user only needs to have access to that reference stored on-chain.

Fourth pattern includes third party to perform complex computations, i.e. it includes the outsourcing of computation off-chain. Last pattern includes the way to design contracts to reduce the cost of transactions being done on-chain. These above mentioned patterns could be helpful in devising a framework that solves the scalability problem of blockchain platform.

Vujičić et al. [42], presented a paper that introduces blockchain technology, bitcoin and Ethereum. The authors define that information technology landscape is constantly changing and blockchain technology is benefiting the information systems. They first explained that bitcoin is a peer-to-peer distributed network used for performing bitcoin transactions. They also defined that proof-of-work consensus algorithm along with the mining of blockchain concept. The authors emphasize on the fact that scalability is a severe problem faced by blockchain. They also identify that certain solutions are proposed for solution of scalability problem these include SegWit and Lightning, Bitcoin Cash and Bitcoin Gold. These solutions were classified as soft fork and hard fork.

The next section of the paper explained the Ethereum and its dependencies. Ethereum was defined as the system that could be used for representation of blockchain that is built upon the Turing complete language of programming. They also explain that Ethereum can be used for empowering its users to define their own access and ownership rules in the applications they create. The paper explains the Ethereum in great detail as it contains the explanation of Ethereum accounts, transactions, messages, and tokens. In general the complete Ethereum understanding was provided to the readers.

The authors differentiate Ethereum blockchain from bitcoins' blockchain. As, Ethereum contain the transaction details along with the block number, difficulty level and nonce in its header. While bitcoin blockchain do not store the transaction details such as list of transaction and history in its header. They further explained that Ethereum uses Keccak 256-bit hashing algorithm to store the header details of the previous block. Along with the early stages of digital crypto-currency authors defined how blockchain technology has evolved into a technology that serves its purpose in many fields of life. They also identified that scalability is an issue faced by many implementations of blockchain it has certain solutions but it still needs more attention and frameworks to eliminate it further.

Wang et al. [43], conducted a study that focused on smart contracts and its application in blockchain technology. They first introduce the smart contracts, their working framework, operating systems and other important concepts attached with them. The authors also discuss that how could smart contracts be used for the new concept of parallel blockchains. They identify that reason of using smart contracts in blockchain is due to the decentralization that is offered through the programming language code written in them. Moreover, bitcoin technology can only be considered to act as a prototype of smart contract due to its limited functionality. The authors emphasize that actual usage of smart contract can be viewed in the Ethereum technology which they explain throughout the paper.

After introducing the basics of smart contract the author explained the various layers of blockchain that combine together to keep system functioning. These layers are data, network, consensus, incentive, contract, and application layer. Smart contract resides in the *contract layer* along with algorithm and code. The paper further explores the way a smart contract is operated on the blockchain, a party signing the contract needs to take care of its details, conditions and liabilities that are applicable in case of contract breaches. From an external perspective it might look like a business contract that follows these rules but these contracts are deployed on the blockchain. They are decentralized and trusted piece of code that is to be deployed on the blockchain for its functions to work. The authors explained that functioning of a smart contract as that when all the trusted nodes on the network sign the smart contract they are then deployed onto the blockchain for further operations. They contain a set of code that defines the functionality they intend to fulfill on the blockchain.

The paper not only discusses the architecture and framework followed by smart contracts but it also gives an insight on its applications and challenges. Authors identify that smart contracts can be used in financial applications, prediction systems and internet of things (IoT) mainly. Moreover, as mentioned earlier it also contains the challenges or barriers faced by smart contract these are recursive calls to the smart contract functions, lack of availability of trustworthy information needed for smart contract and privacy concerns. In the last section, authors define some recent advancement in usage of smart contract and future trends. One important future trend was identified to be the concept of parallel blockchain that intends to create such blockchain that can optimize two different but important modules. The authors define that aim of this paper was to act as a guide for future research in domain of smart contract. As smart contracts is expected to be used widely in various applications of blockchain technology.

Kuo et al. [44], conducted a review that discussed several applications of blockchains in biomedical and healthcare sector. The authors identified that using blockchains for this domain offers many advantages and some of these are decentralization, persistence of clinical or medical records, data pedigree, and continuous accessibility to data and lastly secure information being accessible to biomedical or healthcare stakeholders.

After discussing the benefits blockchain offers to biomedical or healthcare sector they then discussed the applications of blockchain in this sector. The first identified application was managing the medical records, as now the system would be decentralized and records would be intact due to the persistence feature of blockchain so it would be easier and secure to manage the record. The second application was related to insurance claims, as blockchain also offers the benefits of having a sole owner of the records i.e. data provenance or data pedigree. Also the better and secure accessibility to records will also help to enhance the insurance claims. Other applications included the acceleration in research work being conducted in these domains and blockchain serving as a data ledger for biomedical or healthcare sector.

They authors also discussed the limitations associated with blockchain these included confidentiality, speed, scalability and threat of malicious attack i.e. 51% attack. The authors identified these limitations to be critical for healthcare or biomedical sector as they are being used to store sensitive medical or clinical records. Lastly, the authors presented some solutions to mitigate these limitations faced by blockchain technology. These included storage of records using encryption mechanism to solve confidentiality problem. Storage of only metadata on the chain while storing any sensitive medical record off-chain on any other trusted storage medium to solve the scalability and speed problem. Using, VPNs (Virtual Private Networks) to enforce HIPAA-compliance rules to ensure safety from malicious attack. This review could be helpful to advance the blockchain technology in various domains of biomedical or healthcare by using the identified applications discussed in the review.

3.2. Prototype / Implementation Blockchain-Based Research

Sahoo and Baruah [36], proposed a scalable framework of blockchain using Hadoop database. In order to solve the scalability problem of blockchain, they proposed to use the scalability provided by the underlying Hadoop database along with the decentralization provided by the blockchain technology. They used the method to store blocks on the Hadoop database, the blockchain on top this framework include all of the needed dependencies of blockchain but the blocks are stored on Hadoop database to improve scalability of the blockchain technology.

This study was not focused on the healthcare sector but still it was focused to solve the scalability problems faced by the blockchain platform. As mentioned earlier, the architecture proposed in this study was based on Hadoop database system, particularly Apache Hadoop Databases system. It is known as a big data store that provides scalability and is also functioning on a distributed system. The proposed system in this study was to use nodes and these nodes connected in a federation form a blockchain. Each node connected in this federation, have same privileges accessible that makes this whole architecture to be decentralized. These federation nodes are responsible for managing the blockchain; also any transactions on the blockchains are also to be managed by them. To ensure the security of the blockchain, the architecture also includes a validation system in which any transaction occurring on the blockchain needs to be validated by a list of voters that in this case are the federation nodes that check basic elements of that transaction. These elements include the hash of the current transaction, signature and any inputs in the transaction. Presence of these elements in the transaction would indicate to the federation nodes to include the block on the blockchain.

To tackle the scalability problem of blockchain platform this study offers to use Hadoop database system, along with SHA3-256 for hashing used for transactions and blocks. The programming language used for this architecture was Java. This study, was helpful in understanding that blockchain can be used with other platforms that are scalable to improve or solve the scalability of this platform.

Zhang et al. [45], proposed a scalable solution to the blockchain for clinical records. The basic aim of this study was to design such an architecture that complies with the Office of National Coordinator for Health Information Technology (ONC) requirements. They identified the barriers that this technology faces mainly include concerns related to privacy and security of blockchain, healthcare entities not sharing any trust relationship among them, scalability problems related to huge volume of datasets being transmitted on this platform, and lastly there is no universal standard enforced for data being exchanged on blockchain.

They also proposed possible solution for these barriers; firstly they proposed to comply with the technical requirements to ensure interoperability for healthcare –based blockchain system. The authors explained that the reason behind inclusion of this objective in their study is to ensure a stable system for improvement of clinical decision making. The second objective for this study was to design an FHIR chain architecture based on blockchain and this architecture was made while keeping in consideration the ONC technical requirements. This study also include a demonstration of a decentralized application (DAPP) based on the design formulated on the ONC requirements as mentioned before. Lastly, they included the lessons learnt and how can FHIR chain be improved.

The benefits that this architecture presents according to the authors include is modularization of data or clinical records being stored on the blockchain, data integrity with scalability, faster and granular access controls and enhancing trust among associated entities. This study, would be helpful on many levels as it not only provides an in-depth understanding of the blockchain platform but also the challenges, barriers and limitations faced by the blockchain platform in the conventional healthcare sector.

Jiang et al. [46], also proposed a framework for exchange of healthcare information using blockchain technology. They firstly, examined the requirements and reasons of implementing healthcare information on this technology. The architecture of this paper includes two kinds of healthcare information to be stored i.e. electronic medical records (EMR) and personal healthcare data (PHD). The authors name these as EMR-chain and PHD –chain, these two were coupled loosely to handle the different kind of information they hold. The authors identified that the aim of this study was for improvement of throughput and fairness factors of the system.

The study also includes mechanism to fulfill the need of privacy and authentication factor in the healthcare information being exchanged using blockchain technology. They also proposed to use Fair-first and Tp & Fair algorithms that are fairness-based algorithms used for transaction packing. The authors described that the reason behind using these algorithms is for the improvement of throughput and fairness of the system. The paper describes the system's architecture to be consisted of three main components i.e. blockchain network, medical institutions, and individuals storing their personal healthcare information. The identified stakeholders of this system were the medical institutions and individuals, where medical institutions are storing information on EMR-chain and individuals on PHD-chain.

The system design of EMR-chain consists of off-chain and on-chain mechanism. The mechanism as defined by the authors, include storage of information off-chain and its verification being done on the chain. The storage was being done on the databases system deployed at the hospitals and for verification the medical records were allotted a hash, that hash was used for on-chain verification. For the PHD-chain, the authors proposed to use fairness-based algorithms specific for transactions packing. The algorithms used were Fair-First and Tp & Fair, the first one for allowing the nodes to work for same transaction thus using it for EMR-chain while later for allowing nodes to work on different transactions thus using it for PHD-chain. The authors named this whole architecture as BlocHIE; this was a very thorough framework that was intended to tackle the issues related to privacy, authentication, throughput and fairness of the system.

Kim et al. [47], proposed a system for management of medical questionnaires and the aim of this system is data sharing through blockchain technology. The authors explain that reason behind the selection of data storage and sharing of medical questionnaire is to use this data for further medical and clinical research purposes. They emphasize that this could also be helpful to researchers for developing diagnosis system for patients. The reason behind using blockchain technology in this system was due to the fact that hospitals were previously storing this information in EMR systems. With each hospital having its own EMR system there were issues of conflict between the terminologies being used in them due to no defined standard. Moreover, the security issues associated with these systems was also a reason due to which authors selected blockchain technology for their proposed framework.

The proposed framework has two main functions they are to create, store the data gathered by questionnaires and to share that data. Another benefit that this system proposes is the validation of the questionnaire being submitted in the system. The questionnaires that are added on this system are first validated to be correct specified format and then are parsed to differentiate the personal data and specific data related to questionnaire results. This would ensure that data could be shared for future research purposes. The authors also address the scenario when a third party requests to access this questionnaire data, this would need the patients' permission that is asked by the doctor to let third party view that data.

The authors also explore the fact how can this data be made secure from third party intervention. They explained that EMR systems need to take certain security measures to make them secure but their system provide patient the authority to control the rules that who can view their data. As previously, the data was being secure but it was also not being accessible to patients and also they were not being asked about their consent to use that data. The authors stress over the fact that their proposed framework would give patients the right to control their information.

Table 7: Aim and Contributions of Literature Review

References	Description
[10]	*Aim:* Identification of the reasons why healthcare should make use of blockchain technology and how it would facilitate the healthcare sector *Contribution:* The authors presented certain factors which would be helpful for adaptation of blockchain technology in healthcare sector. Also some potential challenges that it might face are also discussed in the study. This study also directs that how we can solve these problems in near future to create a secure and fully-functional system for healthcare sector on blockchain.
[41]	*Aim:* To find the solution to the prevalent scalability problem in blockchain technology. Also the identification of such projects that are meant to solve this problem. And lastly to identify that to solve scalability problem through off-chain and on-chain storage what should be stored off and on-chain. *Contribution:* The authors presented five patterns that would be helpful in devising solutions to the scalability problem. These patterns were different from one another and were all holding certain benefits for blockchain technology. The identification of these patterns could be helpful in devising a framework that solves the scalability problem of blockchain platform.
[42]	*Aim:* Explanation of blockchain technology in finer details and also introduced the Ethereum and its dependencies to give an insight to the reader of how this technology actually operates. *Contribution:* The author differentiated the Bitcoin and Ethereum, which included their architecture and various elements holding the mechanism of these technologies. They also identified scalability as a problem faced by blockchain technology and offered certain solutions for this problem.
[43]	*Aim:* The focus of the study was to introduce the smart contracts used in Ethereum and in blockchain technology. This paper contains a detailed introduction of smart contract, its applications and the functionalities offered by it in blockchain. The aim of this work was to act as a guide for future research in domain of smart contract. *Contribution:* The paper discusses the architecture of smart contracts, its framework, challenges and applications in various domains. It also explains various layers in the smart contracts that make up together to keep them functioning. It would helpful in understanding the smart contract in detail as this technology is expected to be used widely in various applications of blockchain technology.
[44]	*Aim:* This was a systematic review and the aim was to discuss several applications of blockchain in biomedical and healthcare sector. *Contribution:* The author identifies and discusses the benefits offered by the blockchain in biomedical and healthcare sector along with the

	limitations associated with it. This review could be helpful to advance the blockchain technology in various domains of biomedical or healthcare by using the identified applications discussed in the review.
[36]	*Aim:* To solve the scalability problem of blockchain by proposing a framework functioned on the Hadoop database. *Contribution:* They proposed a scalable solution to blockchain by making use of Hadoop database. To tackle the scalability problem of blockchain platform this study offers to use Hadoop database system, along with SHA3-256 for hashing used for transactions and blocks. The programming language used for this architecture was Java. This study, was helpful in understanding that blockchain can be used with other platforms that are scalable to improve or solve the scalability of this platform.
[45]	*Aim:* To solve the scalability problem of blockchain in compliance with the Office of National Coordinator for Health Information Technology (ONC) requirements. *Contribution:* This study intended to solve the problem of scalability being in compliance with ONC requirements. They also identified barriers faced by the blockchain technology and the solutions to them as well. This study would be helpful on many levels as it not only provides an in-depth understanding of the blockchain platform but also the challenges, barriers and limitations faced by the blockchain platform in the conventional healthcare sector.
[46]	*Aim:* The aim of this study was to propose a framework for exchange of healthcare information using blockchain. This study considered the need of privacy and authentication in the exchange of healthcare information using blockchain technology. *Contribution:* The authors named this whole architecture as BlocHIE; this was a very thorough framework that was intended to tackle the issues related to privacy, authentication, throughput and fairness of the system. They designed two different chains named as EMR and PHD chains which were of different nature as one was for medical institutions and the other was for individual storing their personal healthcare information.
[47]	*Aim:* This study proposed a system for management of medical questionnaires and the aim of this system is data sharing through blockchain technology. *Contribution:* The proposed framework has two main functions they are to create, store the data gathered by questionnaires and to share that data. Another benefit that this system proposes is the validation of the questionnaire being submitted in the system. The authors also explore the fact how can this data be made secure from third party intervention. They explained that EMR systems need to take certain security measures to make them secure but their system provide patient the authority to control the rules that who can view their data.

3.3. Comparison of Proposed Framework with Related Work

The following table 3 compares our proposed framework benefits and features with that of the related work [36] [45] [47]**.** The above defined features offered by our proposed framework are blockchain-based, scalability; usability and integrity are included in this comparison. These features are then compared and observed that whether they exist in the related work under consideration or not.

The first work which is compared is of Sahoo and Baruah [36], they proposed a scalable framework of blockchain using Hadoop database. The main aim of the study was to solve the scalability problem of blockchain, they proposed to use the scalability provided by the underlying Hadoop database along with the decentralization provided by the blockchain technology. The study was blockchain-based but the system or platform being used to solve this problem was not decentralized. The authors presented the framework to store blocks on the Hadoop database which was functioning on top of this framework. The rest of the system was working just like a blockchain-based application but the blocks were stored on the Hadoop database.

The second work compared is of Zhang et al. [45], proposed a scalable solution to the blockchain for clinical records. The basic aim of this study was to design such an architecture that complies with the Office of National Coordinator for Health Information Technology (ONC) requirements. They identified the barriers that this technology faces mainly include concerns related to privacy and security of blockchain, healthcare entities not sharing any trust relationship among them, scalability problems related to huge volume of datasets being transmitted on this platform, and lastly there is no universal standard enforced for data being exchanged on blockchain.

The third work compared is of Kim et al. [47], also proposed a framework for exchange of healthcare information using blockchain technology. They firstly, examined the requirements and reasons of implementing healthcare information on this technology. The architecture of this paper includes two kinds of healthcare information to be stored i.e. electronic medical records (EMR) and personal healthcare data (PHD). The authors name these as EMR-chain and PHD – chain, these two were coupled loosely to handle the different kind of information they hold. The authors identified that the aim of this study was for improvement of throughput and fairness factors of the system.

Our proposed framework uses the off-chain mechanism of IPFS protocol to resolve the issue of scalability faced by the blockchain technology. The three works does not pose a feature of usability. Moreover, our proposed framework also makes sure that blockchain features are make use of while creating the system. The features offered by this technology such as data integrity, confidentiality, transparency, decentralization, security, privacy and temper-proof metadata storage on the blockchain.

Table 8: Comparison with related work

References	Blockchain-Based	Scalability	Usability	Integrity
[36]	Y	Y	N	Y
[45]	Y	Y	N	Y
[47]	Y	N	N	Y
Our proposed system	Y	Y	Y	Y

Chapter 4

SYSTEM DESIGN & ARCHITECTURE

In this section, we describe the framework that we propose for implementing healthcare information on blockchain.

4.1. Use Case

Before understanding the system design and basic architecture of the proposed framework let us first define a use case that would be helpful in understanding it better. The figure 2 depicts the scenario being discussed below.

The healthcare or medical information is the most important and crucial data for patients in a care provider organization. This data keeps on increasing as patients visit the hospital for consultations. That data should be readily available whenever it is needed. Let us, consider the scenario where a patient visits hospital to get consultation for any ailment. When he first arrives at the hospital, the nursing staff or concerned individual creates a record for the patient. That record is updated when the doctor suggests the patient to get some lab tests done. Patient then visits lab for tests and his records gets updated when he consults doctor again after getting lab test results. The doctor gives some diagnosis that is again stored in the record of the hospital.

At this point, the record of patient is located at only one point but the problem arises when patient visits another hospital. At that hospital again a new record of patient is created that is updated if patient visits again, so now there is more than one medical record for patient. Moreover, if patient's medical history is needed and it is not stored in the current hospital he is getting his treatment in, then we need to transfer data from the previous hospital that patient visited. That is again a long process as first of all the patient or his legal representative would have to sign a consent form to allow the hospital to transfer the medical records to another hospital. This whole process is long and it could take time for data to be transferred to another location. In today's world this system is not convenient as data should be readily available whenever patient needs it and with the defined approach it is difficult for patient to have access to his records and for care providers to have complete medical history or record of patient available at one secure point.

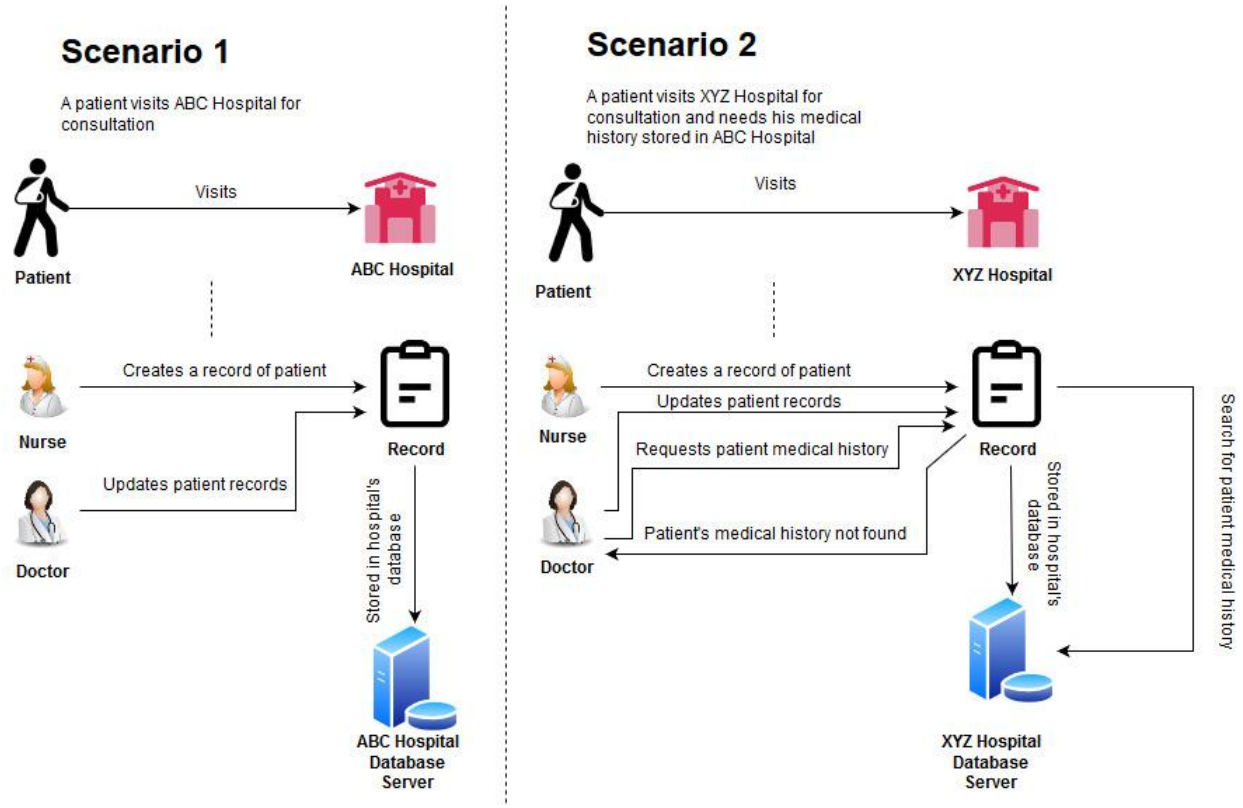

Figure 3: Simple Scenario of a patient visiting a hospital

By using the proposed framework of this research endeavor we can solve this scenario as it would provide medical records without any need of consent form but in a secure way. Using blockchain-based solution, the data would be stored in a decentralized storage system and it would be available to the concerned entities when needed without any third party having complete control on that data.

4.2. Preliminaries

In this section, we formally describe the preliminaries used in proposed framework. The software platform used to develop the framework. Also the advantages of using these and how they benefit our framework.

4.2.1. Ethereum

Just like the popular crypto currency Bitcoin [15], Ethereum is a distributed blockchain network. It was formally introduced in year 2015, by the founders Vitalik Buterin, Gavin Wood and Jeffrey Wilcke who began their work on this ground breaking technology in year 2014 [48]. The idea behind Ethereum was to create a trustless smart contract platform that would be open-source and would also hold the feature of programmable blockchain. This technology also shares the

peer-to-peer networking that makes it distributed. This platform also makes use of its own crypto currency known as *Ethers* [48] [49]. This crypto currency can be used for sharing it between accounts connected on Ethereum blockchain [50]. Ethereum also provides the programmers a language in which they can customize their own blockchain, this language is known as *Solidity*. It was developed for smart contracts that are the main feature of Ethereum.

The Ethereum blockchain comprises of following components:

- Accounts
- Transaction
- Gas
- Blocks
- EVM
- Smart Contracts
- Consensus Algorithm
- Merkle Patricia Trie

4.2.2. Accounts

The interaction between various users existing on the Ethereum blockchain is actually between accounts existing on it. Every account on the Ethereum has a public address and private key associated with it. The public address is visible to everyone and is actually used for identification of a user on the blockchain; the public address is 20 bytes address.

Ethereum has two types of accounts,

- ***Contract Accounts:*** These accounts are governed by the code existing inside them. The contract code when gets executed it performs the operations and this execution is triggered by internal transactions and message calls. It contains an ether balance and smart contract code inside it.
- ***Externally Owned Accounts:*** This account is used for sending transactions from one account to another account. This account also holds an ether balance but it does not have a contract code residing inside it. The control is of private keys in externally owned accounts (EOA).

The working of Ethereum blockchain by using these accounts can be understood as that this scenario starts when a transaction is fired an EOA. This transaction is received by contract account which contains the code that gets executed by EVM [51]. The main difference between EOA and contract accounts is that EOA can sign a transaction by its private key and can send this transaction to another EOA or contract accounts. Whereas, contract accounts are not able to initiate transaction by themselves [52]. The contract accounts function when they are triggered by a transaction being sent from an EOA.

These both types of accounts hold an account state that comprises of four components. These states are [52]:

Table 9: Components of an account state

Four Components of Account State	
nonce	EOA: Number of transactions that are sent from account address Contract Account: Number of contracts created
balance	Wei owned by this account address
storageRoot	Merkle tree's hash of the root node
codeHash	Contract Account: the code inside them is hashed and stored as codeHash EOA: Empty string is hashed and stored here

4.2.3. Transactions

In Ethereum, transaction is the way external entity would interact with Ethereum. It can be used by external user to update the state of the record or information stored on the Ethereum blockchain network. A transaction is a piece of information that contains cryptographic signatures. This transaction is generated by externally owned accounts (EOA) and then they are submitted to the Ethereum blockchain [52]. This signed transaction can be send from one account to another on the blockchain [51].

Transactions primarily have two types:

- ***Contract Creation:*** Every transaction contains smart contract that contain the code for performing various functions on the blockchain. This type of transactions is performed when we want to create Ethereum contract.
- ***Message Calls:*** Ethereum contracts have the ability to send message calls to another account on the blockchain. These message calls are not from externally owned account (EOA) but from inside the contract.

An Ethereum transaction regardless of types contains following elements [51][52]:

- ***From*** **–** sender of the message using the blockchain to the recipient. Both sender and recipient have a 20-bytes address.
- ***To*** **–** message recipient, also having a 20-bytes address.
- ***Value*** **-** the fund amount (wei) transferred from sender to recipient
- ***Data (optional)*** **–** contains the message that is being sent to the recipient
- ***Gas*** **–** For every transaction on the Ethereum blockchain the sender needs to pay some fees for performing that operation this fee is known as Gas. Every transaction contains the *gas limit* and *gas price* in it. This process is secure because the gas not consumed in a transaction is refund to the sender. And sender is charged for the only gas he consumes during transaction [51].
    - ***Gas Price*****:** that fee the transaction sender is willing to pay for gas
    - ***Gas Limit*****:** maximum gas that could be paid for this transaction

- **v, r, s –** used for creation of the signature that is used for identification of transaction sender
- **nonce –** keeps the count of number of sent transaction
- **init –** used for contract initialization for the first type of transaction i.e. Contract Creation

The contracts that are existing inside the decentralized applications they can interact with other contracts existing inside that defined scope. This whole process is called message calls or internal transactions [52].

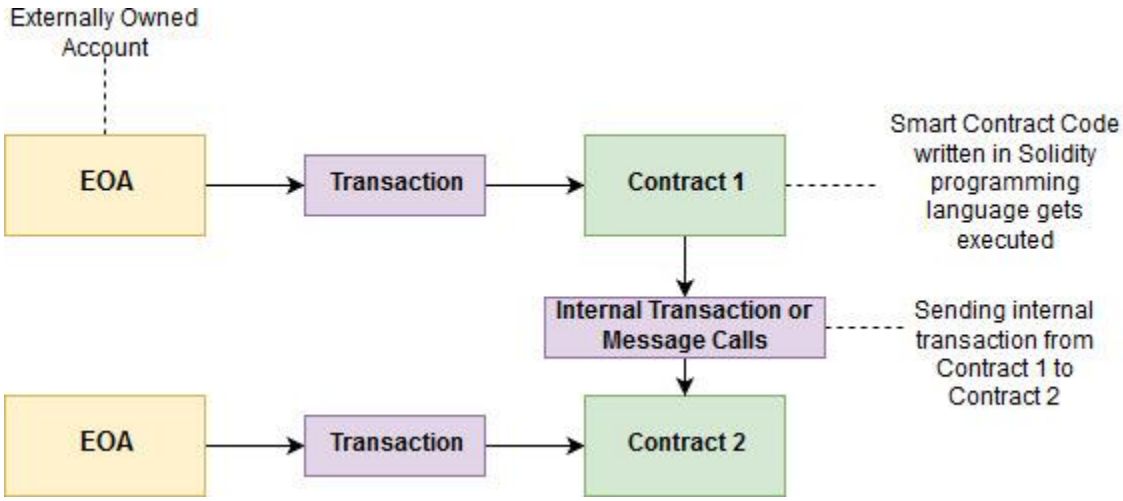

Figure 4: How a transaction is sent within the same state

The figure 2 above explains the scenario where two different contracts held by two different EOA i.e. externally owned accounts are performing the transactions. One transaction is done when an EOA wants to execute the Contract and second transaction is performed when a Contract 1 wants to interact to Contract 2 through an internal transaction or message calls. This transaction does not cost the sender any transaction fee i.e. Gas. This concept of Gas is discussed in the immediate next section.

4.2.4. Gas

For every transaction on the Ethereum blockchain the sender needs to pay some fees for performing that operation this fee is known as Gas. This transaction occurs due to the smart contract code being executed by the internal transaction or message. The transaction in this case would have a cost associated with it which is expressed as the number of gas units [51]. This 'Gas' or transaction fee is purchased from the miners in exchange of ethers as they are processing the transaction you just send. The miners can determine their minimum gas limit and you can add ether to your account to get it and pay to the miners for processing the transaction or one can use the Ethereum client to purchase gas which is specified for a transaction to be done [51]. For each transaction sender would have to set the Gas Price and Gas Limit.

- ***Gas Price*:** that fee the transaction sender is willing to pay for gas; it is measured in 'gwei'. Here 'wei' is smallest unit of ether and,

  1^{018} Wei = 1 Ether

  1 Gwei = 1,000,000,000 Wei

- ***Gas Limit*:** maximum gas that could be paid for this transaction. The gas being paid could not be less than this because in that case the transaction would not be processed.

The maximum fee a transaction needs to pay for it to be processed is measured by the product of gas price and gas limit. Mathematically it can be represented as,

Transaction Fee (Max.) = Gas Limit x Gas Price

Let us consider an example; a sender wants to send a transaction with a gas price of 20 gwei and gas limit of 50,000 [52]. So, the transaction fee would be,

Transaction Fee (Max.) = 50,000 x 20 gwei = 1,000,000,000,000,000 Wei

And,

1,000,000,000,000,000 Wei = 0.001 Ether

So, Transaction Fee (Max.) = 0.001 Ether

Another state of Gas component is 'out of gas', when the sender does not provides the sufficient gas for transaction execution this state occurs. In this case, the transaction results in failure and the reason are shown to be out of gas. The gas being used for transaction or the fee being paid for transaction is eventually for the miners who are responsible for performing the transaction.

Till this point, we have discussed that Gas is needed for transaction being executed on the blockchain but there is another scenario where the sender would need to pay the fee. That scenario is storage of a piece of information on the blockchain. Blockchain technology is not made specifically for data storage but it can still hold the metadata in it. This storage capability is helpful for many applications that are performing both the financial transactions and metadata storage tasks.

4.2.5. Block

As explained earlier blockchain are formed together by a number of blocks connected together in a peer-to-peer network thus making a decentralized application. These blocks contain the hashes in their header of the previous blocks on the chain. These blocks contain three things in them:

- Data
- Hash of current block

- Hash of previous block

The data could be anything as it depends on the type of blockchain. As in case of bitcoin, the data consists of coins that are actually electronic cash [15]. The hash that is stored in these blocks contains a SHA 256 cryptographic algorithm. This hash is used for unique identification of a block on the chain and due to this reason they are stored on the previous block so they can be connected to each other.

In Ethereum blockchain the block has following three elements in it:

- Header of block
- Transactions information
- Header of current block's ommers

The blocks in Ethereum have a parent block and the blocks that have an 'uncle' relationship with the parent block this block is referred as ommer block [52]. Before understanding this let us first know about different forms a block can take in Ethereum blockchain. These are:

- *Parent blocks:* acts as parent of the immediate next block in the chain
- *Orphan blocks:* blocks having no parent blocks attached with them
- *Ommers blocks:* these blocks are linked to the chain of blocks but are not included in the final chain of blocks and ate stale blocks

The following figure contains the contents and sub-contents of an Ethereum block. From the following diagram let us understand an important concept related to the Ethereum block i.e. difficulty. It is used while blocks are validated by the nodes connected in the network to ensure that it takes some time to compute the transactions and add them to blockchain. A nonce included in the block header is a cryptographic hash that should be computed and solved by the miners to mine a block using the respective consensus algorithms. The block difficulty level and nonce are mathematically represented as follows:

$$n \leq {}^{2^{256}}/_{H_d} \qquad \textit{here } H_d \textit{ is the difficulty of the block}$$

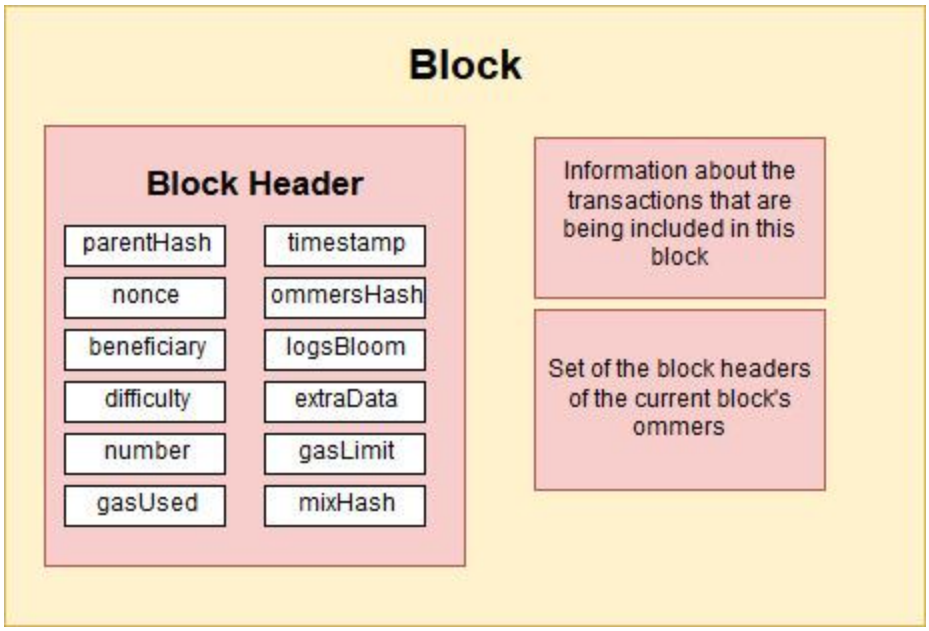

Figure 5: Components of an Ethereum Block

4.2.6. Smart Contracts

Smart contract are known as the piece of code that is used to perform any task on the blockchain. That task could be to exchange money, or any other valuable piece of work on the blockchain[53]. They run on the blockchain directly thus making themselves secure from any kind of tampering and alterations. Smart contract commonly use solidity language and they can be used to program any kind of operation that a programmer wants to do on the blockchain. After programming the required operations the programmers can compile them by using EVM bytecode that would be explained in next section. And after compiling them it could be executed and deployed on the Ethereum blockchain [54]. The programming language of JavaScript and Python are encapsulated with the Solidity language provided by Ethereum to write code in smart contracts.

The notion of Smart Contracts was given by Nick Szabo, in 1994. He gave the idea that self-executable contracts can be converted into codes; these digital contracts were used from the distributed ledger technology. This idea was used as smart contracts being used in blockchain technology for defining various functionalities that an application deployed on the blockchain should perform.

Smart Contracts was a unique innovation in the sense because it ended the role of third party existing between two entities performing a transaction. Instead of using a third party for performing the function of signing and allowing the transaction to be done. This was replaced by the two entities agreeing upon the smart contract being used as defining the terms and conditions for a function to be done.

The benefits provided by the smart contracts are discussed in detail in the section below,

Secure

For any component of a system to be secure the system should be able to protect itself from any potential threats or attacks from an external entity. The system that holds the property of being secure should be able to protect all of the information stored inside it from being accessed or tempered with by any external entity.

Just like a great feature of blockchain technology was it being secure, smart contracts also enjoy this benefit. As explained earlier, smart contracts contain the code that is executed for performing required functionality of the system in form of transactions. Considering the importance of this contract code it should be secure and temper-proof at all cost. As, blockchain technology is cryptographically encrypted by SHA- 256 algorithm. The smart contracts are also encrypted by this algorithm and this makes the transaction being executed because of the smart contract more efficient. As the danger of a transaction failing to occur would lessen.

Moreover, smart contracts are automated to perform the functionalities written in its code. The user does not need to process the critical steps to perform those functionalities but instead he would only need to send transaction to perform them.

State of Trust

The main aim of smart contract was to act as an automated contract that would need no middle party for helping two entities signs a contract. This middle party was replaced by smart contracts that were automated to perform these operations. The rules and regulations that were at first written legally and were followed by manual mechanism are now written in smart contracts. The blockchain technology i.e. in our case Ethereum blockchain ensures that two parties holding smart contract fulfill their tasks while following the defined rules and regulations in smart contracts.

Also the smart contracts being executed and used for performing transactions while making use of the triggers make it impossible for any third party to intervene and change the conditions of the contract. The Ethereum blockchain under discussion, also treats smart contract like an actual paper based contract, it has certain libraries through which we can define the ownership rights, access rights, roles and responsibilities of an entity using the system. This automation allows for creating a state of trust in the users of the blockchain technology in whole. As they are sure that their work is being hosted or done on a trusted platform backed by trusted smart contracts.

Transparency

In perspective of blockchain technology, achieving data transparency in any technology is to have a trust based relationship between entities. The data or record at stake should be secured and temper proof. Any data being stored on the blockchain is not concentrated at one place and is

not controlled by one node but is instead distributed across the network. The ownership of data is now shared and this makes it to be transparent and confidentially secure from any third party intervention.

For smart contracts, transparency is the rules and regulations defined by using the contract code. The entities associated with the contract are all agreed on these rules so they ensure their transparency during the transactions performed. The terms and conditions of a contract are not compromised during the transaction process it remain same this also ensures the transparency property of smart contracts.

Time-efficient

Another important benefit offered by smart contracts is time-efficiency i.e. the ability to perform the desired task in minimum amount of time. The world has transformed into a global village and time is the most important entity of any process being done in this global village. With the paper based contract system the participating parties needed to wait for the contracts to be processed and this system was not time-efficient.

In order to cope with the changing needs, smart contracts are the best choice for two parties signing a contract. As smart contracts are run and executed on the internet and they are governed by blockchain technology. So, they can be used for eliminating the delay in processing of contract's operation caused by the manual mechanism.

Table 10: Smart Contract Benefits

Benefits of Smart Contracts	
Secure	Smart contracts are automated and encrypted piece of code that is executed for performing various functions of a system. These properties of being automated and encrypted make it a reasonable choice for using it for systems having the requirement of security.
State of Trust	Also the smart contracts being executed and used for performing transactions while making use of the triggers make it impossible for any third party to intervene and change the conditions of the contract this ensure the state of trust among various entities associated with them.
Transparency	For smart contracts, transparency is the rules and regulations defined by using the contract code. The entities associated with the contract are all agreed on these rules so they ensure their transparency during the transactions performed.
Time-efficient	Another important benefit offered by smart contracts is time-efficiency i.e. the ability to perform the desired task in minimum amount of time. As smart contracts are run and executed on the internet and they are governed by blockchain technology. So, they can be used for eliminating the delay in processing of contract's operation caused by the manual mechanism.

4.2.7. Ethereum Virtual Machine (EVM)

The key benefits that Ethereum platform offers include the programmable blockchain. It provides its users with the choice to create their own applications functioning on the Ethereum. This is also the main difference between Ethereum and Bitcoin, as later is only focused on the transaction related to crypto-currencies. While the possibilities for Ethereum are many other applications thus not limiting it to crypto-currencies only [48].

The applications built using this platform are known as *Distributed Applications* (DApps). They contain a number of protocols that are packaged together to create a platform for DApps. These DApps contain smart contracts that have code defined by the user to perform some defined task of an application. That code is deployed and executed using the Ethereum Virtual Machine (EVM) [48]. Thus, the applications that are created using the smart contract are in actual being run on EVM.

A number of steps are followed for a decentralized application to be deployed on the blockchain using EVM. These steps are not needed by the developer or the user of the application to perform but they are automated by the whole blockchain ecosystem that is functioning by following all of these steps. These steps are explained as follows [55]:

1. Smart Contracts that are implemented using the Solidity language.
2. Compilation of the implemented smart contracts to generate the binary file for deployment.
3. By making use of Ethereum client applications developers would deploy the smart contracts on the Ethereum blockchain.
4. As this smart contract has certain functionalities that are needed by the user of the system for this purpose it needs to be available on a web browser or in other terms a DApp browser that is containing the front-end GUI of the decentralized application that we are developing. For this purpose we would need to follow two elements in our project:
    a. A smart contract to be deployed on Ethereum blockchain and it should be currently residing on the EVM block.
    b. A web implementation of the DApp composed of the smart contract code and front-end code for making the functionality visible to the user of DApp.

4.2.7.1.EVM Architecture

The Ethereum Virtual Machine (EVM) follows the stack-based architecture i.e. it stores the data from top-down and pushes down the older stored data when new data is added on the stack. The EVM architecture has three states in it[56]:

- *Immutable:* An immutable object holds the property of being unchangeable i.e. once created its state cannot be modified after that [57] [58]. Immutable data can help in development of simple applications and also in detecting the change that occurs using the

simple logic [58]. In EVM architecture, this state holds the EVM Code which is a vital part of the EVM.

  - *EVM Code:* This is the smart contract code that could be executed by Ethereum virtual machine. The human-readable code written in Solidity programming language needs to be translated into machine-readable code. This is machine-readable code is the EVM code.

- *Volatile:* The machine-readable code i.e. EVM code is used by this state. The term 'volatile' refers that when a machine's power is switched off it would lose all the contents that it has stored in its memory. This state of EVM holds following important components in it:
  - *Program Counter:* Commonly referred as PC, this component is used for storage of address of the current program that is to be executed by the compiler.
  - *Gas:* For every transaction on the Ethereum blockchain the sender needs to pay some fees for performing that operation this fee is known as Gas. Here in EVM it stores the amount of gas available in this state.
  - *Stack:* The memory of stack is 256bits and it can store 1024 elements inside it. It uses the mechanism of stack data to store elements inside it i.e. LIFO (Last In First Out).
  - *Memory:* The EVM memory is linear and this stores the items inside it. Memory component exists inside the volatile state, thus it loses its contents if the machine is powered off.
- *Persistent:* Persistency is the state in which the data is not changed on every alteration but instead it yields the data i.e. non-volatile. This state contains the storage component inside it.
  - *Storage:* The EVM storage is a store that contains key-value pairs. The mapping is 256-bit words to 256-bit words.

The code that is written in Solidity programming language that code is actually a high level language code which needs to be converted to EVM bytecode.

4.2.8. Modified Merkle Patricia Tree

The functioning of Ethereum is dependent or acts on a state machine where as a state machine is defined as a machine having various states that are used for it to function. In case of Ethereum the transactions are used as a mechanism for modifying states in an Ethereum network. These states are important and are stored in Modified Merkle Tree which is the combination of Patricia Trie and Merkle tree. Before understanding the purpose of Modified Merkle Patricia Tree also known as Merkle Patricia Tree (MPT) let us first understand the two concepts attached to it i.e. Patricia Trie and Merkle Tree.

4.2.8.1.Patricia Trie

It is a data structure which is represented as a trie in which an only child having a node is merged with its parent [59] and is also known as Radix Tree. In simpler terms, this data structure has a key which acts as a path so that the nodes that have same prefix or parent can also share the same path [60]. The basic state of every node in Patricia Trie or radix tree is as follows:

$$[i0\ i1\ \dots in, value]$$

It is basically a (key, value) store that is present in every node. Here $i0 \dots in$ are the key values that can store basic hexadecimal or binary value in it. The value part is the actual value that resides at that node where this (key, value) store is located [61]. As mentioned earlier, key stored would acts as a path in the trie for nodes that hold the same parents in the tree.

A major advantage of using this data structure is that it is fast and efficient while finding or looking up for parents or prefixes in the tree. Also this data structure is memory efficient as it does not take a huge amount of memory to store data inside it. This merge of parents of nodes who were an only child in the tree causes the number of child nodes of every internal node to be at radix *r* of the tree. Its application include mainly in routers as because it is used to implement IP routing tables.

4.2.8.2.Merkle Tree

Also known as tree of hashes, this tree contains the hashes of children nodes in the parent nodes along with the sum of hashes of all of the children nodes [60]. The leaf nodes of a merkle tree contain the data that is to be stored inside the tree. By analyzing a sub-tree of main merkle tree we can view the data existing within the tree, this property is known as an important property of merkle tree.

These trees can be used for verification of data which is stored and managed inside of a computer or when two computer systems are interacting with each other. The reason behind it being named as the hash tree is because all of the nodes in it except the leaf node contain hashes in it so due to this it is also known as hash tree. The data comparison also becomes easy in merkle tree because only the top hash values of the two nodes are compared. In case of blockchain technology these trees are used for organizing various transactions to make use of lesser resources.

These hash trees function on the SHA-2 family of algorithm. The hash tree contains its leaves as data block hashes and the nodes in the tree are the hashes of the child or leaf in the tree. The concatenation of these hashes is mathematically represented as follows:

$$hash\ 0 = hash\ (hash\ 00\ ||\ hash 01)$$

Here || is the symbol for concatenation. Concatenation is merging the two hashes i.e. hash 00 and hash 01 into one hash i.e. hash 0. It is actually as hash 0 has two child nodes in this scenario and these child nodes are hash 00 and hash 01, when concatenated these child/leaf nodes are then

concatenated to form the parent node or hash 0. The child/leaf nodes under the parent node can be increased i.e. it can have more than two children.

Any portion of the merkle tree could be used for verification of the correctness of tree in its whole. This can happen when we download a branch of the tree and we intend to check the integrity of this branch in case if tree is unavailable. So, let's consider the following figure 2 to understand this scenario, if we download the branch T_A of the tree we can know the details about Hash A and Hash AB, also we can compute the results for top hash of the merkle tree this could help in efficient verification of the correctness of the tree. Similar would be case for other branches or as we refer in terms of blockchain technology the transactions T. The main purpose of introducing the concept of transaction here is to understand that where does this normal transactions stand in the merkle tree and how can they be verified.

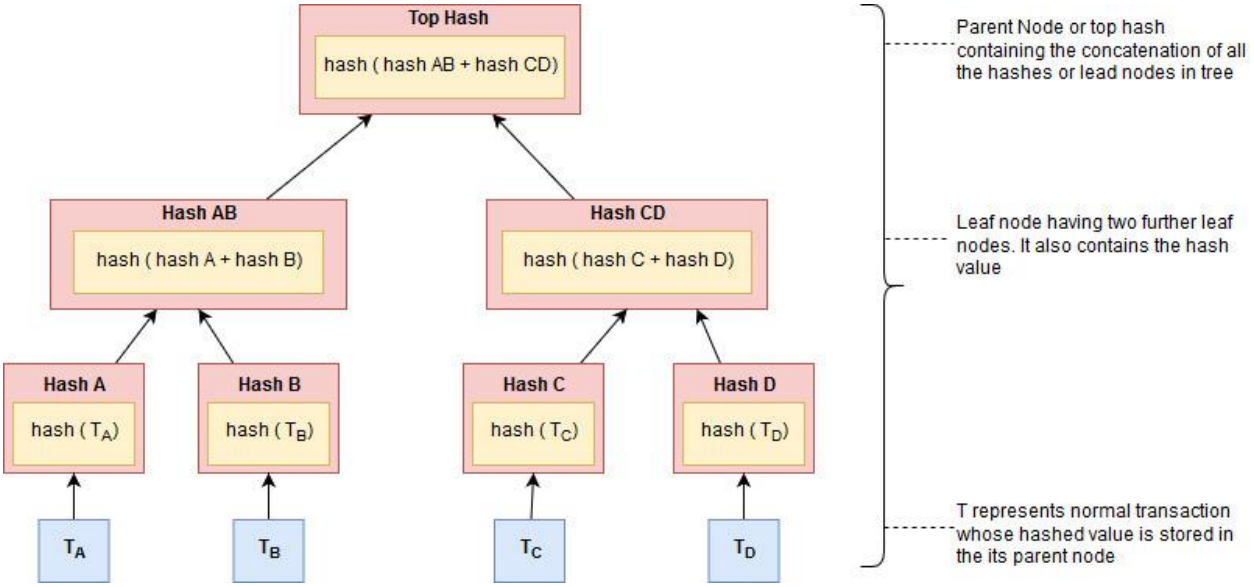

Figure 6: Merkle Tree

Advantages:

Following are the advantages offered by the merkle trees:

- Verification of transaction can be done easily as a user can check a specific transaction without downloading the entire tree.
- The transactions are organized in a way which makes the merkle trees to make use of fewer numbers of resources.
- With the transactions being organized in the merkle tree the verified transactions become temper-proof. As, they store hash values which change when tempered with by any third-party unauthorized access.

4.2.8.3.Merkle Patricia Trie (MPT)

Now that we have understood the underlying topics of MPT, we can know move on to understand the usage and purpose of it in blockchain technology.

In Ethereum all of the merkle tries use Merkle Patricia Tries which have following three roots in it from the MPT:

- State Root
- Transactions Root
- Receipts Root

Instead of jumping on the 3 tries and their usage in Ethereum let us understand this concept with the help of various states existing in Ethereum by its own. These various states are important and vital objects of Ethereum that make up the higher level operations or functioning of Ethereum.

1. World State - State Root/ Trie

The world state as the name refers is a global state which is being updated in a constant manner by the execution of transaction. It is mapping between the account addresses and state of accounts [62]. The world state could only be one and this state gets updated from time to time. Some [62] refer this concept in its simplest form as that network of Ethereum can be thought of as decentralized computer and the hard drive of this system is world state. This analogy explains the concept of world state in an easier manner. Also in a block, the *"stateRoot"* contains the root node's hash of the global or world state trie.

The state root/trie contains the path and value elements inside it. These elements store the two important objects inside it which could be used for referring to various details of an Ethereum account [61].

$$(path, value) \rightarrow (ethereumAddress, ethereumAccount)$$

And if looked in finer details the *account* of Ethereum contains the following 4 items in it. They are:

- ***nonce:*** Number of transactions sent from an EOA or the number of contract create calls made using this account
- ***balance:*** total ether owned by this account
- ***storageRoot:*** root of another patricia trie and is hash of the root node of account storage
- ***codeHash:*** for EOA it would be empty and for contract account it is hash of EVM code for this account

Storage Root/Trie: This concept is worth noting here because this trie is actually where the contract data is stored. Each account on Ethereum has a separate storage trie associated with it

and it could be accessed with a *path* [61]. This storage trie is not empty only in case of contract accounts in case of EOAs it is always empty.

2. Transaction – Transaction Root/ Trie

As explained in details before in Ethereum, transaction is the way external entity would interact with Ethereum. It can be used by external user to update the state of the record or information stored on the Ethereum blockchain network. A transaction is a piece of information that contains cryptographic signatures. This transaction is generated by externally owned accounts (EOA) and then they are submitted to the Ethereum blockchain [52]. This signed transaction can be send from one account to another on the blockchain [51]. It also has two types' message calls and contract calls.

All the transactions in a block are stored in a trie, with its root hash being stored in the block header. A transaction trie does not get updated after a block is mined. And before the block gets mined, the miner decides upon the transactionIndex [61]. This transactionIndex is stored in the *path* in case of transaction trie. In block, the "*transactionRoot"* contains the root node of the transaction trie.

3. Receipts Root/ Trie

This exists in the block header and is referred as the "*receiptsRoot"*. This contains the information about the transaction executed on the Ethereum. And it stores hash of every executed transaction in it. So, it is also known as the transaction receipt and in Ethereum per block there would be one transaction receipt. Again it stores the transactionIndex in it which never gets updated once determined and mined by the miner.

Following table gives a brief review of all the states of Ethereum tries.

Table 11: States in Merkle Patricia Trie

State Trie	The mapping between the accounts' address and account state.
Storage Trie	The contract data and is used for storage of this data.
Transaction Trie	The transactions included in a block
Receipt Trie	The executed transaction's receipt is stored in this trie

4.2.9. InterPlanatery File System (IPFS)

IPFS is a protocol that uses peer-to-peer network for data storage. It provides secure data storage as data stored on IPFS is protected from any alteration. It uses a cryptographic identifier that protects the data from alteration as any attempt to make change on the data stored on IPFS could only be done by changing the identifier. All the data files stored on IPFS contain a hash value

that is generated cryptographically. It is unique and is used for identification of stored data file on the IPFS [63].

This secure storage strategy of IPFS protocol makes it a favorable choice for storing critical and sensitive data. The cryptographic hash that is generated could be stored on the decentralized application to reduce the exhaustive computational operations over the blockchain.

IPFS protocol works using a peer-to-peer (P2P) network, this network contains a data structure known as IPFS object that contains data and link in it. Data is unstructured binary data and link consists of an array. The IPFS protocol works in the following way [64]:

- Files stored on IPFS are assigned a unique cryptographic hash
- Duplicate files are not allowed to exist on the IPFS network
- A node on the network stores content and index information of the node

IPFS follows a P2P network which is distributed due to which it has no single failure and there is also trustless state between connected nodes [65]. By design, it has following sub-protocols that help in functioning of the IPFS protocol. These sub-protocols are defined briefly as follows[65]:

- **Identity/ies –** Each node on the network is identified by *NodeID, public key and cryptographic hash.* These elements are used for identity management and verification of connected nodes
- **Network –** A number of nodes are connected on the IPFS network they all enjoy some benefits such as reliability, integrity and authenticity etc.
- **Routing –** This is used for finding the connected peers network addresses and the peers who can serve as particular objects
- **Block Exchange –** IPFS uses BitSwap protocol for distribution of data by exchange of blocks with peers. It functions like BitTorrent and have a new set of blocks ready for exchange with a list of blocks. This exchange is used for block distribution.
- **Objects –** IPFS builds a directed acyclic graph DAG or precisely Merkle DAG that has links contains hashes of the targets in the source. It is used for addressing of contents, resistance for content temper and to avoid duplication of objects on the IPFS.
- **File –** On top of Merkle DAG, IPFS defines object for modeling a file system.
- **Naming –** A naming system for IPFS file objects, this naming would be self-certified and human friendly.

4.2.10. HTTP VS IPFS

HTTP stands for HyperText Transfer Protocol; it is used by the World Wide Web (www) and it is used for communication between server and web clients. This protocol is also used for defining the various messages that are sent as HTTP requests and received as HTTP responses.

Aside from performing as a communication bridge the HTTP can also be used for other purposes of transmitting hypermedia documents of HTML (HyperText Markup Language). In the TCP/IP model this protocol lies on the 5th layer known as *application layer*. This is used for transmission of messages and is connected directly to the web client who is receiving and sending those messages. Here TCP stands for Transmission Control Protocol and IP stands for Internet Protocol. These are used for defining the mechanism the World Wide Web (www) functions on and the way by which a message is communicated across the server following many layers of TCP/IP model.

The application layer is responsible for the communication purposes and makes use of the underlying protocols for data transfer and exchange in the networking model [66]. The architecture of the HTTP is that of client-server where a client is connected to the web server and the server is responsible for handling all the message requests and responses. The centralization of this protocol was the reason due to which there was the need of a protocol that was decentralized. Before moving on to the decentralization provided by the IPFS protocol let us first understand the centralization of HTTP.

Suppose you are an internet user who is intending to access a document from a server. The document is hosted on a server and when you request to access it, the server directs you to the location the document is stored at. This is a simple scenario where everything works fine as you wanted to access the document and the server gave you access to it. But what if the document was tempered with in this case you are downloading a tempered document.

A way used by many huge corporations functioning on the http was to build data centers around the world to host their data on it. This solution was expensive and was only used by huge corporations.

With decentralization the IPFS allows the user to have access to temper proof documents and files or any other media. With IPFS the files are distributed across the network the bits and pieces of files are present in a distributed manner across the network of nodes. With this the need for data centers would also reduce, as the data would now be secure while it is distributed. Also with the hashing strategy used by IPFS it could be used as a unique identifier by its user to access the data.

Also an another advantage offered by IPFS is that of providing efficiency to access any file on it as when a request for access of a file is received the closest node connected in the network is used for providing access to the file [67]. It also protects the network from being crowded with the duplicates of files being stored in it.

For development and testing purposes following protocols were used,

4.3. Truffle

Truffle can be defined as a development environment and testing framework that is used along with EVM of the Ethereum blockchain for developing DApps [68]. Truffle provides a number of features that help the developers while creating their DApps. They are [68],

- Compile, link, deploy and manage the built-in smart contract
- Test the smart contracts
- Framework for deployment and migrations
- Network management
- Package management
- Contract communication (through console)

4.3.1. Truffle Boxes

Truffle environment has a unique way of providing various frameworks for helping developer to develop their DApps i.e. it contains various functionalities in forms of *boxes*. These boxes are unboxed for using those libraries in their projects by developers. These boxes contain the boilerplates that could be used by developers to customize them for their unique decentralized applications [69]. These boxes are pre-loaded with the smart contracts, migration files, front-end libraries and other dependencies along with the testing code. Some of the famous boxes of truffle framework are Drizzle, React, Metacoin and Webpack etc.

The steps for creating a truffle project for development and testing purposes are explained as follows,

1. Create a new folder or directory where you want to keep this project stored.
2. Inside the directory use the "unbox" command to save the boilerplate code inside the directory.
    a. From the list of truffle boxes available user can specify the box name as:
    Truffle unbox <box name>
    b. User can also create a truffle project having no boilerplate code but defined folders inside it by using command: truffle init
3. After unboxing is successful user can open the project using any trusted text editor to start developing their DApps.

The project would contain

- **Smart Contracts:** Located inside the *Contracts* folder this is the smart contract written in Solidity language and saved with .sol extension. This would contain the functionality that we want our DApp to perform. Truffle also contains the *Migration.sol* contract that is used for keeping track of deployed smart contract [70].

- **Migration Files:** With every smart contract there is a migration file attached that is a JavaScript library saved with .js extension. It is used as the deployment script for the particular smart contract [70].
- **Test Files:** It holds both the .sol and .js files that are written for testing purposes.
- **Configuration Files:** Named as *truffle.js*, this file is used for defining the network configurations for deployment and running of your DApps.

4.3.2. React Box Truffle

React is a JavaScript library which is used to build user interface of applications. The benefits that this library offers are that it's declarative, component-based and encapsulation. It makes use of components that are encapsulated and that manage their own states throughout the life of the components. The data inside the components is of two types: props and state. Props can be understood as the function parameters, they are passed to the components instead of function in react. State contains the input data that would change as the data changes on an input form. It can be understood as an object that is stored inside a component class as its property.

The building blocks of react are considered to be elements and components that combine together to keep the react applications to be functioning. Their brief definition is given below:

- *Elements* are the smallest building blocks of react application and it holds the property of immutability [71].
- *Components* are used for splitting the UI into separate and independent pieces and these pieces also hold the property of reusability [71].

4.3.2.1.Elements

A react element represents what you want to view on the UI screen or browser screen. These elements are plain objects which are cheaper to create. As mentioned earlier they are the smallest building block of the react applications the following example depicts what a react element looks like,

```
const element = <h1>Hello World! This is a react element </h1>
```

As it is evident by looking at the above example a react element is a simple object that is holding a value in it which would be visible on the screen. But for making this element to be visible on the screen we need to render the element into DOM. Here, DOM is Document Object Model which is an API for HTML and XML documents.

To render an element in react DOM you need the function ReactDOM.render () in which you would pass the elements as the arguments of this function. So, the line of code shown above would now be as follows,

```
const element = <h1>Hello World! This is a react element </h1>
ReactDOM.render (element, document.getElementById ('root'));
```

The above code will display the text *Hello World! This is a react element* on the browser but in react way by rendering the element in the react DOM. A number of react elements could be passed in the ReactDOM.render () function.

As mentioned before, react elements are immutable [71] i.e. an element that is unchangeable or not be able to change over time. For making any changes in the elements of the react we could add new elements in it which are rendered to react DOM.

4.3.2.2.Components

Components are used for splitting the UI into separate and independent pieces and these pieces also hold the property of reusability. They can be conceptually understood as JavaScript functions they accept 'props' as input and return the elements on the screen [71].

It makes use of components that are encapsulated and that manage their own states throughout the life of the components. The data inside the components is of two types:

- *Props:* They can be understood as the function parameters, they are passed to the components instead of function in react. The word *prop is* used as a short term for properties.
- *State:* It contains the input data that would change as the data changes on an input form. It can be understood as an object that is stored inside a component class as its property.

The following code snippet could be used for understanding what a react component looks like and how would it work together with react elements.

```
function Index (props) {
return <h1>Hello World! This is a react element, {props.name} </h1>;
}
```

The above example is taken from the guide provided by the React JS official documentation [71]. The above code snippet depicts a JavaScript function that accepts props as an argument and returns a react element which makes the above function a valid component of react. The above function could also be implemented using a Class instead of JavaScript function the following code would explain how a Class would perform the same functionality or operation.

```
class Index extends React.Component {

  render () {

  return <h1>Hello World! This is a react element, {props.name}
</h1>;
            }

}
```

The above given example is also another way for defining a component using Class instead of a JavaScript function. Here the class name is *Index* and it is extending the React component to render. The above discussed JavaScript is also referred as *function components*.

Props: Another important concept to understand here is that when a component is declared as a function or class it should not be able to modify the props being passed as an argument to them [71]. This can also be understood by the *pure* functions used; these functions do not change or modify their input due to which they are called *pure* functions. The functions which change their inputs are known as *impure* function. Here the react components must act as pure functions and should not change the props passed as its argument.

States: It contains the input data that would change as the data changes on an input form. It can be understood as an object that is stored inside a component class as its property. State is just like props but they are private and their control is hold by components [71]. The state is defined by using *this.state* with the objects we want define as states in the components.

Another important feature of React is its lifecycle methods that are included in a react class. The commonly used lifecycle methods are componentDidMount (), componentDidUpdate () and componentWillUnmount (). They are explained as follows:

- *ComponentDidMount:* To mount a component is to add it the tree or in simple terms to insert it into the DOM. This function is invoked when the component is mounted and is also used to set the state that was initially set in the constructor of the class. The setState function should be used for this purpose which would set the state objects; this would also trigger the render function when called [72].
- *ComponentDidUpdate:* Any change to state or props of the react components would cause for this function to be called as now the components are being re-rendered [72]. This function is called right after the update actually happens.
- *ComponentWillUnmount:* Before a component is unmounted or destroyed this function is called to perform any cleanup of the services or properties utilized in the ComponentDidMount function. The function setState should not be called in this lifecycle function because once unmounted it would never be mounted again in that lifecycle [72].

4.3.2.3.Events

An event in simple computing terms is an action that results by a user triggering any activity that activates the event. The following code snippet could be used to understand what react events are and how are they triggered and handled in react classes. The example is taken from [73].

```
class Index extends React.Component {

      state = {  inputText: “”  };

     handleChange = event => {
     const value = event.target.value;
     this.setState ({
      inputText: value
     });
    };
render () {
    return (
       <div>
            { /* Render input entered */}
            <label>ENTER YOUR NAME: {this.state.inputText}</label>
          </div>
          <div>
           { /* the function handleChange() is triggered when text
is entered */ }
            <input
              type="text"
              value={this.state.inputText}
              onChange={this.handleChange}
              placeholder="Enter your name"
            />

       </div>
)};
}
```

The first step is to define the state values in the start of the class before any rendering function is included. And in the render function assign this state value to a UI object like an *input* tag as in the above example. This would contain the input of the user in it; this value would be used in the handleChange function which is defined before the render function. The onChange event is triggered when any value is entered in the *input field.* The handleChange function is for handling the text entered by the user i.e. to set a new state for the input in the state variable which is inputText in the above code snippet.

The above code would not actually display anything on the screen as the state variable value being updated in the handleChange function is not being displayed by using any other UI tag. The above code is only for understanding the process or the way an event is triggered in the react components. The only rule one would need to keep in mind is that react uses the camelCase for defining or naming the events instead of lowercase.

An important functionality offered by react in case of events is of cross-browser compatibility. For this the react uses the synthetic events for defining events in general. The synthetic event is a cross-browser wrapper which enables cross-browser compatibility which acts as a wrapper around the native event of the browser [74]. This occurs by the react normalizing the events so they would have consistent properties across various browsers.

4.3.2.4.Forms

React uses the controlled components mechanism for handling the form elements and the input entered in the form elements by the user. As mentioned earlier react maintains states and its lifecycle inside its components. When a user enters any value in the HTML form which is created by using react at the backend. These form elements maintain their own state which is updated when an event is triggered and updated by using JavaScript functions. This whole process is known as controlled components.

4.3.3. Truffle Configuration

This is a configuration file written using JavaScript language and is located in your root folder of project. The file is named as truffle.config and is used for mainly configuration purposes of the project.

```
const path = require ("path");

module.exports = {
  // See <http://truffleframework.com/docs/advanced/configuration>
  // to customize your Truffle configuration!
  contracts_build_directory: path.join (__dirname,
"client/src/contracts"),
  networks:{
      development: {
           host: "127.0.0.1",
           port: 7545,
           network_id: "*"    //Match any network id
      }
  }
};
```

The code snippet shown above contains the default setting for configuration of the truffle project. The commands for running the project in test environment are used alongside with this configuration file. This command is:

```
truffle migrate
```

The network configuration explained in the above code is used for deployment of contracts on the MetaMask. If this configuration of network is not defined the contract would not be deployed by truffle. While the migration process is initiated the specified network is used for deployment of the contract. The network has sub requirements which are development and live. The *development* has host, port and network id. The host would be 127.0.0.1 which is the localhost of the computer and the port is 7545 or 8545 which is the default port used by MetaMask. The *live* option has some configuration values which include gas, gas price, from, provider etc. It also includes the other configuration values of host, port and network id.

- **Gas:** For every transaction on the Ethereum blockchain the sender needs to pay some fees for performing that operation this fee is known as Gas. This includes the gas limit for deployment of the contract.
- **Gas Price:** The actual price for deployment is included in this portion.
- **From:** This is the address of the account which is used for migration and deployment. It is the first account which is used by the client for deployment.

4.4. Ganache

For developing decentralized applications on Ethereum, Ganache is used as a personal blockchain. It can be used for development of DApps, deployment of smart contracts and running tests [75]. It is also known as a virtual blockchain that can be used by developers as it provides 10 Ethereum addresses for development purposes. These addresses contain 100 ethers pre-loaded in them. These addresses can be used for development of applications and running tests on the smart contracts or overall application's functionality. As specified in the official documentation Ganache offers two flavors i.e. Ganache CLI (Command Line Interface) and Ganache UI (User Interface). It runs on the localhost of the developers system i.e. at 127.0.0.1 and port 7545 for UI and port 8545 for CLI.

The addresses on the Ganache can be considered as nodes running on Ethereum blockchain and the decentralized applications can be run on them. It looks like a virtual network of nodes but acts like a real world application of Ethereum having nodes connected on its network. The following steps define how a user can start using the Ganache blockchain for development and testing purposes on their systems.

1. **Installation:** The user can download the version of Ganache according to his operating system. After download ends, user can initiate the process of installation by clicking on the Ganache setup downloaded. The download wizard would guide user to specify

various requirements. As installation process ends, user is provided with a mnemonic seed that must be kept secure for using it in case of account being lost.

2. **Creating Workspace:** As installation process is completed, user can run the Ganache blockchain. Depending on the type installed by the user i.e. Ganache CLI or Ganache GUI, both of these are used blockchain development. On first setup, user is prompted to select to create a workspace for using Ganache. It provides two options i.e.
    - **Quickstart workspace** – Selecting this option would provide the user with a new blockchain having the genesis block every time you open the Ganache on your system. This workspace would provide four panels:
        i. **Account:** The 10 accounts provided by default for development purposes. The user can view account address, balance, Tx count and private key in the account panel.
        ii. **Blocks:** The blocks mined are visible on this panel. It also includes the block number, data & time when block was mined, gas used and transaction done on the account.
        iii. **Transactions:** All of the transactions that are running on the transactions are presented in the form of list.
        iv. **Logs:** Contains the log of the server.

    - **New workspace** – This option can be selected for the scenario where a user want to customize the workspace according to his own project requirements. The user can save the Quickstart workspace and create a new workspace as well [76].

4.5. MetaMask

MetaMask is another tool that helps in the development and testing of Ethereum blockchain development. The Ethereum addresses provided by Ganache can be used to deploy and run the decentralized application on the MetaMask. It acts as a bridge that allows the developers to view the working of the decentralized applications on the web browser. It also benefits for allowing the developer's browser to act as an Ethereum blockchain without running complete node of the blockchain[77]. The work flow of MetaMask is explained in the following figure,

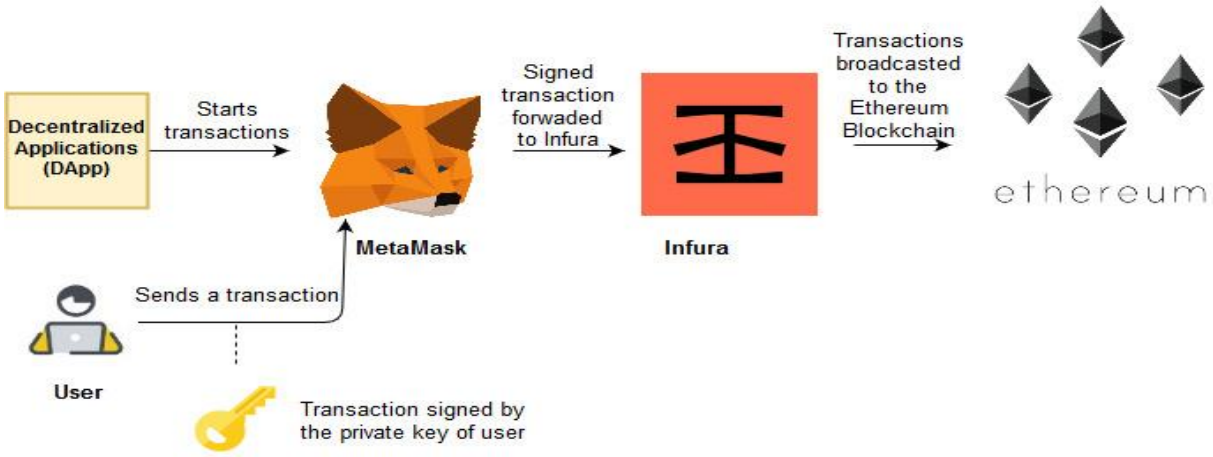

Figure 7: Workflow of MetaMask

As explained by the MetaMask documentation [78], it was created to develop such applications that would fulfill the needs of security and usability. The following are the important tasks that MetaMask can perform:

i. Install and Create Account

Every user of MetaMask can create an account on it, to use its functionalities. The MetaMask would hold two keys inside it i.e. public key and private key. Public key is also considered as the address of the user on the MetaMask. Private Key as known from name is private and needs to be kept secure. If this key is given the user would lose all the transactions he performs on the blockchain.

The process of creating account on the MetaMask is simple and easy to understand. It starts by the user accessing the official website of MetaMask. The website offers various browser extensions to use MetaMask in your browser. After enabling the add-on in your browser, user can initiate the account creation process. For user to create an account, a password needs to be entered in the installed extension of MetaMask. After entering password, the MetaMask provides a mnemonic seed that must be kept secure by the user; it would be helpful in case if user needs to recover his account. As mentioned before, user account has two keys public and private. These keys are used for interaction with the decentralized application

ii. Import Account

MetaMask allows the user to import an account from any other account. In our case, we are using Ganache as that client for importing accounts in MetaMask. The steps that should be followed to import an account in MetaMask are explained as follows:

a. Login to the MetaMask and click on the drop down menu to view the import account option.
b. Upon clicking the import account, the user is provided with a selection panel or drop down panel that holds two options that are used for authentication purposes.

    - *Private Key:* Enter the private key of the account user want to import in the MetaMask
    - *JSON File:* Or user can upload the JSON file attached with his account and that JSON file would be used for importing that account in MetaMask.

After performing the above steps, the accounts are imported from the client and these accounts can be used by the user for performing any transactions or other purposes.

iii. Network

The default network for MetaMask is main net but it also allows various other networks to be used by making use of its browser's extensions. These networks are:

- Main Ethereum Network
- Ropsten Test Network

- Kovan Test Network
- Rinkeby Test Network
- Localhost 8545
- Custom RPC

As understandable by their names above specified networks could be divided into two categories i.e. Main and Test network. Main network is the Ethereum network that is by default used for interaction of decentralized applications to the Ethereum. While the test networks are used by the developers to test their decentralized applications without paying any transaction fees etc. All of the networks also have different consensus mechanisms.

Customer RPC is used by defining the network developer is working on such as localhost 7545 in case of working with Ganache. It is done by

a. Opening the MetaMask extension in developer's browser and clicking on the *network* drop down menu.
b. Developer would select the Custom RPC network from this menu which would lead to the menu for customizing the network.
c. Enter the URL the developer wants to add on MetaMask and start using it by selecting it as a network while running your DApps.

4.5.1. MetaMask Connection

4.5.1.1.Ganache

The following code could be used for connecting the MetaMask and Ganache, the two tools that we have explained above.

```
if (typeof web3! == 'undefined') {
  App.web3Provider = web3.currentProvider;
  web3 = new Web3 (web3.currentProvider);
} else {
  // If no injected web3 instance is detected, fallback to Ganache.
  App.web3Provider = new web3.providers.HttpProvider('http://127.0.0.1:7545');
        web3 = new Web3 (App.web3Provider);
      }
```

The above code is in JavaScript language the 'App' is the class name of the JavaScript class. This component of class is used for injecting web3 for connecting MetaMask and Ganache. The Http Provider differs for Ganache UI and Ganache CLI. As shown in above code snippet for UI it is *http://127.0.0.1:7545* and for CLI the http provider is *http://127.0.0.1:8545* .

4.5.1.2.Truffle

The following code could be used for connecting the MetaMask and Truffle, the two tools that we have explained above.

```
if (typeof web3! == 'undefined') {
  App.web3Provider = web3.currentProvider;
  web3 = new Web3 (web3.currentProvider);
} else {
  // If no injected web3 instance is detected, fallback to Truffle Develop.
  App.web3Provider = new web3.providers.HttpProvider('http://127.0.0.1:9545');
        web3 = new Web3 (App.web3Provider);
      }
```

The above code is in JavaScript language the 'App' is the class name of the JavaScript class. This component of class is used for injecting web3 for connecting MetaMask and Truffle.

4.6. System Design

System design is the most important and vital part of any framework as it is used for the development of the system from its theory. This section includes the modules, architecture and various elements that are combined together to form the whole system's framework. As defined earlier the purpose behind this proposed framework is to create such a decentralized system that is temper-proof, secure and confidential blockchain-based system for electronic health records.

Every system have some entities that are functioning together to keep the system functioning. The entities of this proposed framework along with the system design are explained in next section. As visible in below figure 2, the proposed framework or system has three entities or modules. These modules when combined together would keep our system working. These entities or modules have further concepts that need to be understood they are explained as follows.

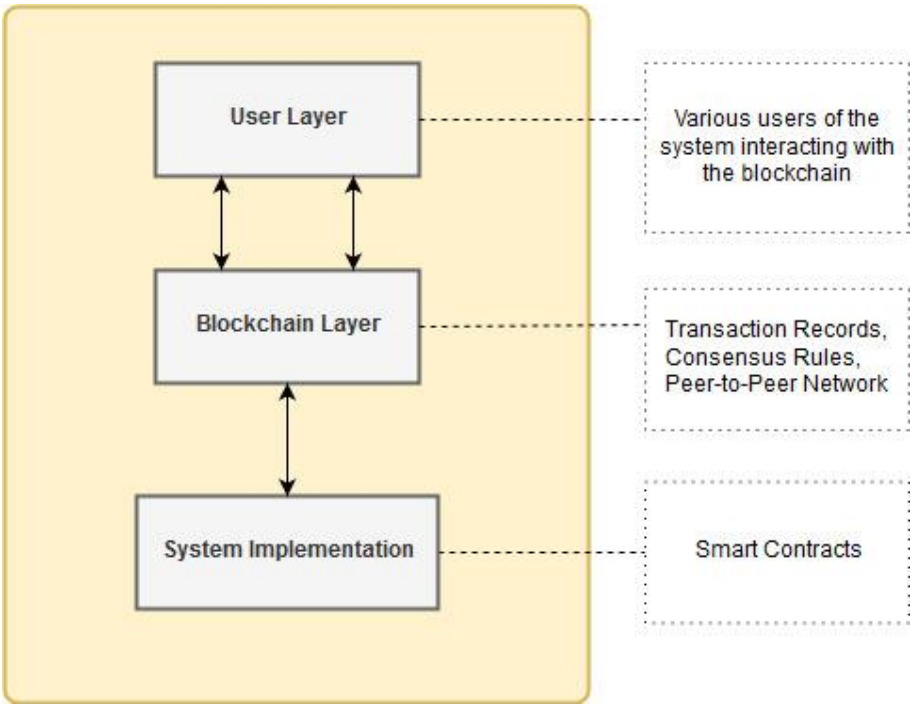

Figure 8: System Design of Proposed Framework

4.6.1. User Layer

A user of a system is defined as an individual who makes effective use of the system and its resources. A user can have various features these can be a role name, an account on the system and some identification, authentication system [79]. These features are used to let a user be identifiable on the system.

Just like any user attached with a system, blockchain-based decentralized system also has users associated with it. Their basic task is to be able to use the proposed system and its resources. The users could belong to hospitals including patients, doctors, nursing staff and administrative staff etc. The main task of these users could be to interact with the system and perform basic tasks such as create, read, update and delete the medical records.

The users using this system would be accessing the system's functionality by a browser which in technical terms we refer as DApp browser, as it is containing the GUI (Graphical User Interface) of the DApp i.e. our proposed system framework. The GUI contains all the functions that could be accessed by a particular user. The user according to the assigned role could use this GUI for interacting with the other layer of the system i.e. blockchain layer.

4.6.2. Blockchain Layer

The next layer on the system is the blockchain layer; this layer contains the code or mechanism for interaction of user with the DApp which is functioning on the blockchain. This layer contains three elements inside it. They are:

- **Blockchain Assets:** In Ethereum blockchain, transaction is the way external entity would interact with Ethereum. It can be used by external user to update the state of the record or information stored on the Ethereum blockchain network. This layer stores the record of transactions that are done on it. These transactions are treated as *assets* by the Ethereum blockchain as they are piece of information or data that user intends to send to another user on the chain or to simply store it for using it later.
- **Governance Rules:** As explained in previous sections, blockchain technology in general follows consensus rules for transactions to be done and computed. For this they need some consensus algorithms to keep the blockchain temper-proof and secure. Also, usage of these consensus algorithms ensures that blockchain technology remains decentralized, distributed, transparent and confidential. Ethereum blockchain uses Proof of Work (PoW) consensus algorithm that is explained in previous sections in detail. The reason behind using a consensus algorithm is also for ensuring that *governance* of blockchain is maintained in a trusted manner which is through consent from all the trusted nodes attached to the blockchain network.
- **Network:** Ethereum blockchain uses the peer-to-peer network. In this network all the nodes are connected as *peers*. With no node acting as the central node controlling all the functions of the network. The reason behind using this network was because the idea was to create a distributed platform not a centralized. So, using a network where all the connected nodes have equal status and right was the best choice this technology could have done.

The above mentioned components were explained in an abstract for getting an insight as to why these elements are important to the DApp. As mentioned before, our proposed framework consists of users that could be patients, doctors, nursing staff, pharmacist and insurance companies. They are given granular access as they should have varying level of authority on the system. The following section explains how these access rights and basic functions are being performed in the system.

4.6.3. Transaction

The system includes following transactions:

- **Add records** would create patient's medical records in the DApp. It contains the fields of ID, name, co-morbid, blood group, and IPFS hash. The patient's basic medical records is stored along with the IPFS hash that contains the file uploaded containing the lab results or other medical records of patient.
- **Update records** would update the medical records of patient. This can only change the basic information of the patient not the IPFS hash. IPFS hash is non-updateable to ensure security of records.
- **View records** would let the user view the medical records of a patient stored in DApp.
- **Delete records** would make the user be able to delete record of any patient.

- **Grant access** for each of the above mentioned transactions, user would need certain role to have access to them i.e. only the doctor or nursing staff can make changes in the records of the patient or add them. So, add and update records would only be accessible to these entities. A patient can view his medical records but won't be given the access to add or update them.

4.6.4. System Implementation

As already explained in the previous sections, the system would be implemented by using the Ethereum and its dependencies. The users need to have a wallet or a personal account address on the blockchain for the system to be fully functional. The system implementation is explained in the following section.

4.6.4.1.Smart Contracts

As explained earlier, smart contracts are an important part of DApps as they are used for performing basic operations. Following contracts are included in this framework:

- Patient Records
- Roles

These contracts are used for giving access to the users on the DApp and performing CRUD operations on the records of patient. The *Patient Records* smart contract is made purely for implementing the functionality of the proposed framework. It performs the CRUD operations along with the defining roles for access of these functions.
The second contract mentioned above i.e. *Roles* is a pre-defined smart contract by the OpenZeppelin smart contract library. This library contains several smart contracts performing various functionalities that could be used for creating your own smart contracts. The reason behind using this library was to make use of the benefits it provides i.e. tested and community reviewed code. The *Roles* smart contract belongs to the Asset library, which is a sub-library of the OpenZeppelin library. The asset library contains various other contracts for defining the access rules but the roles library provide a granular role definition mechanism which was the main reason behind selection of this smart contract.
The algorithm for defining the *Patient Records* smart contract is given below. It defines all the operations that are being performed in it and various conditions that are associated with them. It also explains how the *roles* are being maintained for granting access to a particular functionality.

Algorithm 1 Smart Contract for Patient Records

```
Assign Roles:
function Define Roles (New Role, New Account )
        add new role and account in
        roles mapping
end function
Add Data:
function Add Patient Record ( contains variables to add data )
        if ( msg.sender  = = doctor ) then
                add data to particular patient's record
        else Abort session
        end if
end function
Retrieve Data:
function View Patient Record ( patient id )
        if ( patient id) = = true then
                retrieve data from specified patient ( id )
                return (patient record)
                to the account that requested the retrieve operation
        else Abort session
        end if
end function
Update Data:
function Update Patient Record ( contains variables to update data )
        if ( msg.sender  = = doctor ) then
                if ( id = = patient id && name = = patient name ) then
                        update data to particular patient's record
                        return success
                else return fail
                end if
        else Abort session
        end if
end function
Delete Data:
function Delete Patient Record ( patient id )
        if(msg.sender = = doctor ) then
                if ( id = = patient id ) then
                        delete particular patient's record
                        return success
                else return fail
                end if
        else Abort session
        end if
end function
```

4.6.4.2.OpenZeppelin Library

This library is loaded with the smart contract codes needed for secure development of contract codes. This library contains several smart contracts performing various functionalities that could be used for creating your own smart contracts. The reason behind using this library was to make use of the benefits it provides i.e. tested and community reviewed code. Its sub libraries include

- Access
- Crowdsale
- Cryptography
- Drafts
- Introspection
- Lifecycle
- Math
- Ownership
- Payment
- Token
- Utils

These sub-libraries contain the contract code which can be used in your own contract code and can be customized according to the need of the system. This code is tested and reviewed by the community i.e. group of authorized individuals who are responsible for this task. This tested and reviewed smart contract can be then be used by developers by ease because they are no longer worried about the possible security flaws that could occur due to their smart contract. So, in short developers now only need to focus on the optimization of their decentralized applications and this library is going to focus on preventing any risks associated with security of your DApp. The OpenZeppelin library could be installed in your project by using the following command:

```
npm install openzeppelin-solidity
```

This is installed in your project directory and the pre-requisite for this library is node.js and npm package. These are responsible for installation of the library and saving its packages in the root directory of your project. The *Roles* smart contract belongs to the Asset library, which is a sub-library of the OpenZeppelin library. The asset library contains various other contracts for defining the access rules but the roles library provide a granular role definition mechanism which was the main reason behind selection of this smart contract. The *Roles* contracts as mentioned earlier, is a predefined by OpenZeppelin library. This contract is used for defining access rules of a user. The smart contract contains three functions i.e. add, remove and has.

- **Add:** This function is used to assign role to an account. It contains the role name and account address to which a role is assigned. This function at first checks that the account

to which the role is being assigned to is not already having a different role stored in roles list.

- **Remove:** This function is used for removing the account address' access to a role. The respective account address would no longer be holding that role name for performing that function for which this role is needed.
- **Has:** This function is used for checking that does an account hold the role or not. This could be used in Add and Remove function to check whether the account address already has a role name assigned to it or not.

These role names along with the account address are stored in a mapping that in simpler terms could be referred to as *Roles List.* This list is used for accessing and validating while using a function to check that does an account is allowed and assigned a role to perform a function.

Table 12: OpenZeppelin Sub Libraries

Description of Sub-libraries of OpenZeppelin Library	
Access	For access control and defining the access rules for users who are accessing the system's functionalities
Crowdsale	For management of Crowdsale of tokens, it would help the investors to buy the tokens in return of some ether.
Cryptography	It includes the smart contract of ECDSA and Merkle Proof. This is basically used for performing cryptographic operations
Drafts	It includes the smart contract of Counter, ERC20Migrator, Signature Bouncer, Signed Safe Math, and Token Vesting.
Introspection	It is used for finding whether a contract supports an interface you want to include in your project.
Lifecycle	It is used for maintaining the life cycle of a smart contract and include the pausable contract which is used for implementation of emergency stop operation.
Math	This provides mathematical functions that protect contract from overflows and underflows.
Ownership	This is used for defining user permissions and is also used for authorization purposes by defining contract ownership.
Payment	This could be used when payment needs to be taken from a group of people or needs to be pull payments.
Token	It is the mapping of addresses to the balances and these balances are of the user's accounts.
Utils	This offers a number of utilities that could be used for development purposes. These include smart contracts of address, arrays, and reentrancy guards.

4.6.5. Illustrative Use Case Scenario

The basic working of this decentralized application (DApp) could be understood by following through the process by which user would interacts with it. As depicted in figure 2 below, a user interacts with the User Interface (UI) of the DApp. The UI is used for interaction with the DApp deployed on the Ethereum blockchain. The user is only shown the front end of the application with no source code or working of the Ethereum blockchain visible in the browser.

At the back end, the Ethereum blockchain has some components or packages that combine together to keep it functioning. The DApp contains three main packages that are used for running and deploying the application on the Ethereum client. These packages or libraries are ReactJS, Web3JS, and Smart contracts; they are explained in below,

4.6.5.1.ReactJS

React is a JavaScript library which is used to build user interface of applications. The benefits that this library offers are that it's declarative, component-based and encapsulation. It makes use of components that are encapsulated and that manage their own states throughout the life of the components. The data inside the components is of two types: props and state. Props can be understood as the function parameters, they are passed to the components instead of function in react. State contains the input data that would change as the data changes on an input form. It can be understood as an object that is stored inside a component class as its property.

4.6.5.2.Web3JS

A collection of different libraries bundled together for interaction of developer's system to the Ethereum node that can be remote or local [80]. The main purpose of web3js library is to help in developing such client application that can interact with the Ethereum blockchain. The various functionalities that it offers are creating smart contract, read & write operations using smart contracts and sending ether (Ethereum crypto-currency) from one account to another on Ethereum blockchain. For interaction with the Ethereum, web3js uses the JSON-RPC (remote procedure calls) for its interaction with Ethereum blockchain.

Web3JS has a number of sub-dependencies, an important one of them is explained below,

4.6.5.3.Infura

For a DApp to function it needs to interact with Ethereum blockchain the above mentioned libraries are used for this purpose but to have access to the Ethereum blockchain we need to form some sort of connection between them, this is provided by the Infura RPC URL. Its main features are that it is blockchain-based service, reliable and a secure distributed storage system [81]. It can be used for providing developers a free service to connect to the Ethereum node. Infura can be used as a tool that is an alternative for Geth and Parity that are used for running your own Ethereum node.

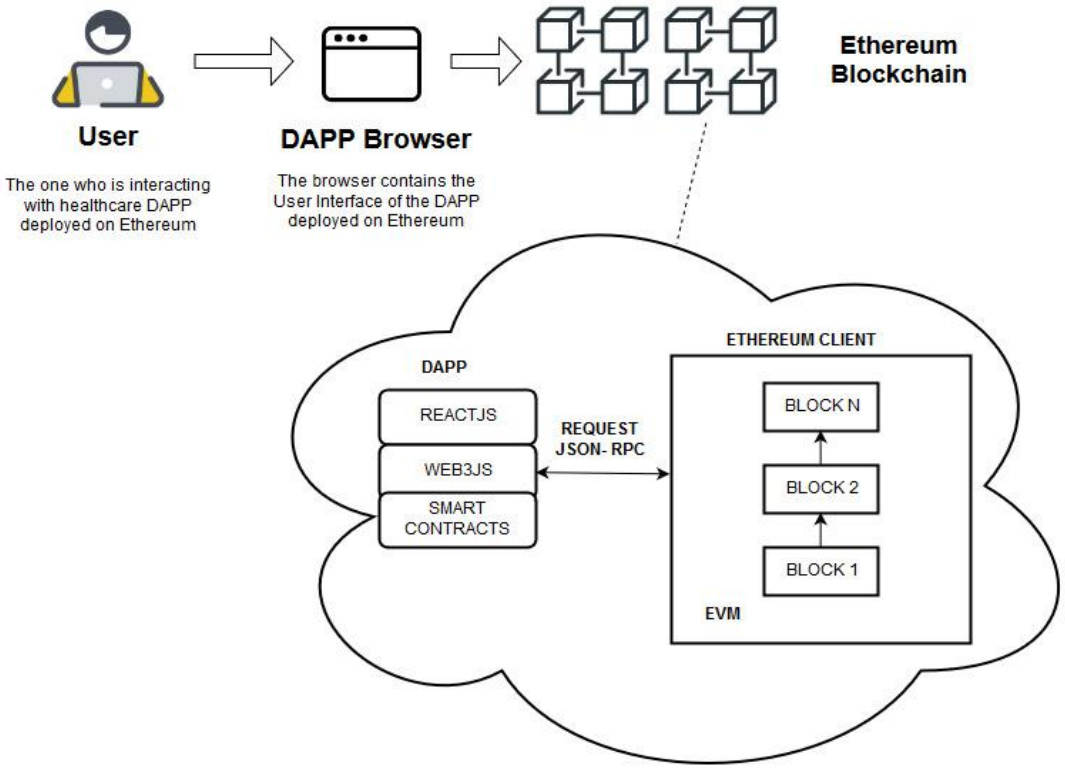

Figure 9: User Interaction with DApp

In technical terms, the working of system could be defined as that a user be able to perform basic create, read, update and delete operations. The access to these functions depends upon the role of the user i.e. a doctor would be allowed to make some changes in the patient's medical records. And a patient would be able to view his medical records that are available on the system. As record of patient would be of huge volume so we also provide on off-chain storage solution of using InterPlanatery File System (IPFS). The patient record being stored would contain basic patient information along with IPFS hash. The IPFS hash could contain the lab tests or other information that are to be stored with patient's medical records.

Usage Scenario 1 – Activity Flow

Access Granted: Let us know understand the flow of activities for a user i.e. doctor who intends to add a patient records on the system. The figure 4 below depicts the whole process of this scenario's activity flow. This whole process starts from the Administrator, who is some trusted individual inside the hospital or healthcare organization. The administrator is responsible for assigning the *Roles* to various users of the system. This individual would obviously have some technical skills and experience as well for understanding and using the system. So, the first activity would be that administrator assigns roles and this would include *Role Name* and *Account Address* of the user who is being assigned that role. Every user of this proposed system would have a role name and account address for using the system. So, after administrator assigns this

user some role, that role name and account address is stored in a roles list for validation purpose required in later steps.

After roles are assigned, now when a *doctor* wants to perform some operations on the proposed system he would at first request to perform them. The system would verify the doctor's role name and account address from the *Roles List* and allows the doctor to perform those functions after validation returns success. The doctor would perform the desired functions and the system would store the information on the *Blockchain* that would perform transactions for that information. Once the transaction is confirmed the system receives the message of success from the blockchain layer that doctor can view on the DApp browser on which the whole proposed system is being visible.

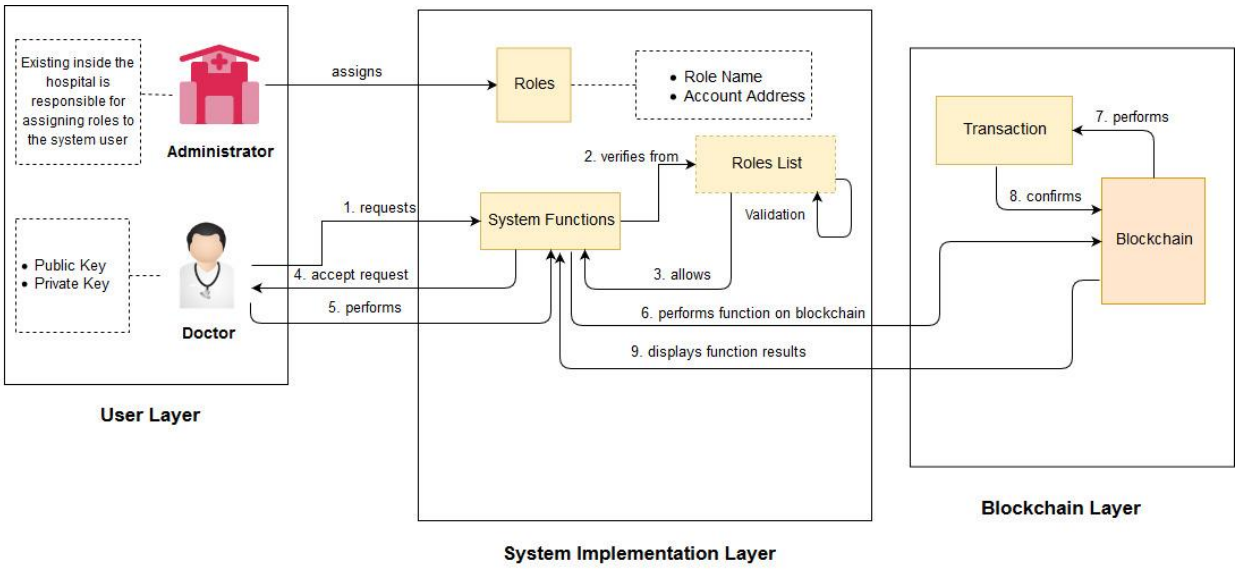

Figure 10: Usage Scenario Doctor (Access Granted)

Access Denied: Let us know understand the scenario when a user who is not assigned a role by the administrator or is not using the account address that is assigned the roles would try to access the system functions. As seen in figure 5 below in such a scenario system would deny access to the user requesting to use to the function.

In this process when a user requests to perform certain functions the system checks *Roles List* to determine that should the user be allowed or not to use the various functions of the system. If the user is not assigned a role for performing these functions the system deny access to the user and these functions cannot be used by the user.

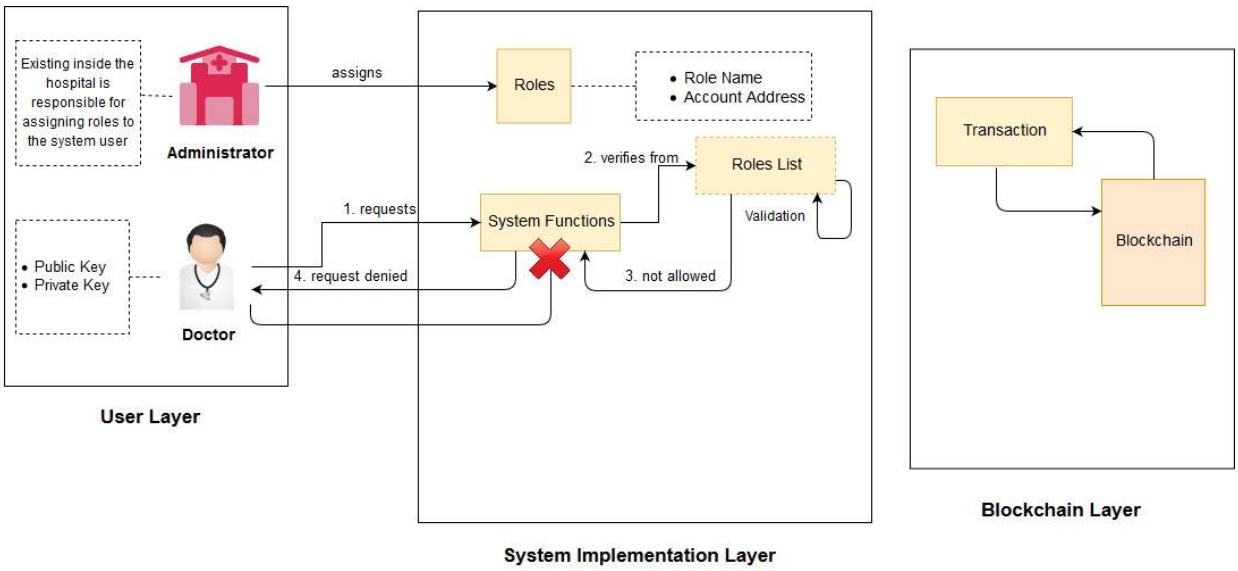

Figure 11: Usage Scenario Doctor (Access Denied)

Usage Scenario 2 – Activity Flow

The second scenario for this system would be when a patient whose medical records are stored on the system wants to view those records. The system's administrator would assign a role to the patient, containing the role name and an account address. The patient using that account requests to use the system function of *View Records*. As the patient is not allowed to add, update and delete his medical records but is only allowed to view his own medical records.

When the patient requests to use the view function the system verifies from the *Role List* and after validation is done and it results in success the system allows the patient to view his medical records. The system fetches the information from the *Blockchain* that would perform transactions for that information. Once the transaction is confirmed the patient can view this information on the DApp browser on which the whole proposed system is being visible. The figure 6 below depicts this whole scenario.

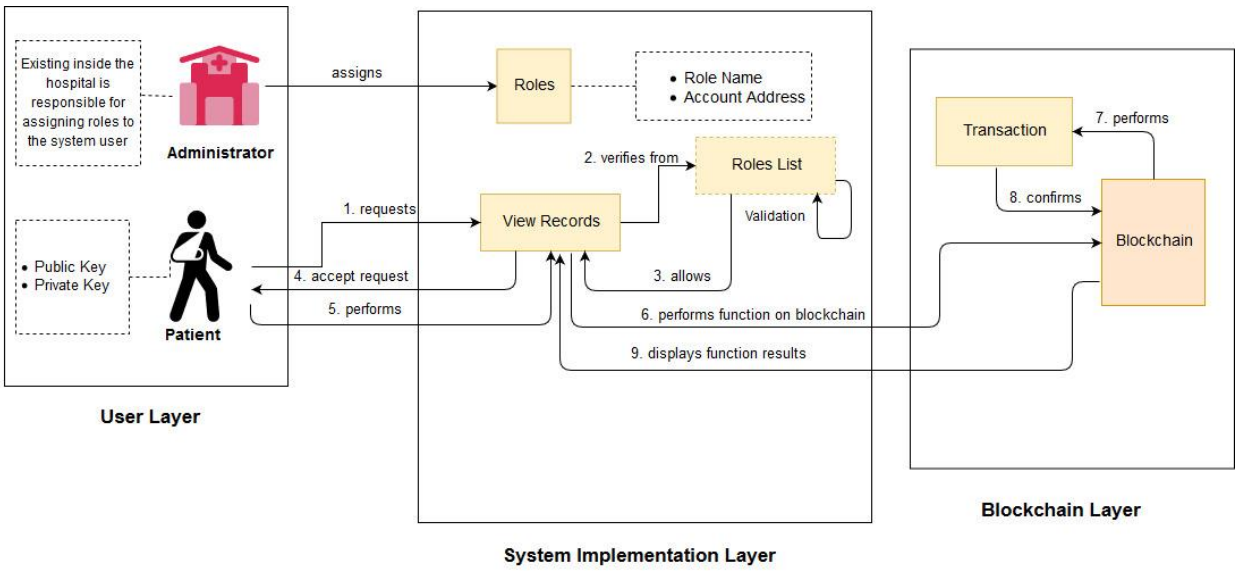

Figure 12: Usage Scenario Patient

The above section explains the usage process of various scenarios that arises during the usage of the decentralized applications. The next section explains the decentralized system's modular state. In short the functionalities provided by the system how these are incorporated while the implementation of the system is explained in this section.

Table 13: Proposed Scheme Notation

Assigned Notation	Description
D	Doctor
P	Patient
RN	Role Name
RADD	Role Address
R	Medical Records of patient
BC	Blockchain
AR	Assign Roles
RL	Roles List
P_{id}	Patient Identification

A. Assign Roles:

This phase of the DApp is handled by an administrative entity, which is trusted by the hospital and is assigned with the job of defining the roles of a user on the system. The AR sends (RN, RADD) to the BC. These both act as the unique identification of a user on the system. Just like the system has two different users on the system, the roles would also be of two kinds i.e. AR might assign roles to D or AR could also assign roles to P. The AR sends (RN, RADD) to BC that is stored in RL for P and D that are separate to both of these entities.

B. Add Patient Records:

At this stage, D is assigned the role to perform various functions on DApp. When requesting the BC to add records R on the BC,

1. The doctor D sends RN, RADDR, R to the system
2. The system first would verify and validate the RN, RADDR from RL that are stored on BC. After ensuring that sender is the doctor D who is allowed to perform the Add operation.
3. If the check results as success, the record R is stored on BC.

C. Update Patient Records:

When doctor D requests to update records R on the BC. The system would follow the steps below to perform this function,

1. The doctor D sends RN, RADDR, P_{id} to the system
2. The system first would verify and validate the RN, RADDR from RL that are stored on BC. After ensuring that sender is the doctor D who is allowed to perform the Update operation.
3. After validation step is done, the system now matches the P_{id} with the record R stored in BC.
4. After validation is done, the record R is updated on BC.

D. View Patient Records:

When doctor D requests to view/ retrieve record R from the BC. The system would follow the steps below to perform this function,

1. The doctor D sends RN, RADDR, P_{id} to the system
2. The system first would verify and validate the RN, RADDR from RL that are stored on BC. After ensuring that sender is the doctor D who is allowed to perform the View/ Retrieve operation.
3. After validation at step 2 is done, the system now matches the P_{id} with the record R stored in BC.
4. After validation at step 3 is done, the record R is available on the user's system.

E. Delete Patient Records:

When doctor D requests to delete record R from the BC. The system would follow the steps below to perform this function,

1. The doctor D sends RN, RADDR, P_{id} to the system

2. The system first would verify and validate the RN, RADDR from RL that are stored on BC. After ensuring that sender is the doctor D who is allowed to perform the delete operation.
3. After validation at step 2 is done, the system now matches the P_{id} with the record R stored in BC.
4. After validation at step 3 is done, the record R is deleted from BC.

F. View Records (For Patients):

This function is for patient P it was mentioned in our proposed framework that patient is only allowed to view his medical records R fetched from BC. The patient should not be allowed to Add, Update or Delete his medical records as this is the right and duty of the doctor to perform these tasks.

The patient uses RADD assigned to him to perform the view function on his medical records R. The system would follow the steps below to perform this function,

1. The patient P sends RN, RADDR, P_{id} to the system
2. The system first would verify and validate the RN, RADDR from RL that are stored on BC. After ensuring that sender is the patient P who is allowed to perform the View/ Retrieve operation.
3. After validation at step 2 is done, the system now matches the P_{id} with the record R stored in BC.
4. After validation at step 3 is done, the record R is available on the user's system.

CHAPTER 5

TESTING & PERFORMANCE

5.1. Testing

For testing the performance of the proposed framework we will go through this step by step guide of running its implementation. This would help to get an insight about the performance of the proposed framework while it is implemented in a test environment. These steps are explained in detail as follows.

Step 1:

The first step while testing the proposed framework or the actual system is to confirm and check if the device on which the system is being run has node.js installed in it. Another important package that the computer must have installed is npm i.e. node package manager. This would be used for running the system on the localhost of the tester's computer. Following is the list of the software requirements that is needed for testing this system.

- Node.js
- NPM
- Truffle
- Ganache (CLI or GUI)
- Git
- MetaMask

Step 2:

The next step after confirming the presence of above mentioned software or packages on the tester's device is to start preparing the proposed system for implementation. For this following command should be entered in the Git console which is opened from the root folder of the project containing the implementation of proposed framework. The command is as follows:

```
truffle compile
```

This command would compile the project and would mention if there are any errors faced while compiling it. It would check the smart contracts for any possible errors which could be syntax errors etc. This is a crucial step which should be included in the testing process as problems would arise later on while running the project.

Step 3:

This is the step where the project is migrated to the Ganache (CLI or GUI) for deployment of smart contracts using the accounts provided by it for development purposes. This is done by using following command,

```
truffle migrate –reset
```

This would allow us to migrate or push the smart contracts to the Ethereum blockchain using Ganache. As mentioned earlier Ganache is a test blockchain used by developers for deployment and testing the DApps. But an important file that our project must hold for deployment of smart contracts is 'migration' file. This file is existing with every smart contract included in the project and is also checked while the compilation is being done.

The 'reset' is a flag used with the above mentioned command for migration of all the contracts from the start this would push all the contracts to be deployed. The result of this command is visible on the Git console screen where the contract deployed information is displayed. It includes mainly the account used for deployment, the account address, the gas price, gas limit, and total number of blocks added to Ganache.

Step 4:

After migration the next step would not really be necessary but it should be done to avoid any problems in the later steps. This step is to open the Ganache GUI or UI and look up the blocks added in it after migration is completed. If the migration was successful then the blocks would have been added in the Ganache and it would be visible to the tester on the Ganache.

If the migration was displayed as successful on the previous step but the blocks are not added in the Ganache then you need to check the network requirements or truffle.config file to look for any possible problems in the configuration settings of the project.

Step 5:

The next step is to make the project to actually start running on the system's browser. For this open the Git console in the 'client' folder of the project and type the following command,

```
npm start
```

This would start the localhost server of your device and would open the project on your browser. The client folder contains the source files i.e. the react code containing the UI elements necessary for user interaction with the system's functionalities.

Step 6:

After the project is up and running on the localhost server and the UI is visible on the browser screen the next step is to open your MetaMask extension. When the project is running on the browser the MetaMask would prompt the user / tester to connect the project to the MetaMask.

Step 7:

Now that the project is completely functional we would test the various operations that could be performed. The first task would be to assign roles to the users of the system which would be doctor and patient in our proposed framework. These tasks are explained as follows:

Step 7.1:

To *assign roles* is the first task that would be done while testing this implementation. The tester would use one account imported on the MetaMask to assign roles to user. This account would be the administrator account and would be responsible for assigning roles. While assigning roles the account address would also be included along with the role name.

Step 7.2:

The account address assigned to the doctor would be used for performing the *Add patient records* operation. Which is the second task that would be done while testing this implementation. The doctor would add records of the patient using the MetaMask account assigned the role of doctor.

Step 7.3:

The other task of doctor would be to update patient records, view patient records and delete these records using the patient details and by the account assigned to the doctor at the step 7.1.

Step 7.4:

The other user of the system would be a patient who would have been assigned a separate account address which is also stored along with the patient records. The task to *view records* would be done in this step where the patient could view his medical records.

Step 8:

After the testing is done and we wish to stop the project running on our localhost we can stop this by using Control + C keyboard keys as a command on the Git console to close or stop the project running.

5.2. Performance

In this section we evaluate the performance of the proposed framework. By assessing the performance we can mitigate the risks associated with this novel technology that is understandable by very few individuals.

5.2.1. Experimental Setup

For testing performance of the proposed framework we have conducted experiments by using the following configurations:

- Intel Core i7-6498DU CPU @ 2.50GHz 2.60 GHz processor
- And 8.00 GB of memory with Windows 64-bit OS (version 10)

We developed our proposed framework by using the Solidity which is programming language of Ethereum. JavaScript and Python are encapsulated in the Solidity language which is provided by the Ethereum to write code in smart contracts.

5.2.2. Data Collection for Performance Evaluation

This section explains what kind of data is used for evaluation of performance of the proposed framework. This section also discusses the metrics that are used to explain the results of this performance evaluation being conducted.

1) Transaction Data

To evaluate the performance of the proposed framework following transaction data with its details are used. For each transaction following data is collected:

- **Transaction Deployment Time (t_{x1})**
  It is defined as the time when transaction gets deployed. In Ethereum, a smart contract is deployed using the transaction so this deployment time refers to that time.

- **Transaction Completion Time (t_{x2})**
  It is defined as the time when the transaction is completed and confirmed by the blockchain which in this case is Ethereum.

2) Evaluation Metrics

The metrics used for evaluation include the execution time, latency and throughput of the proposed framework. These are explained briefly as follows:

- **Execution Time** is defined as time duration (in seconds) between the transaction confirmation and its execution in the blockchain network. Mathematically, it is (max (t_{x2}) - min (t_{x1})).
- **Throughput** refers to the amount of data that could be transferred from one location to another in a unit amount of time.

- **Latency** is known as the delay that occurs when a system component is waiting for another component of the system to respond to an action. In terms of time it could be referred as the difference of deployment and completion time of transaction.

5.2.3. Results

1) Performance Assessment

In order to understand how our proposed framework would perform in real-case scenario of various users performing different functions on the framework we conducted performance evaluation using Apache JMeter version 5.1.1 and Apache Version 2.00. Apache JMeter is a desktop performance testing tool which is used for analysis and testing of applications [82] .

Average Execution Time: The execution time increases with the number of transactions being increased. These transactions are performed for the various functions that are included in the smart contract whose algorithm is defined in Section V. When there is only one user using the system the functions *Assign Roles, Add Patient Records* and *View Patient Records* would take 18.29 sec, 1 min 48 sec and 50 sec respectively for these functions to be executed. This time would increase when 100 users are using the system simultaneously.

Throughput: Algorithm 1 explains various functions that are included in the smart contract of the proposed framework. By using JMeter we simulated number of users from 100 users to 500 users (with period of 10 to 35), who are using the system and performing its various functions. In JMeter the throughput is represented in Data/time i.e. KB/sec units. While conducting the experiments we simulated the number of users as specified above and evaluated the performance of the system. These simulations are run on the proposed framework and at the end throughput is analyzed.

The following figure shows the throughput of the proposed framework.

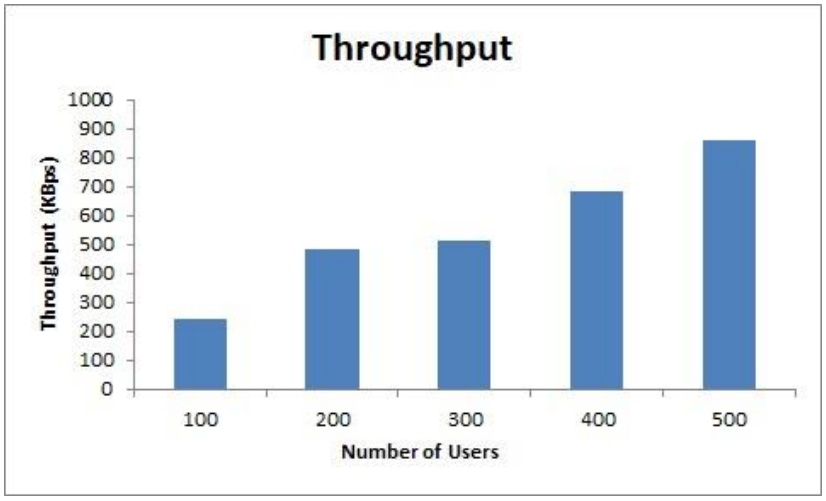

Figure 13: Throughput of the proposed framework

It is observed while conducting this experiment that as the number of users and requests increase the throughput of the system increased considerably in a linear manner. This linear increase in throughput indicates the efficiency of the proposed framework.

Average Latency: Latency as defined earlier is the delay or difference in time when one system component sends a request and a response is generated by any other system component. The difference between these two actions is defined as latency. Here we have evaluated the average latency of the proposed framework by using JMeter. While evaluating the latency of the proposed framework we simulated the number of users by JMeter. In JMeter latency is measured in terms of milliseconds.

The following graph Figure 14 gives an overview of average latency of the system along with the throughput of the proposed framework. The highest recorded latency in this experiment is 14ms.

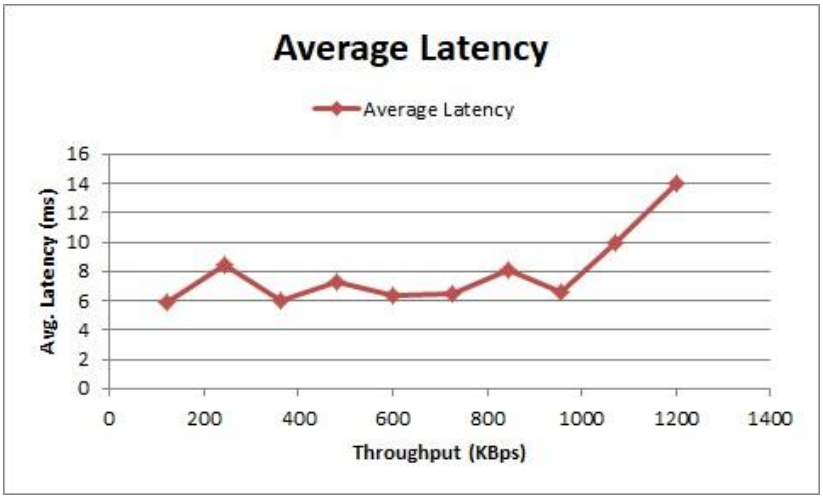

Figure 14: Average Latency of the proposed framework

We also evaluated the performance of the proposed framework by assessing the size and cost of the transaction. Before assessing the transaction size we also analyze the transaction payload. This assessment is discussed in detail in the following section.

2) PERFORMANCE EVALUATION (TRANSACTION)

Every transaction on Ethereum contains a data payload field. Data payload is included in that transaction which is meant to invoke smart contract functions. This data payload is in the hex-serialized format and has bytes associated with it. Here we would discuss two functions from Algorithm 1 in order to understand the data payload included in the transactions being generated.

Data payload is the optional field of a transaction which is only used when there is some form of interaction with contract functions. It has two important parts,

- Function Selector
- Function Arguments

The *function selector* are first 4 bytes of Keccak-256 hash, it is used for identification of the smart contract function which is being invoked. The *function arguments* include various static and dynamic element types which have different rules for encoding them in payload.

Let us now understand the payload of *Define Roles* function from Algorithm 1 to get an understanding as to how the data payload is generated. Firstly we would separate the function selector and arguments. The function selector is actually the function signature which in this case is:

```
DefineRoles (string, address)
```

For the above function the Keccak-256 hash is as follows,

```
0x6c0abd24edce8ce20a2dfb1cd2026179214468cde47681e871b6e14bf9d39efd
```

The first 4 bytes of the generated hash (0x6c0abd24) are of the function selector which points to the function being invoked from the contract. After function selector is calculated let us now understand how the function arguments are encoded.

For this we encode the head part of two arguments, the address is the static type and string is the dynamic type. The static type is passed directly while for dynamic type the offset in bytes are used, it is also measured from the start of the value encoding. The first four bytes containing the hash of function signature is not counted in it.

The encoding for dynamic type string with value “Doctor” would be as follows,

```
0x0000000000000000000000000000000000000000000000000000000000000006
```

The number of elements or bytes is 6 and they are represented as seen above. Following is the string value “Doctor” padded to 32 bytes on the right.

```
0x48976c7c7f2c20667f626c6421000000000000000000000000000000000000000000
```

The encoding of this function with only its dynamic types is as follows,

```
0x6c0abd24
  0x0000000000000000000000000000000000000000000000000000000000000006
  0x48976c7c7f2c20667f626c6421000000000000000000000000000000000000000000
```

The static and dynamic type in this function would have size of 32 bytes. The total byte size of this function is 64 bytes. In the same way we can calculate the data payload of other functions of the Algorithm 1. The following table 2 represents the data payload size of various functions.

Table 14.Data Payload of Transactions used in proposed framework

Data Payload	Contents Types	Size
Payload of F_{assign}	string, address	64 bytes
Payload of F_{add}	string, string, string, string, string, address	192 bytes
Payload of F_{view}	string, uint	64 bytes
Payload of F_{update}	string, uint, string, string, string	160 bytes
Payload of F_{delete}	string, uint	64 bytes

By using the data payload we can calculate the transaction sizes of various functions of our proposed framework. Table 3 represents the transaction sizes in bytes for these functions. We can also calculate the fee or cost associated with various transactions of the proposed framework. In Ethereum the transaction fees is calculated in 'ETH' [83] which is Ethereum coin and it has certain units such as wei, gwei associated with it. The formula to calculate Ethereum transactions fee is also discussed here. The transaction fee for a transaction is the product of gas consumed and gas price. It could be represented as follows,

$$Transaction\ Fee = gasConsumed \times gasPrice$$

We can calculate the transaction fee by using the recommended figure for gas consumed which is 21000 and is 21 Gwei for gas price. So,

$$Transaction\ Fee = 21000 \times 21 = 441000\ Gwei$$

And to calculate the transaction fee of 1ether we would use the following method,

$$1\ Ether = 1000{,}000{,}000\ Gwei$$

$$Transaction\ Fee\ for\ 1\ Ether = 441000 / 1000{,}000{,}000\ Gwei$$

$$= 0.00041\ Gwei$$

The transaction fees for various functions of Algorithm 1 are presented in table 3.

Table 15.Transactions size and fee for proposed framework

Transaction	Size	Fee
TxF_{assign}	132 bytes	0.00006 ETH
TxF_{add}	548 bytes	0.0003 ETH
TxF_{view}	122 bytes	0.0004 ETH
TxF_{update}	420 bytes	0.00001 ETH
TxF_{delete}	132 bytes	0.0003 ETH

3) COMPARISON OF PROPOSED FRAMEWORK WITH RELATED WORK

By assessing the performance of the proposed framework we can mitigate the risks associated with this novel technology that is understandable by very few individuals. In order to assess the performance of the proposed system framework we use some parameters that should be fulfilled by the system to evaluate its performance. These parameters included:

- Scalability
- Content-Addressable Storage
- Integrity
- Access Control
- Information Confidentiality

Scalability

As discussed in previous section, storing data on the blockchain creates two main problems i.e. confidentiality and scalability [33]. The data on the blockchain is visible to everyone that is present on the chain this makes the data vulnerable and is not a desired outcome for a decentralized platform. As, the purpose behind using blockchain was to get data security that was somewhat compromised in EHR systems.

The healthcare data that could be stored on the blockchain would contain patient records as data on the block and would also include current and previous block's hash. The data stored can contain the patient medical history, records, lab results, X-rays reports, MRI results and many other related results and reports, all of this voluminous data is to be stored on the blockchain that would highly affect the storage capacity of blockchain[34]. Because blockchain can store data on it but its protocol was not designed for this purpose.

Scalability in simpler terms refers to the ability of an information system to perform it functions well in such situations when the storage volume of the system increases or decreases. In case of blockchain technology scalability is an issue that needs some permanent solution. As data size or volume is increasing on the blockchain and this creates the scalability problems in the

applications functioning on the blockchain. As mentioned earlier, blockchain main purpose was to perform transactions that are temper-proof, secure and confidential. Its purpose was not to store sheer size of user data on it. So, the researchers were in a constant process of finding such a way through which the scalability issue of blockchain could be resolved.

Our proposed system used the off-chain storage mechanism for solving this issue. The patient's data stored on the blockchain contains the basic information of patient along with the IPFS hash i.e. the off-chain scaling solution used in our proposed system framework. This solves the scalability issue mentioned as now huge volume of patient medical record is not stored on the blockchain. As, the data size being stored on the blockchain has now decreased the transactions could be performed faster.

Content-Addressable storage

Content-addressable storage refers to the off-chain storage mechanism of IPFS used in the proposed framework [84]. The sensitive record of patient is stored on the IPFS, which ensures that a hash of the stored record is generated. That hash is now stored in the blockchain and is accessed when needed by the doctors and patients. The IPFS generates the cryptographically secure hash which ensures the security of the data being stored on it. And this also ensures security in our proposed framework.

Integrity

Integrity of a system is measured by the trustfulness of that system and also that system storing that information is temper-proof and reliable. This blockchain-based system ensures that it does not compromise this feature. The information stored in this system is intact and is not changed by any unauthorized channel. Moreover, this information is available to only the associated parties that are doctors and patients.

Also as mentioned in previous section, smart contracts are used for defining the functions of this system. The smart contract once published on the blockchain could not be changed by any third party. The users of the system do not have the right to make any changes in the smart contract as they are not having any access to it.

Access Control

Using the Role-based access mechanism, this framework makes sure that every entity of the system is assigned a role. This ensures that every entity only performs the role he is assigned in the system. Such as doctors should not be allowed for assigning the role to the system user but should be allowed to perform the CRUD operations on the medical records of patients. Similarly, patients should be allowed to view their records and not make any changes to them as this control is limited only to the doctors.

Using this Role-based access mechanism, any third party who is not authorized to have access to the system would not be able to access the system. This system provides a two core security as firstly blockchain technology in itself is secure and uses certain protocols and mechanism to keep itself secure from third-part intrusions. And secondly our system uses the Role-based access that also only allows the users having defined roles to have access to the system and its functions. So, our system would not only ensure security of patient records but would also make sure the access control of entities associated with it.

Information Confidentiality

The patient medical record stored on the blockchain should be secured from any third party access to ensure the confidentiality of the patients' record. The reason why the medical records should be confidential is due to the sensitive nature of that data. The patient's data include the important information of patient such as the patient medical history, blood group, records, lab results, X-rays reports, MRI results and many other related results and reports. All of this information is critical not only to patients but also to the hospital. As any information being leaked from the hospital's system would also question the security of information being stored in the hospital system.

Smart contracts are a really helpful element in this system as they ensure transparency, precision and trust on the transactions being performed. The record being stored and accessed in the system are only accessible by the trusted parties. Any untrusted third party trying to access the system is denied access by the system.

Evaluation

The following table 5 compares our proposed framework benefits and features with that of the related work [36] [45] [47]**.** The above defined features offered by our proposed framework are blockchain-based; scalability, usability and integrity are included in this comparison. These features are then compared and observed that whether they exist in the related work under consideration or not.

The first work which is compared is of Sahoo and Baruah [36], they proposed a scalable framework of blockchain using Hadoop database. The main aim of the study was to solve the scalability problem of blockchain, they proposed to use the scalability provided by the underlying Hadoop database along with the decentralization provided by the blockchain technology. The study was blockchain-based but the system or platform being used to solve this problem was not decentralized. The authors presented the framework to store blocks on the Hadoop database which was functioning on top of this framework. The rest of the system was working just like a blockchain-based application but the blocks were stored on the Hadoop database.

The second work compared is of Zhang et al. [45], proposed a scalable solution to the blockchain for clinical records. The basic aim of this study was to design such an architecture that complies with the Office of National Coordinator for Health Information Technology (ONC) requirements. They identified the barriers that this technology faces mainly include concerns related to privacy and security of blockchain, healthcare entities not sharing any trust relationship among them, scalability problems related to huge volume of datasets being transmitted on this platform, and lastly there is no universal standard enforced for data being exchanged on blockchain.

The third work compared is of Kim et al. [47], also proposed a framework for exchange of healthcare information using blockchain technology. They firstly, examined the requirements and reasons of implementing healthcare information on this technology. The architecture of this paper includes two kinds of healthcare information to be stored i.e. electronic medical records (EMR) and personal healthcare data (PHD). The authors name these as EMR-chain and PHD – chain, these two were coupled loosely to handle the different kind of information they hold. The authors identified that the aim of this study was for improvement of throughput and fairness factors of the system.

Our proposed framework uses the off-chain mechanism of IPFS protocol to resolve the issue of scalability faced by the blockchain technology. The three works does not pose a feature of usability. Moreover, our proposed framework also makes sure that blockchain features are make use of while creating the system. The features offered by this technology such as data integrity, confidentiality, transparency, decentralization, security, privacy and temper-proof metadata storage on the blockchain.

Table 16: Comparison of Proposed framework with Related Work

References	Blockchain-Based	Scalability	Content-addressable storage	Integrity	Access Control
[36]	Y	Y	N	Y	N
[45]	Y	Y	N	Y	Y
[47]	Y	N	N	Y	Y
Our proposed system	Y	Y	Y	Y	Y

CHAPTER 6

CONCLUSION & FUTURE WORK

6.1. Overview

The rapid development in technology or the technology boom has touched and affected all parts of human life. It is also changing the way we use to previously use and perceive things. Just like the changes that technology has made in various other fields of life, it is also finding new ways for improvement of healthcare sector. The main focus of these technologies being introduced in this sector is to improve the state of security, quality, user experience and many other important aspects of healthcare. And with the introduction of these technologies, there is much attention being given to make healthcare more patient-centric.

Following this technology boom many systems were developed and are widely used by some institutions for easier and efficient management of healthcare related information, the systems being used for this purpose are known as Electronic Health Record (EHR) and Electronic Medical Record (EMR) systems. EHR is a system that was designed to combine both paper-based and electronic medical records (EMR) in order to improve the quality of healthcare sector. The basic functionalities these system offers are electronic storage of medical records, patients' appointment management, billing and accounts and lab tests. Despite of all the benefits it provided these systems also faced some problems related to security, user ownership of data, data integrity etc. The solution to them could be the use of blockchain technology in healthcare sector. This technology is helpful in not only making the health care sector to be enable ownership of patient medical data but will also make it secure.

With his popular work of digital crypto-currency i.e. bitcoin Satoshi Nakamoto [15], introduced the concept of blockchain technology. Blockchain is a chain of blocks that are connected together and are continuously growing by storing transactions on the blocks. This platform uses a decentralized approach that allows the information to be distributed and that each piece of distributed information or commonly known as data have shared ownership.

A blockchain offers many benefits that mainly include security, anonymity, and integrity of data with no third party intervention. These benefits make it a reasonable choice to store patient's medical records on it, because the innovation of technology in the healthcare industry has made the security of patient's medical data a top priority. Using this technology makes it feasible for patient's records to be encrypted and accessible across different platforms with the consent of patient. A number of researchers have also identified that using blockchain technology in healthcare would be a feasible solution [10] [16] [17].

The challenges that EHR systems faced were addressed by the Blockchain technology. With blockchain technology the information is not centralized and it does not have one medium that controls the whole network. Using this technology it can be ensured to provide granular level access to all the participating entities i.e. doctors, patients, nursing staff, care providers, pharmaceutical companies and insurance companies.

The aim of this thesis endeavor was to create such a framework that would ultimately improve healthcare sector by storage and management of medical records on the blockchain. We intended to create such a decentralized platform that could store patient's medical records and give access of those records to providers or concerned individuals i.e. patient. In our research work we also focused to solve the scalability problem of blockchain, as it was not designed to store huge volumes of data on it. So, we proposed to use off-chain scaling method that makes use of the underlying medium to solve the scalability problem by storing the data on that medium. For this we made use of InterPlanatery File System (IPFS) a distributed file system platform that could store the patients' records using a peer-to-peer network and would also help in solving the scalability problem as now the patient's record would not be stored on the main blockchain but on the IPFS. So, the contribution of this proposed framework is two cores as it is used for storing medical records on the blockchain and solving scalability issue of blockchain.

6.2. Conclusion

In this research work we discussed how blockchain technology could be useful for healthcare sector and how can it be used for electronic health records. Despite the advancement in healthcare sector and technological innovation in EHR systems they still faced some issues that were addressed by this novel technology i.e. blockchain. The issues faced by EHR systems were also discussed in great detail in order to give an insight as to why there is a need of blockchain technology based solution in the healthcare sector.

We made use of Ethereum and its dependencies for implementation of the proposed framework. Ethereum is a distributed blockchain network, the idea behind it was to create a trustless smart contract platform that would be open-source and would also hold the feature of programmable blockchain.

Our proposed framework is a combination of secure record storage along with the granular access rules for those records. It creates such a system that is easier for the users to use and understand. Also, the framework proposes measures to ensure the system tackles the problem of data storage as it utilizes the off-chain storage mechanism of IPFS. And the role-based access also benefits the system as the medical records are only available to the trusted and related individuals. This also solves the problem of information asymmetry of EHR system.

The benefits provided by this proposed framework include scalability, usability, integrity, and access control and information confidentiality. These benefits could be considered as a great aspect of this framework as it would help in providing an efficient access to the medical records

of the patients. Moreover, our proposed framework also makes use the off-chain mechanism of IPFS protocol to resolve the issue of scalability faced by the blockchain technology.

6.3. Future Work

The current proposed technology includes the efficient and scalable solution to storage of the medical records. The functionalities that could be performed by using this framework include the storage, access, update, and deletion of the medical records. This is provided for both the physicians or doctors, and patients according to the level of access needed by them in any scenario. The payment module as already mentioned is not included in the existing framework. The patient after getting consultations from the doctor should be able to pay for that consultation and that too by using this proposed framework.

For the future we plan to implement the payment module in the existing framework. For this we need to have certain considerations as we need to decide how much a patient would pay for consultation by the doctor on this decentralized system functioning on the blockchain. For this we would need to define certain policies and rules that comply with the principles of the healthcare sector. These policies should be made in compliance to the existing standards of healthcare.

Appendix A

Patient Records Smart Contract Code

```
pragma solidity 0.5.0;
import "openzeppelin-solidity/contracts/access/Roles.sol";

contract PatientRecords{
	using Roles for Roles.Role;

	struct Record{
		uint id;
		string name;
		string comorbids;
		string bloodGrp;
		string ipfsHash;
	}

	mapping(uint => Record) public records;
	uint public count;

	mapping (string => Roles.Role) private roles;

	function defineRoles(string memory newRole, address newaccount ) public {

		roles[newRole].add(newaccount);

	}

	function addPatientRecord(string memory roleName,string memory _name, string memory
_comorbids, string memory _bloodGrp, string memory _ipfsHash)
	public {
		require(roles[roleName].has(msg.sender));
			count++;
			records[count] = Record(count, _name, _comorbids, _bloodGrp,_ipfsHash);

	}

	function viewPatientRecord(uint id)
	public view returns(uint, string memory, string memory, string memory, string memory) {
		return(id,records[id].name, records[id].comorbids, records[id].bloodGrp,
records[id].ipfsHash);
	}

	function updatePatientRecords(string memory roleName ,uint _id, string memory _name, string
memory _comorbids, string memory _bloodGrp)
```

```
        public returns(bool){
                require(roles[roleName].has(msg.sender));

                if( compareStrings(_name, records[_id].name) ) {

                        records[_id].name = _name;
                        records[_id].comorbids = _comorbids;
                        records[_id].bloodGrp = _bloodGrp;
                        return true;
                }
                else{
                        return false;
                }
        }

        function compareStrings (string memory a, string memory b) public view
        returns (bool) {
                return keccak256 (abi.encodePacked(a)) == keccak256 (abi.encodePacked(b));
        }
}
```

Role Smart Contract - Library

```
pragma solidity ^0.5.0;

/**
 * @title Roles
 * @dev Library for managing addresses assigned to a Role.
 */
library Roles {
   struct Role {
      mapping (address => bool) bearer;
   }

   /**
    * @dev give an account access to this role
    */
   function add(Role storage role, address account) internal {
      require(account != address(0));
      require(!has(role, account));

      role.bearer[account] = true;
   }

   /**
    * @dev remove an account's access to this role
    */
```

```
    function remove(Role storage role, address account) internal {
        require(account != address(0));
        require(has(role, account));

        role.bearer[account] = false;
    }

    /**
     * @dev check if an account has this role
     * @return bool
     */
    function has(Role storage role, address account) internal view returns (bool) {
        require(account != address(0));
        return role.bearer[account];
    }
}
```

Migration Code:

```
const Records = artifacts.require("./PatientRecords.sol");

module.exports = function(deployer){
        deployer.deploy(Records);
};
```

JavaScript Code:

App.js Code

```
import React from "react";
import Main from "./Main";
import Header from "./Header";

import "./App.css";

const App = () =>(

    <div>
      <Main />
     </div>

)

export default App;
```

Home.js

```
import React from "react";

const Home = () => (
  <div>
    <h1>Welcome</h1>
  </div>
)

export default Home;
```

Main.js

```
import React from "react";
import { Switch, Route } from "react-router-dom";
import { NavLink, HashRouter} from "react-router-dom";
import IPFSUpload from "./IPFSUpload";
import PatientRecords from "./PatientRecords";
import Home from "./Home";
import "./index.css"
//import {Provider} from "react-redux";
//import {createStore} from "redux";

//const store = createStore();

class Main extends React.Component{
        render(){
                return(
                        <HashRouter>
                        <div>
                                <h1>Blockchain Based Healthcare Records Saving System</h1>
                                <ul className = "header">
                                        <li><NavLink exact to="/">Home </NavLink> </li>
                                        <li><NavLink exact to="/patientRecords">Patient
Records</NavLink></li>
                                        <li><NavLink exact to="/ipfsUpload">Upload
Records</NavLink></li>
                                </ul>
                                <div className = "content">
                                        <Route exact path='/' component={Home}/>
                                    <Route exact path='/patientRecords' component={PatientRecords}/>
                                    <Route exact path='/ipfsUpload' component={IPFSUpload}/>
                                </div>
                        </div>
                        </HashRouter>
                );
        }
}
```

```
export default Main;
```

Header.js

```
import React from "react";
import { Link } from "react-router-dom";
import "./main.css";


const Header = () => (

        <header>
                <nav>
                        <ul>
                                <li><Link to='/'>Home</Link></li>
                                <li><Link to='/patientRecords'>Patient Records</Link></li>
                                <li><Link to='/ipfsUpload'>Upload Records</Link></li>
                        </ul>
                </nav>
        </header>

        )

export default Header;
```

PatientRecords.js

```
import React from "react";
import getWeb3 from "./utils/getWeb3";
import ipfs from "./ipfs.js";
import { withStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import "./App.css";

class PatientRecords extends React.Component {

        constructor(props){
   super(props)
   this.state = {
     web3: null, accounts: '0x0', contract: null, web3Provider: null,
     patientRoleName: "", patientAddr: 0,
```

```
    pID: 0,ipfsHash: null, redirect: false,
    eoa: "", eID: 0, eroleName: "",
    vPID: 0,vroleName: "",
    uPID: 0, upname: "", ucomorbids: "", ubloodGrp: "",
    pname: "",comorbids: "", bloodGrp: "",
    userrole: "", roleUser: "", roleName: "", addr: 0,
    tableID: "", tablepatientname: "", tablecomorbids: "", tablebloodGrp: "",
    encryptedData: "", pk: 0, payld: "",
    delRoleName: "", delpID: 0,
    showing1: false, showing2: false, showing3: false,
    showing4: false, showing5: false, showRoles: false
   };

   this.handleChange = this.handleChange.bind(this);
   this.handleSubmit = this.handleSubmit.bind(this);
   this.handleSubmit1 = this.handleSubmit1.bind(this);
   this.toggleBox1 = this.toggleBox1.bind(this);
   this.toggleBox2 = this.toggleBox2.bind(this);
   this.toggleBox3 = this.toggleBox3.bind(this);
   this.toggleBox4 = this.toggleBox4.bind(this);
   this.toggleBox5 = this.toggleBox5.bind(this);
   this.toggleBox6 = this.toggleBox6.bind(this);
   this.handleAdd = this.handleAdd.bind(this);
   this.handleGet = this.handleGet.bind(this);
   this.handleDelete = this.handleDelete.bind(this);
   this.handleUpdate = this.handleUpdate.bind(this);
   this.handleGet2 = this.handleGet2.bind(this);

 }

 componentDidMount = async () => {
   try{
     const web3 = await getWeb3();
     const accounts = await web3.eth.getAccounts((err,acc) => {
       this.state.accounts = acc[0];
     });


     const myContract = web3.eth.contract ([{
"constant": true,
     "inputs": [],
     "name": "count",
     "outputs": [
       {
         "name": "",
         "type": "uint256"
```

```
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function",
  "signature": "0x06661abd"
},
{
  "constant": true,
  "inputs": [
    {
      "name": "",
      "type": "uint256"
    }
  ],
  "name": "records",
  "outputs": [
    {
      "name": "id",
      "type": "uint256"
    },
    {
      "name": "name",
      "type": "string"
    },
    {
      "name": "comorbids",
      "type": "string"
    },
    {
      "name": "bloodGrp",
      "type": "string"
    },
    {
      "name": "ipfsHash",
      "type": "string"
    },
    {
      "name": "EOA",
      "type": "address"
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function",
  "signature": "0x34461067"
```

```
},
{
  "constant": false,
  "inputs": [
    {
      "name": "newRole",
      "type": "string"
    },
    {
      "name": "newaccount",
      "type": "address"
    }
  ],
  "name": "defineRoles",
  "outputs": [],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function",
  "signature": "0x6c0abd24"
},
{
  "constant": false,
  "inputs": [
    {
      "name": "newRole",
      "type": "string"
    },
    {
      "name": "newaccount",
      "type": "address"
    }
  ],
  "name": "definePatientRoles",
  "outputs": [],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function",
  "signature": "0xd0ffce11"
},
{
  "constant": false,
  "inputs": [
    {
      "name": "roleName",
      "type": "string"
    },
```

```
    {
      "name": "_name",
      "type": "string"
    },
    {
      "name": "_comorbids",
      "type": "string"
    },
    {
      "name": "_bloodGrp",
      "type": "string"
    },
    {
      "name": "_ipfsHash",
      "type": "string"
    },
    {
      "name": "_eoa",
      "type": "address"
    }
  ],
  "name": "addPatientRecord",
  "outputs": [],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function",
  "signature": "0xebd7c6e3"
},
{
  "constant": true,
  "inputs": [
    {
      "name": "roleName",
      "type": "string"
    },
    {
      "name": "id",
      "type": "uint256"
    }
  ],
  "name": "viewPatientRecord",
  "outputs": [
    {
      "name": "",
      "type": "uint256"
    },
```

```
    {
      "name": "",
      "type": "string"
    },
    {
      "name": "",
      "type": "string"
    },
    {
      "name": "",
      "type": "string"
    },
    {
      "name": "",
      "type": "string"
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function",
  "signature": "0x9eef0368"
},
{
  "constant": true,
  "inputs": [
    {
      "name": "roleName",
      "type": "string"
    },
    {
      "name": "_id",
      "type": "uint256"
    }
  ],
  "name": "viewRecord",
  "outputs": [
    {
      "name": "",
      "type": "uint256"
    },
    {
      "name": "",
      "type": "string"
    },
    {
      "name": "",
```

```
      "type": "string"
    },
    {
      "name": "",
      "type": "string"
    },
    {
      "name": "",
      "type": "string"
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function",
  "signature": "0x8ae41a28"
},
{
  "constant": false,
  "inputs": [
    {
      "name": "roleName",
      "type": "string"
    },
    {
      "name": "_id",
      "type": "uint256"
    },
    {
      "name": "_name",
      "type": "string"
    },
    {
      "name": "_comorbids",
      "type": "string"
    },
    {
      "name": "_bloodGrp",
      "type": "string"
    }
  ],
  "name": "updatePatientRecords",
  "outputs": [
    {
      "name": "",
      "type": "bool"
    }
```

```
  ],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function",
  "signature": "0x3e1f7cc4"
},
{
  "constant": false,
  "inputs": [
    {
      "name": "roleName",
      "type": "string"
    },
    {
      "name": "_id",
      "type": "uint256"
    }
  ],
  "name": "deletePatientRecords",
  "outputs": [
    {
      "name": "",
      "type": "bool"
    }
  ],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function",
  "signature": "0x6206de17"
},
{
  "constant": true,
  "inputs": [
    {
      "name": "a",
      "type": "string"
    },
    {
      "name": "b",
      "type": "string"
    }
  ],
  "name": "compareStrings",
  "outputs": [
    {
      "name": "",
```

```
        "type": "bool"
      }
    ],
    "payable": false,
    "stateMutability": "view",
    "type": "function",
    "signature": "0xbed34bba"
     }],function(error,result){
    if(!error){
     console.log(JSON.stringify(result));
    }
    else{
     console.error(error);
    }
   });
    const instance = myContract.at('0x4E0E259A405BFef04dA153D2c41Dd4293018eb84');


      this.setState({web3,accounts,contract:instance});
   }
   catch(error){
    alert(
       `Failed to load web3, accounts, or contract. Check console for details.`,
     );
    console.error(error);
   }

}



 handleChange(event,key)
 {
  this.setState({[key]: event.target.value});


 }

//HAVE TO GET VALUES IN A FIELD
 handleGet (event){

  const { accounts, contract} = this.state;


  const response = contract.viewPatientRecord(this.state.vroleName, this.state.vPID,
    {from: accounts}, function(error,result){
```

```
            console.log(JSON.stringify(result));
            alert('Patient Records: ' + JSON.stringify(result));
            const FileSaver = require('file-saver');
            var blob = new Blob([JSON.stringify(result)],{type: "text/plain;charset=utf-8"});
            FileSaver.saveAs(blob,"Records.txt");
          });

   console.log(response);

   };


  toggleBox1(){
   const {showing1} = this.state;
   this.setState ({showing1: !showing1});
  }
  toggleBox2(){
   const {showing2} = this.state;
   this.setState ({showing2: !showing2});
  }
  toggleBox3(){
   const {showing3} = this.state;
   this.setState ({showing3: !showing3});
  }
  toggleBox4(){
   const {showing4} = this.state;
   this.setState ({showing4: !showing4});
  }
  toggleBox5(){
   const {showing5} = this.state;
   this.setState ({showing5: !showing5});
  }
  toggleBox6(){
   const {showRoles} = this.state;
   this.setState ({showRoles: !showRoles});
  }

handleAdd (event){

   const { accounts, contract} = this.state;

     alert('A user was submitted: '+ this.state.userrole+ this.state.pname + this.state.comorbids+
       this.state.bloodGrp  + '    Account:' +accounts );
     contract.addPatientRecord(this.state.userrole,this.state.pname,
       this.state.comorbids, this.state.bloodGrp, this.state.ipfsHash,this.state.eoa,
```

```
    {from: accounts},
    function(error,result){
       if (!error) {
          console.log(JSON.stringify(result));
       } else
          console.log(error);
       });

  };


  // Doctor Role assignment
handleSubmit (event){

  const { accounts, contract} = this.state;


   alert('A role was defined: '+ this.state.rolename + this.state.addr);
    contract.defineRoles(this.state.rolename, this.state.addr,
     {from: '0xc094e7d63CfA7c940163f1EB102A6Bca5f57D0c5'},function(error,result){
       if (!error) {
          console.log(JSON.stringify(result));
       } else
          console.log(error);
       });

  };

   // Patient Role assignment
handleSubmit1 (event){

  const { accounts, contract} = this.state;


   alert('A role was defined: '+ this.state.patientRoleName + this.state.patientAddr);
    contract.definePatientRoles(this.state.patientRoleName, this.state.patientAddr,
     {from: '0xc094e7d63CfA7c940163f1EB102A6Bca5f57D0c5'},function(error,result){
       if (!error) {
          console.log(JSON.stringify(result));
       } else
          console.log(error);
       });

  };

handleUpdate (event){
```

```
    const { accounts, contract} = this.state;


      alert('A record was updated: '+ this.state.roleUser + this.state.uPID +this.state.upname +
this.state.ucomorbids+this.state.ubloodGrp);
       contract.updatePatientRecords(this.state.roleUser,this.state.uPID, this.state.upname,
this.state.ucomorbids, this.state.ubloodGrp,
        {from: accounts},function(error,result){
           if (!error) {
               console.log(JSON.stringify(result));
           } else
               console.log(error);
           });
    // alert('Records Updated');
    };

    handleDelete(event){
      const {accounts, contract} = this.state;
     const response =  contract.deletePatientRecords(this.state.delRoleName,this.state.delpID,
        {from: accounts}, function(error,result){
          if (!error) {
               console.log(JSON.stringify(result));
           } else
               console.log(error);
             });
      console.log(response);
      alert('Patient ID: '+ this.state.delpID + ' record was deleted successfully!');
    }


   //Turns the file submitted into a buffer
  captureFile = (event) => {
    event.stopPropagation();
    event.preventDefault();
    const file = event.target.files[0];
    let reader = new window.FileReader();
    reader.readAsArrayBuffer(file);
    reader.onloadend = () => this.convertToBuffer(reader);
  };

  //Helper function for turning a file into a buffer
  convertToBuffer = async(reader) => {
    const buffer = await Buffer.from(reader.result);
    this.setState({buffer});
  };
```

```
  //function for sending the buffer to ipfs node
  //and shows the ipfs hash onto the UI
  onIPFSSubmit = async(event) => {
    event.preventDefault();
    await ipfs.add(this.state.buffer, (err,ipfsHash) => {
      console.log(err,ipfsHash);
      this.setState({ipfsHash:ipfsHash[0].hash})
    })
  };

/*
  handleSave(event,value){
    const FileSaver = require('file-saver');

    var blob = new Blob([this.state.encryptedData],{type: "text/plain;charset=utf-8"});
    FileSaver.saveAs(blob,"Records.txt");
  } */

  //Encrypted records storage off-chain
  handleGet2 (event){

     const { accounts, contract} = this.state;

    const response = contract.viewRecord(this.state.eroleName,this.state.eID,
      {from: accounts}, function(error,result){
          console.log(JSON.stringify(result));
          console.log(JSON.stringify(response));
          const FileSaver = require('file-saver');
          var blob = new Blob([result],{type: "text/plain;charset=utf-8"});
          FileSaver.saveAs(blob,"Your Records.txt");
           });
  }


render() {
    const { showing1, showing2, showing3, showing4, showing5, showRoles } = this.state;
   // const {tableID,tablepatientname,tablecomorbids,tablebloodGrp} = this.state;
    return (
      <div className="App">
<button onClick={this.toggleBox6}>Assign Roles</button>

<button onClick={this.toggleBox5}>View Records</button>

<button onClick={this.toggleBox1}>Add Patient Records</button>
```

```
<button onClick={this.toggleBox2}>View Patient Records</button>

<button onClick={this.toggleBox3}>Update Patient Records</button>

<button onClick={this.toggleBox4}>Delete Patient Records</button>

{ showRoles &&

   <div style={{ display: (showRoles ? 'block' : 'none') }}>

      <h1>Assign Doctor Roles</h1>
      <hr/>
      <br/>


       <label htmlFor="name">Role Name   </label>         
    
       <input id="rolename" type="text" value = {this.state.rolename}
         onChange={event => this.handleChange(event,'rolename')}
        />

       <br/>
       <br/>

       <label htmlFor="name" >Account Address</label>    
       <input id="addr" type="text"
         value = {this.state.addr}
         onChange = {event => this.handleChange(event,'addr')}
       />

       <br/>
       <br/>

       <button id="btn1" onClick = {this.handleSubmit}>Assign</button>
              
       <br/>
       <br/>

       <h1>Assign Patient Roles</h1>
      <hr/>
      <br/>


       <label htmlFor="name">Role Name   </label>         
    
       <input id="rolename" type="text" value = {this.state.patientRoleName}
```

```
        onChange={event => this.handleChange(event,'patientRoleName')}
       />

       <br/>
       <br/>

       <label htmlFor="name" >Account Address</label>    
       <input id="addr" type="text"
        value = {this.state.patientAddr}
        onChange = {event => this.handleChange(event,'patientAddr')}
       />

       <br/>
       <br/>

       <button id="btn1" onClick = {this.handleSubmit1}>Assign</button>
              
       <br/>
       <br/>

       </div>
}



{ showing5 &&

   <div style={{ display: (showing5 ? 'block' : 'none') }}>
       <h2> Patient Records View</h2> <br/><br/>
       <label htmlFor="name">Your ID</label>            
                 
       <input id="name" type="text" value = {this.state.eID}
        onChange={event => this.handleChange(event,'eID')}
       />
       <br/><br/>
       <label htmlFor="name">Your Role Name</label>           
 
       <input id="name" type="text" value = {this.state.eroleName}
        onChange={event => this.handleChange(event,'eroleName')}
       />
       <br/><br/>

       <button id="btn2" onClick = {this.handleGet2}>View Records</button>
              
       <br/> <br/>
       </div>
```

```
}


{ showing1 &&

   <div style={{ display: (showing1 ? 'block' : 'none') }}>
   <hr/>
      <h3>Add records</h3>
      <label htmlFor="name">Your role</label>              
    
      <input id="name" type="text" value = {this.state.userrole}
        onChange={event => this.handleChange(event,'userrole')}
       />

      <br/>
      <br/>
      <label htmlFor="name">Patient Name</label>        
      <input id="name" type="text" value = {this.state.pname}
        onChange={event => this.handleChange(event,'pname')}
       />

      <br/>
      <br/>

      <label htmlFor="name" >Co-morbids</label>            
      <input id="comorbids" type="text"
        value = {this.state.comorbids}
        onChange = {event => this.handleChange(event,'comorbids')}
      />

      <br/>
      <br/>

      <label htmlFor="name" >Blood Group</label>         
      <input id="bGrp" type="text"
      value = {this.state.bloodGrp}
      onChange = {event => this.handleChange(event, 'bloodGrp')}
      />
      <br/>
      <br/>

      <label htmlFor="name" >Patient Public Address</label>         
      <input id="bGrp" type="text"
      value = {this.state.eoa}
      onChange = {event => this.handleChange(event, 'eoa')}
```

```
      />
      <br/>
      <br/>

      <h3>Add a file to IPFS here </h3>
      <form id="ipfs-hash-form" className="scep-form" onSubmit={this.onIPFSSubmit}>
        <input
          type="file"
          onChange={this.captureFile}
          />
          <button
            type = "submit">
            Send it
            </button>
      </form>
      <p>The IPFS hash is: {this.state.ipfsHash}</p>

      <button id="btn1" onClick = {this.handleAdd}>Save Records</button>
      <br/> <br /> </div>
}

{showing2 &&
  <div style={{ display: (showing2 ? 'block' : 'none') }}>
 <hr/>
      <h2>View Patient Records</h2>
      <p>Enter your details below to view the medical records!</p>
      <br/>
      <label htmlFor="name" >Patient ID</label>         
      <input id="pID" type="text"
      value = {this.state.vPID}
      onChange = {event => this.handleChange(event, 'vPID')}
      />
      <br/>
      <br/>
      <label htmlFor="name" >Role Name</label>        
      <input id="pID" type="text"
      value = {this.state.vroleName}
      onChange = {event => this.handleChange(event, 'vroleName')}
      />
      <br/>
      <br/>
      <button id="btn2" onClick = {this.handleGet}>View Records</button>
             
```

```
      <Table>
       <TableHead>
        <TableRow>
         <TableCell align = "right">Registration ID</TableCell>
         <TableCell align = "right">Patient Name</TableCell>
         <TableCell align = "right">Co-morbids</TableCell>
         <TableCell align = "right">Blood Group</TableCell>
        </TableRow>
       </TableHead>
       <TableBody>
       <TableRow>
         <TableCell align = "right">{this.state.tableID}</TableCell>
         <TableCell align = "right">{this.state.tablepatientname}</TableCell>
         <TableCell align = "right">{this.state.tablecomorbids}</TableCell>
         <TableCell align = "right">{this.state.tablebloodGrp}</TableCell>
       </TableRow>
       </TableBody>
      </Table>

      <br/>
      <br/> </div>
}

{showing3 &&
  <div style={{ display: (showing3 ? 'block' : 'none') }}>
      <hr/>
      <h2>Update Patient Records</h2>
      <br/>
      <label htmlFor="name">Role Name</label>        
      <input id="rolename" type="text" value = {this.state.roleUser}
        onChange={event => this.handleChange(event,'roleUser')}
       />

      <br/>
      <br/>
      <label htmlFor="name" >Patient ID</label>             
      <input id="pID" type="text"
      value = {this.state.uPID}
      onChange = {event => this.handleChange(event, 'uPID')}
      />
      <br/>
      <br/>


      <label htmlFor="name">Patient Name</label>    
      <input id="name" type="text"
```

```
        value = {this.state.upname}
          onChange={event => this.handleChange(event,'upname')}
         />

        <br/>
        <br/>

        <label htmlFor="name" >Co-morbids</label>         
        <input id="comorbids" type="text"
          value = {this.state.ucomorbids}
          onChange = {event => this.handleChange(event,'ucomorbids')}
        />

        <br/>
        <br/>

        <label htmlFor="name" >Blood Group</label>      
        <input id="bGrp" type="text"
        value = {this.state.ubloodGrp}
        onChange = {event => this.handleChange(event, 'ubloodGrp')}
        />
        <br/>
        <br/>
        <button id="btn3" onClick = {this.handleUpdate}>Update Records</button>
               </div>
}

{showing4 &&
  <div style={{ display: (showing4 ? 'block' : 'none') }}>
        <hr/>
        <h2>Delete Patient Records</h2>
        <br/>
        <label htmlFor="name">Role Name</label>        
        <input id="rolename" type="text" value = {this.state.delRoleName}
          onChange={event => this.handleChange(event,'delRoleName')}
         />

        <br/>
        <br/>
        <label htmlFor="name" >Patient ID</label>             
        <input id="pID" type="text"
        value = {this.state.delpID}
        onChange = {event => this.handleChange(event, 'delpID')}
        />
        <br/>
        <br/>
```

```
        <button id="btn4" onClick = {this.handleDelete}>Delete Records</button>
               </div>
}

        <br/>
      </div>
    );
  }

}

export default PatientRecords;
```

Bibliography

[1] B. Aldosari, “Patients’ safety in the era of EMR/EHR automation,” *Informatics Med. Unlocked*, vol. 9, no. October, pp. 230–233, 2017.

[2] Q. Gan, “Adoption of Electronic Health Record System : Multiple Theoretical Perspectives,” *2014 47th Hawaii Int. Conf. Syst. Sci.*, pp. 2716–2724, 2014.

[3] H. Wu and E. M. LaRue, “Linking the health data system in the U.S.: Challenges to the benefits,” *Int. J. Nurs. Sci.*, vol. 4, no. 4, pp. 410–417, 2017.

[4] R. Gartee, “Electronic health records: understanding and using computerized medical records,” *Electron. Heal. Rec.*, p. 550, 2011.

[5] R. S. Evans, “Electronic Health Records: Then, Now, and in the Future,” *Yearb. Med. Inform.*, vol. 25, no. S 01, pp. S48–S61, 2016.

[6] “Summary of HIPAA Security Rule,” 2013. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html.

[7] “Certified Health IT,” 2019. [Online]. Available: https://www.healthit.gov/playbook/certified-health-it/.

[8] M. Rouse, “ONC (Office of the National Coordinator for Health Information Technology),” 2017. [Online]. Available: https://searchhealthit.techtarget.com/definition/ONC.

[9] M. Reisman, “EHRs: The Challenge of Making Electronic Data Usable and Interoperable.,” *P T*, vol. 42, no. 9, pp. 572–575, 2017.

[10] W. J. Gordon and C. Catalini, “Blockchain Technology for Healthcare : Facilitating the Transition to Patient-Driven Interoperability,” *Comput. Struct. Biotechnol. J.*, vol. 16, pp. 224–230, 2018.

[11] “Why Healthcare needs the Blockchain,” *Medium Corporation*, 2018. [Online]. Available: https://medium.com/sherpa-protocol/why-healthcare-needs-the-blockchain-f5e52f35f5c2. [Accessed: 03-Nov-2019].

[12] “Healthcare Data Breach Statistics,” *HIPAA Journal*, 2018. [Online]. Available: https://www.hipaajournal.com/healthcare-data-breach-statistics/.

[13] M. Tindera, “Government Data Says Millions Of Health Records Are Breached Every Year,” 2018. [Online]. Available: https://www.forbes.com/sites/michelatindera/2018/09/25/government-data-says-millions-of-health-records-are-breached-every-year/#181263be16e6.

[14] C. S. Kruse, B. Smith, H. Vanderlinden, and A. Nealand, “Security Techniques for the Electronic Health Records,” *J. Med. Syst.*, vol. 41, no. 8, 2017.

[15] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electrnic Cash System,” pp. 1–9, 2008.

[16] A. Boonstra, A. Versluis, and J. F. J. Vos, “Implementing electronic health records in hospitals : a systematic literature review,” *BMC Health Serv. Res.*, no. September, 2014.

[17] T. D. Gunter and N. P. Terry, “The emergence of national electronic health record architectures in the United States and Australia: models, costs, and questions,” *J. Med. Internet Res.*, vol. 7, no. 1, pp. e3–e3, Mar. 2005.

[18] D. Lee Kuo Chen, *Handbook of digital currency*, 1st Editio. Elsevier, 2015.

[19] “Peer to Peer,” *Wikipedia*. Wikimedia Inc, 2019.

[20] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,” *Proc. - 2017 IEEE 6th Int. Congr. Big Data, BigData Congr. 2017*, no. June, pp. 557–564, 2017.

[21] “Mining,” 2018. [Online]. Available: https://en.bitcoin.it/wiki/Mining. [Accessed: 04-Feb-2019].

[22] T. Xue, Y. Yuan, Z. Ahmed, K. Moniz, G. Cao, and C. Wang, “Proof of Contribution : A Modification of Proof of Work to Increase Mining Efficiency,” *2018 IEEE 42nd Annu. Comput. Softw. Appl. Conf.*, pp. 636–644, 2018.

[23] S. Voshmgir and V. Kalinov, “Blockchain A Beginners guide,” 2017, p. 57.

[24] “Proof of work,” *Bitcoin Wiki*. 2019.

[25] A. Tar, “Proof of Work, Explained.” [Online]. Available: https://cointelegraph.com/explained/proof-of-work-explained.

[26] S. Ray, “What is Proof of Stake?,” 2017. [Online]. Available: https://hackernoon.com/what-is-proof-of-stake-8e0433018256.

[27] S. Ray, “What is Delegated Proof of Stake?,” 2018. [Online]. Available: https://hackernoon.com/what-is-delegated-proof-of-stake-897a2f0558f9.

[28] B. Curran, “What is Practical Byzantine Fault Tolerance? Complete Beginner’s Guide,” 2018. [Online]. Available: https://blockonomi.com/practical-byzantine-fault-tolerance/.

[29] I. M. Coelho, V. N. Coelho, P. Lin, and E. Zhang, “Chapter 8 – Delegated Byzantine Fault Tolerance : Technical details , challenges and perspectives,” 2019.

[30] B. Curran, “What is The Stellar Consensus Protocol? Complete Beginner’s Guide,” 2018. [Online]. Available: https://blockonomi.com/stellar-consensus-protocol/.

[31] V. Buterin, “On Public and Private Blockchains,” *Ethereum Blogs*, 2015. [Online]. Available: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/.

[32] “SHA-256,” *Bitcoin Wiki*, 2018. [Online]. Available: https://en.bitcoinwiki.org/wiki/SHA-256.

[33] G. Greenspan, “Scaling blockchains with off-chain data,” *MultiChain, Private*

blockchains, 2018. [Online]. Available: https://www.multichain.com/blog/2018/06/scaling-blockchains-off-chain-data/. [Accessed: 03-Dec-2019].

[34] C. Pirtle and J. Ehrenfeld, "Blockchain for Healthcare: The Next Generation of Medical Records?," *J. Med. Syst.*, vol. 42, no. 9, p. 172, Aug. 2018.

[35] A. A. Siyal, A. Z. Junejo, M. Zawish, K. Ahmed, A. Khalil, and G. Soursou, "Applications of Blockchain Technology in Medicine and Healthcare: Challenges and Future Perspectives," *Cryptography*, vol. 3, no. 1, p. 3, 2019.

[36] M. S. Sahoo and P. K. Baruah, "HBasechainDB -- A Scalable Blockchain Framework on Hadoop Ecosystem," in *Supercomputing Frontiers*, 2018, pp. 18–29.

[37] "InterPlanetary File System," *Wikipedia*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/InterPlanetary_File_System. [Accessed: 03-Dec-2019].

[38] H. Fraser, "How Blockchains Can Provide New Benefits for Healthcare," *IBM Blog*, 2017. [Online]. Available: https://www.ibm.com/blogs/think/2017/02/blockchain-healthcare/.

[39] R. Tabata, "How Blockchain Is Transforming Health Care," *Forbes Technology Council*, 2018. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2018/11/07/how-blockchain-is-transforming-health-care/#5765e4954e4d.

[40] "Costs of Storing Data on the Blockchain," *1Kosmos BlockID Blog*, 2018. [Online]. Available: https://onekosmos.com/blog/cost-of-storing-data-on-the-blockchain/.

[41] J. Eberhardt and S. Tai, *On or Off the Blockchain? Insights on Off-Chaining Computation and Data*. 2017.

[42] D. Vujičić, D. Jagodić, and S. Randić, "Blockchain technology, bitcoin, and Ethereum: A brief overview," *2018 17th Int. Symp. INFOTEH-JAHORINA, INFOTEH 2018 - Proc.*, vol. 2018-Janua, no. March, pp. 1–6, 2018.

[43] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F. Y. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends," *IEEE Intell. Veh. Symp. Proc.*, vol. 2018-June, no. Iv, pp. 108–113, 2018.

[44] T. T. Kuo, H. E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," *J. Am. Med. Informatics Assoc.*, vol. 24, no. 6, pp. 1211–1220, 2017.

[45] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "FHIRChain : Applying Blockchain to Securely and Scalably Share Clinical Data," *Comput. Struct. Biotechnol. J.*, vol. 16, pp. 267–278, 2018.

[46] S. Jiang, J. Cao, H. Wu, Y. Yang, M. Ma, and J. He, "BlocHIE : a BLOCkchain-based platform for Healthcare Information Exchange," *2018 IEEE Int. Conf. Smart Comput.*, pp.

49–56, 2018.

[47] M. G. Kim, A. R. Lee, H. J. Kwon, J. W. Kim, and I. K. Kim, “Sharing Medical Questionnaries based on Blockchain,” *Proc. - 2018 IEEE Int. Conf. Bioinforma. Biomed. BIBM 2018*, pp. 2767–2769, 2019.

[48] “What is Ethereum?,” 2018. [Online]. Available: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html#a-next-generation-blockchain.

[49] “Ethereum,” 2018. [Online]. Available: https://www.investopedia.com/terms/e/ethereum.asp.

[50] U. W. Chohan, “Cryptocurrencies : A Brief Thematic Review,” *SSRN Electron. J.*, 2017.

[51] “Account Types, Gas, and Transactions,” 2018. [Online]. Available: http://ethdocs.org/en/latest/contracts-and-transactions/account-types-gas-and-transactions.html#what-is-a-transaction.

[52] P. Kasireddy, “How does Ethereum work, anyway?,” *Medium Corporation*, 2017. [Online]. Available: https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369.

[53] “What is Ethereum? The Most Comprehensive Gudie Ever,” 2018. [Online]. Available: https://blockgeeks.com/guides/ethereum/.

[54] “Ethereum,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Ethereum.

[55] I. Karamitsos, M. Papadaki, and N. B. Al Barghuthi, “Design of the Blockchain Smart Contract: A Use Case for Real Estate,” *J. Inf. Secur.*, vol. 09, no. 03, pp. 177–190, 2018.

[56] T. Takenobu, “Ethereum EVM illustrated,” 2018.

[57] “Immutable Object,” *Wikipedia*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Immutable_object.

[58] “Immutable collections for JavaScript,” *npm*, 2019. [Online]. Available: https://www.npmjs.com/package/immutable.

[59] S. Edelkamp, “Patricia tree,” *Dictionary of Algorithms and Data Structures*. [Online]. Available: https://www.nist.gov/dads/HTML/patriciatree.html. [Accessed: 05-Aug-2019].

[60] K. Kiyun, “Modified Merkle Patricia Trie — How Ethereum saves a state,” *Medium Corporation*, 2018. [Online]. Available: https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd. [Accessed: 05-Sep-2019].

[61] “Patricia Tree,” *Github Inc.*, 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/Patricia-Tree. [Accessed: 05-Sep-2019].

[62] L. Saldanha, “Ethereum Explained: Merkle Trees, World State, Transactions, and More,”

2018. [Online]. Available: https://pegasys.tech/ethereum-explained-merkle-trees-world-state-transactions-and-more/.

[63] T. Dey, S. Jaiswal, S. Sunderkrishnan, and N. Katre, “A Medical Use Case of Internet of Things and Blockchain,” *2017 Int. Conf. Intell. Sustain. Syst.*, no. Iciss, pp. 486–491, 2017.

[64] “InterPlanatery File System (IPFS).” [Online]. Available: https://ipfs.io/. [Accessed: 04-Feb-2019].

[65] J. Benet, “IPFS - Content Addressed, Versioned, P2P File System,” no. Draft 3, 2014.

[66] “Application Layer,” *Wikipedia*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Application_layer.

[67] “IPFS vs HTTP: How distributed networks make the internet great again,” 2019. [Online]. Available: https://cryptoinsider.com/distributed-networks-internet/.

[68] “Truffle,” 2019. [Online]. Available: https://www.truffleframework.com/docs/truffle/overview.

[69] “Truffle Boxes,” 2019. [Online]. Available: https://truffleframework.com/boxes.

[70] “Truffle QuickStart,” 2019. [Online]. Available: https://www.truffleframework.com/docs/truffle/quickstart#creating-a-project.

[71] “React Main Concepts,” *Facebook Inc.*, 2019. [Online]. Available: https://reactjs.org/docs/hello-world.html.

[72] “React.Component,” *Facebook Inc.*, 2019. [Online]. Available: https://reactjs.org/docs/react-component.html#componentdidmount.

[73] K. Silas, “Working with events in react,” 2018. [Online]. Available: https://css-tricks.com/working-with-events-in-react/.

[74] “Synthetic Event,” *Facebook Inc.*, 2019. [Online]. Available: https://reactjs.org/docs/events.html.

[75] “Ganache,” 2019. [Online]. Available: https://truffleframework.com/ganache. [Accessed: 04-Mar-2019].

[76] “Creating Workspaces,” 2019. [Online]. Available: https://truffleframework.com/docs/ganache/workspaces/creating-workspaces.

[77] “MetaMask,” 2019. [Online]. Available: https://metamask.io/. [Accessed: 04-Mar-2019].

[78] “MetaMask Developer Documentation,” 2019. [Online]. Available: https://metamask.github.io/metamask-docs/.

[79] “User (Computing),” *Wikipedia*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/User_(computing).

[80] “web3.js - Ethereum JavaScript API,” 2019. [Online]. Available: https://web3js.readthedocs.io/en/1.0/. [Accessed: 04-Jul-2019].

[81] “Infura,” 2019. [Online]. Available: https://infura.io/. [Accessed: 04-Jul-2019].

[82] Niranjanamurthy, K. Kumar S, A. Saha, and D. D. Chahar, “Comparative Study on Performance Testing with JMeter,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 5, no. 2, pp. 70–76, 2016.

[83] G. Wood, “Ethereum: A Secure Decentralised Generalised Transaction Ledger. EIP-150 REVISION,” *2017*, no. August 1, 2017, p. 33, 2017.

[84] J. Eberhardt and S. Tai, “On or Off the Blockchain? Insights on Off-Chaining Computation and Data,” *Smart SOA Platforms Cloud Comput. Archit.*, no. October, pp. 11–45, 2014.