: N° d'ordre
: N° de série

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID HAMMA LAKHDAR - EL OUED

FACULTY OF EXACT SCIENCES
Computer Science Department

Thesis
submitted in partial fulfilment of the requirements for the degree of

ACADEMIC MASTER

Field: **Mathematics and Computer Science**
Option: **Computer Science**
Specialty: **Distributed Systems and Artificial Intelligence**

Presented by :

- **Belgacem Meraghni**
- **Cherif Ahmed Cherif**

Design and development of a medical application based on Blockchain technology

Defended on June 16 th 2022

Examination Commitee :

M.	Medileh Saci	MCA	Chair
M.	Beggas Mounir	MAA	Examinator
M.	Meftah Mohammed Charaf Eddine	MAA	Supervisor

Academic year: 2021-2022

Abstract

One of the most critical elements of information systems is the storage method, especially if you need to share and collect data from other individuals and entities, since you need to trust their information. Central databases are typically used as a general answer to this issue, but in order to implement this solution, all partners must have trust in the organization that maintains the database as the central authority to manage the data.
In the case of storing medical health records, the centralisation of the information poses a risk of patient's data manipulation. Blockchain is a technology that has been around for several years, this technology allows you to share information, when you need to trust data and participants without using a central database. In this project the objective is to implement Blockchain Smart Contracts to store sensitive patient data "Electronic Health Records", in a medical application, which guaranties the immutability of these records.

key-words : Blockchain, Trust, , Electronic Medical Record, access control, Smart contract, IPFS.

ملخص

تعد طريقة تخزين البيانات احد اهم العناصر في انظمة المعلومات ، خاصة اذا كنت بالحاجة الى مشاركة وجمع المعلومات من الأفراد والكيانات الأخرى ، حيث يتعين عليك الوثوق بمعلوماتهم. تُستخدم قواعد البيانات المركزية عادةً كإجابة عامة لهذه المشكلة ، ولكن لتنفيذ هذا الحل ، يجب أن يثق جميع الاطراف في المنظمة التي تحتفظ بقاعدة البيانات باعتبارها السلطة المركزية لإدارة البيانات.
في حالة تخزين سجلات صحية ، وضع المعلومات في قاعدة بيانات مركزية سيضعها في خطر التلاعب و التغيير.
البلوك شاين هي تقنية ظهرت منذ عدة سنوات ، تتيح هذه التقنية مشاركة المعلومات ، عندما تحتاج إلى الوثوق بالبيانات ومصادرها دون استخدام قاعدة بيانات مركزية.
الهدف الاساسي لهذا المشروع هوتصميم و إنشاء تطبيق يستعمل العقود الذكية للبلوكشاين لتخزين البيانات والمعلومات الحساسة للمرضى (السجلات الصحية الالكترونية) ، مما يضمن ثباتها وعدم تغيرها.

الكلمات المفتاحية :البلوك شاين ، الثقة ، الملفات الطبية الإلكترونية ، العقود الذكية ، التحكم بالمعلومات ، آي بي اف اس .

Acknowledgement

Before presenting this work, we would like to begin by thanking Almighty God, for allowing us to reach this level of studies, as well as blessing us with patience and courage.
We want to thank our dear parents, who supported us emotionally and financially to help secure our futures.
I would like to thank everyone who has helped us to complete this work and overcome all obstacles especially our professor "Mohamed Charaf Eddine" who has always given us advice and good guidance. We also extend our gratitude to the people who have given us the honour to participate in the juries of this thesis. Thank you to all those who have contributed in the accomplishment of this job.

Contents

List of Figures

List of Tables

General Introduction:

Sensitive patient data is contained in electronic health records; if a doctor makes a mistake during a consultation, the health record may be utilized to determine who is to blame. Therefore, the health record's immutability is crucial, and here is where blockchain technology comes into effect as it grantees that all the parts in the network have the same information and when the information is entered, no one can edit all the information written in it. as opposed to central databases, which permits data manipulation freely.

This work will be looking into blockchain technology by applying it in a a medical application which it going to be a good example for the advantages that the blockchain can provides in the medical field.

The paper is organized as follows:

- **The first chapter** Introduces general notions of Blockchain technology, including history, fields of application, architecture and operation.
- **The second chapter** The second chapter presents the applications of Blockchain in healthcare and the main research works of systems using Blockchain in the medical field
- **The third chapter** shows the design of the proposed system and therefore the development stages
- **The fourth chapter** presents the realization and the implementation of the system.

Finally, ending the thesis with a general conclusion.

Chapter 1

Blockchain Technology

1.1 Introduction

The Blockchain started a revolution in the computer science field, due to the advantages that it provides, mainly the transparency and the privacy, although these characteristics are usually opposing, Blockchain ensures both of them due to it's unique architecture.

Blockchain aids in the verification and trace-ability of multi-step transactions that require them. It can provide secure transactions, reduce compliance costs, and speed up data transfer processing.

This chapter presents the general notions of the Blockchain technology, it will dive into the definition and the history behind the revolution of the Blockchain, it will also recount it's characteristics, and the different models it can be implemented by, It will then focus on the development approaches of a Blockchain application. Lastly it will mention the application fields of the technology.

1.2 Blockchain Definition

To summarize the above definitions, Blockchain is a record-keeping and contract-enforcement technology that uses cryptography to make it extremely difficult to change previous history. It allows participants to share work streams by tracking changes on a shared ledger and without an intermediary.

Blockchain is a decentralized and distributed network as opposed to the traditional centralized way of storing information. the figure 1.1 [20] demonstrates the differences between a centralized and decentralized and distributed network Each participant in the network becomes a part of the database that stores the blockchain and when the information is shared it is hard to compromise. When a traditional centralized system is hacked all the data is corrupted, but to hack a blockchain database you would need to have access to over 50% of the network.

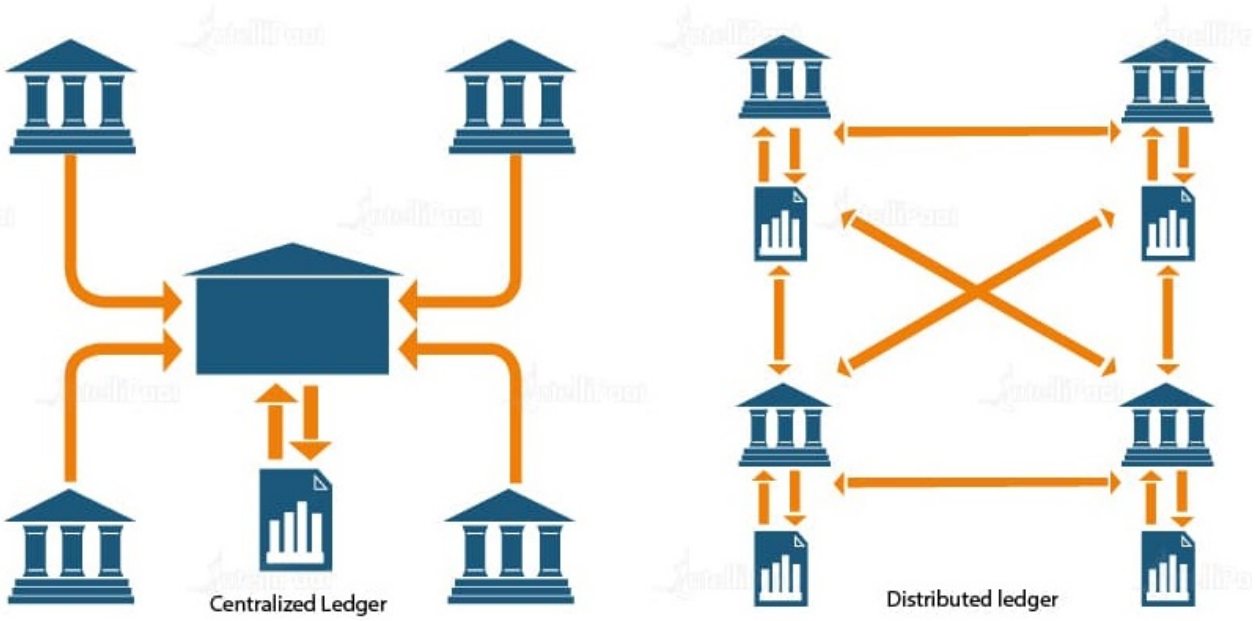

Figure 1.1: the differences between a centralized and decentralized and distributed network

1.3 Brief history of Blockchain

- **1990s:** In 1982 David Caum was the one who proposed first protocol of blockchain [23], Furthermore in 1991 Stuart Haber and W.Scott Stornetta tried to develop a cryptographically secured chain of blocks [16],their goal was to create an immutable system that could not be changed or mess with, which is now considered to be one of the main features of blockchain technologies.Later, Haber, Stornetta, and Dave Bayer added Merkle trees into the design, allowing multiple document certification. Eventually they published their work in 1995. [2]

- **2008 :** In a paper titled "Bitcoin: A Peer-To-Peer Electronic Cash System," published by a group going by the name of "Satoshi Nakamoto" in 2008, the most well-known blockchain project by the name of Bitcoin was introduced. This paper outlined the blockchain features in general and the first cryptocurrency Bitcoin specifically, which had the immutability and enabled secure transactions, transparency and directly peer to peer without any intermediary. [4]

That was the beginning of Blockchain, the figure 1.2 bellow gives an overview of the evolution of the Blockchain through the history.

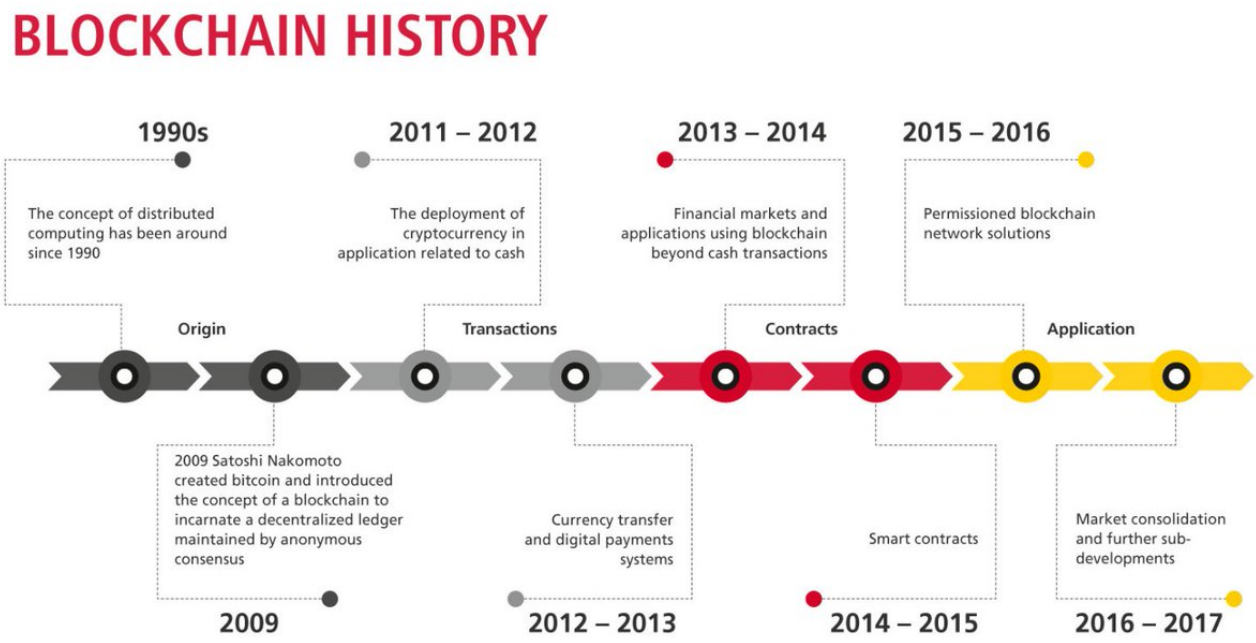

Figure 1.2: Blockchain timeline history

1.4 Blockchain characteristics

1.4.1 Smart Contracts Enabled by Blockchain

Smart contracts, enabled by the blockchain technology, are self-executing contracts without extra enforcement. The contractual clauses between nodes are converted into computer programs in a form such as "If-Then" statements. The executable computer programs are then securely stored in the blockchain. When the predefined conditions in smart contract are satisfied, the clauses in smart contracts will be executed autonomously, and the execution will be recorded as an immutable transaction in the blockchain. the next figure 1.3 give an idea about the smart contracts in Blockchain

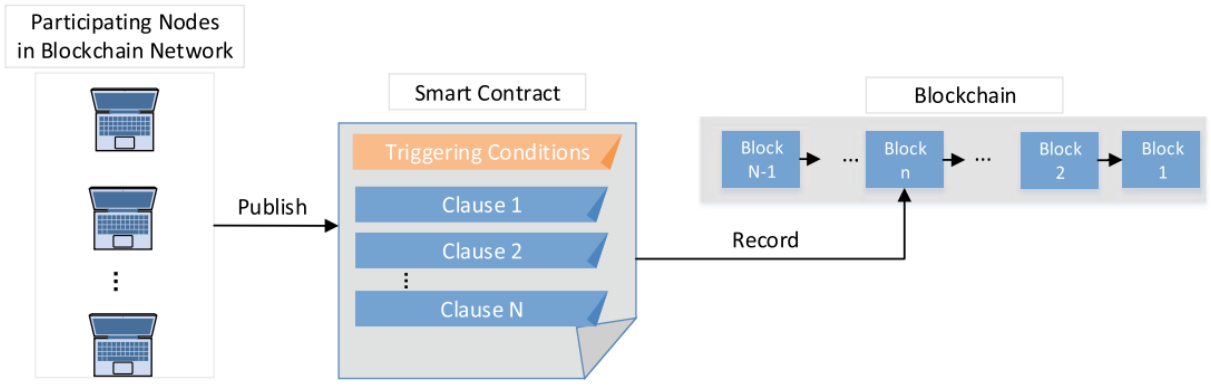

Figure 1.3: a figure represent the functionality of Smart Contracts in Blockchain

1.5 Structure of Blockchain

1.5.1 Transactions

Transactions are the main unit in a Blockchain, they represent the smallest part of it, a transaction can have different meanings depending on the application, the most popular example is the exchange of value in payment systems, in this case, it usually comprises of three different components; a recipient address, a sender address and a the value itself.

1.5.2 Blocks

Blocks in one of the most based parts in blockchain , blocks is a type of data structure where the blockchain information are permanently stored and encrypted, a block can store all the information or just the recent informations that have not been yet verified , when those information are verified this block close and new block get generated to store new information that waiting to verifie. [6]

1.5.3 Consensus

In blockchain as it knows there is no central authority to determine if a block contains a valid and verified information or not. For that, blockchain relies on consensus, which uses algorithms to achieve a consensus on the network's current state. The most common consensus algorithms are Proof of Work (PoW) and Proof of Stake (PoS).

1.5.4 Blockchain Hash

Another of the main aspects the blockchain is safe is by using cryptographic hashing rather than encryption. To be clear, hashing only functions in one way, and as of this writing, no known technique has been able to pass the hashing. Traditional encryption, on the other hand, works in two ways, allowing what is encrypted to be decrypted using the encryption key. [17] The hash is considered as type of ID for the informations in the blocks because the hash for identical data is always the same and changes if even one letter in the data changes. [12]. [17] The next figure display the difference between hash and traditional cryptography 1.4

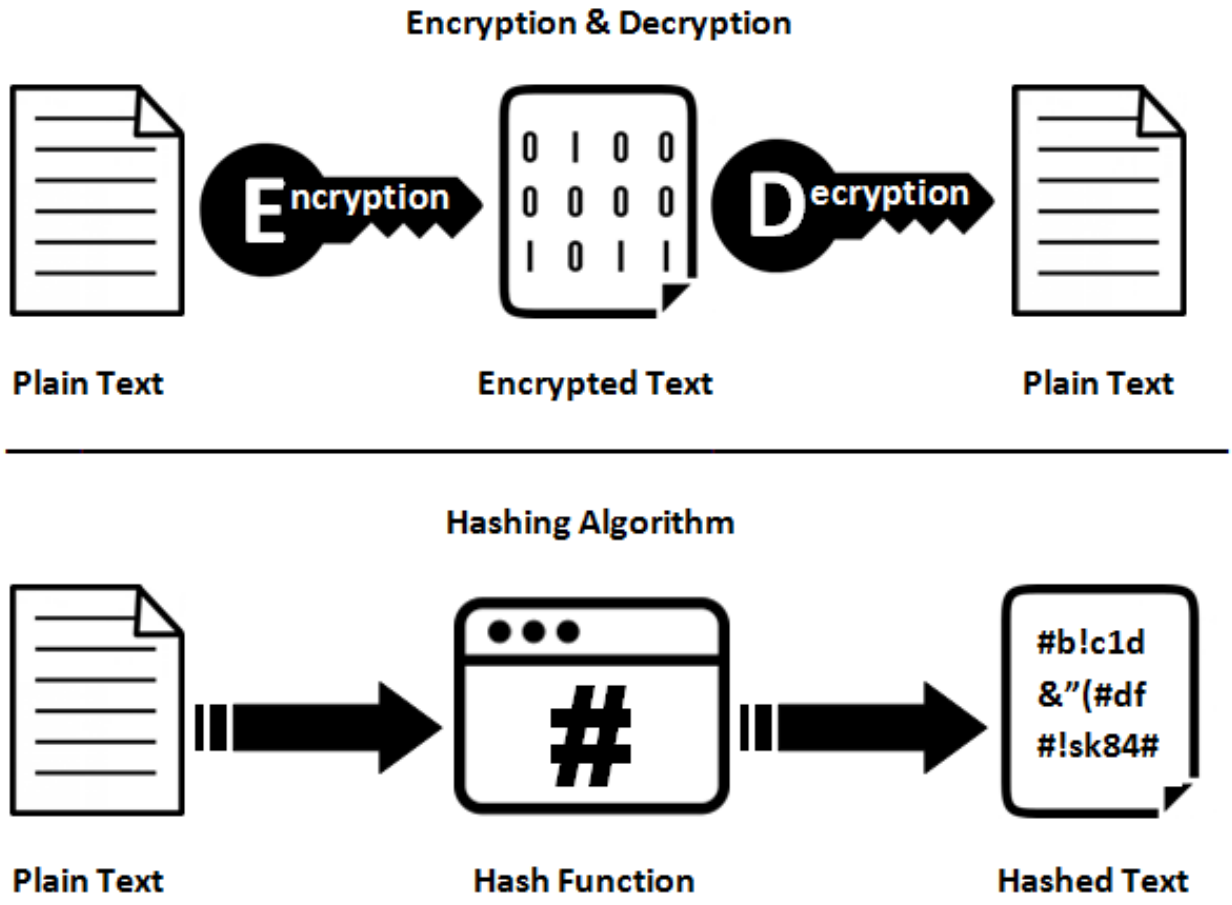

Figure 1.4: The difference between hash and traditional cryptography

1.5.5 Nodes:

In a nutshell, there are two main types of nodes, full nodes and light nodes. Another term to describe nodes is clients which supply wallet functions. Full ones contain a copy of the Blockchain's history, including all blocks created. Light nodes or SPV (Simple Payment Verification) nodes are all wallets that download only the headers of blocks and save hard drive space for users.

1.5.5.1 Full Nodes

Full nodes act as a server in a decentralized network. Their main tasks include maintaining the consensus between other nodes and verification of transactions. They also store a copy of the Blockchain, thus being more secure and enable custom functions such as instant send and private transactions.

When making decisions for the future of a network, full nodes are the ones that vote on proposals. If more than 51% of them don't agree with the proposition, it gets skipped. In some cases, this can lead to a hard fork in which the community cannot agree on a certain change and thus go their separate ways, creating two chains. The most well-known example of that happening is the Bitcoin Cash Fork.

1.5.5.2 Nodes That Can Add Blocks

They depend on the consensus rules being enforced and require at least one full archival node in the network to operate the most popular type is a miner node:

- **Miners:** miners are nodes (either full or light ones) which aim to prove that they completed the required work to create a block. Hence the consensus name Proof of Work. They employ hardware components (be that CPUs, GPUs or ASICs) to solve a cryptographic problem. The first to complete the task broadcasts his results to the network so it can be verified by full nodes and once consensus is achieved – he is granted the right to add a block to the existing Blockchain. For their work, miners are rewarded a pre-defined amount of coins in addition to any transaction fees for the block.

1.5.6 Smart Contracts

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met.

They're stored on the blockchain as machine code, which garanties immutability.

1.6 Blockchain Types

the Blockchain is generally categorized into four types, according to the manner in which a node is allowed permission to participate in the verification of blocks. [8]

1.6.1 Public Blockchain

On a public blockchain, like those used by Bitcoin [15] and Ethereum [25], every node has an equal chance of adding the following block and participating in consensus.

1.6.2 Private Blockchain

Private blockchain, allow only a limited number of permissioned nodes to participate in the consensus process and creation of the next block.

1.6.3 Hybrid Blockchain

Hybrid Blockchain solutions it comes between the private blockchain and public blockchain, basically any node can participate in consensus but the creation of the next block are only limited on a specific group of nodes, for example the cryptocurrency Ripple [22] uses a hybrid paradigm.

1.6.4 Permissioned Blockchain

"Permissioned Blockchain systems have managed to carve out a niche for themselves among all of these topologies" [1] [7]. Any node can join the consensus process in a permissioned blockchain, but all participants identities must be known in advance, therefore any node that wants to participate in consensus and the generation of the next block will lose its anonymity feature.For example Libra currency [21], and RESILIENTDB [7].

1.7 Blockchain Advantages

- **Enhanced security:** As previously said, blockchain uses hash instead of encryption, so technically no one can access your information by breaking security. The recorded information on blockchain is also immutable, and your personal data is anonymized and limited access which gives a better security.
- **Greater transparency:** Since blockchain is a distributed ledger, data and transactions are recorded identically in different locations at the same time. As a result, all nodes having access to a certain data will always view the same information without any changes.

- **Increased efficiency and speed:** As is common knowledge, peer-to-peer blockchain transactions can be done without the need for a third party, that reduces transaction time and costs. Also in contrast to the conventional method, blockchain does not utilize papers, which is prone to human error.
- **Automation:** One of the significant factors leading to blockchain's efficiency and speed is the use of "Smart Contracts," which are triggered automatically when the requirements are met. For instance, if the customer provides all the necessary documents to make a claim, the claim can be quickly settled and paid.

1.8 Blockchain Disadvantage

- **High implementation costs:** Unfortunately, while this technology offers inexpensive costs to customers, it also comes with substantial implementation costs for businesses, delaying its widespread adoption and implementation. [24]
- **Inefficiency:** One of the issues with blockchain is that, during the mining process, every user is validating the same data and and the reward to all the process participants. Also, a significant quantity of energy is wasted during that process, makes this technology unfriendly to the environment. [24]
- **Private keys:** Although the high level of security provided by blockchain is a good thing, But if a wallet's private key is lost, it will be difficult to reconnect to the wallet or restore the data inside. [24]
- **Storage:** As the number of users grows, so will the number of operations that must be integrated into the blocks to be stored, requiring more storage inside the miners computers, eventually exceeding the hard disks capacity. [24] Another problem with blockchains is that as the network expands, more operations will need to be stored as part of the blocks. As a result, the hard disk capacity in miners machines will eventually run out and won't be enough.
- **Unemployment:** Blockchain simplifies transactions because it works peer to peer, that's eliminates the need for a third-party to complete transactions, which reduces the demand for jobs. [24]

1.9 Blockchain can be two-edged sword

At least a few of aspects of the Blockchain network can be seen as both an advantage and a disadvantage , some of this aspects are:

- **Immutability of information:** blockchain technology it is not excepted from errors, and the fact that the information contained in the blocks cannot be changed represents a major problem in the face of these possible errors.

- **Anonymity:** Although anonymity is a virtue for most users, it stems from their faith in the Blockchain network to validate person-to-person transactions. This feature has also been exploited, even to do illegal acts, because it makes transaction tracing impossible.

1.10 blockchain fields of application

Blockchain can be used in many fields, such as:

- **Banking and Finance:** Perhaps no industry stands to benefit from integrating blockchain into its business operations more than banking. because it will add the speed and extends the works hours from 8 hours 5 days a week to 24 hour 7 days a week. [9]

- **Currency:** the most known use of blockchain is currency , the bitcoin itself has known as the first blockchain application. Blockchain lets Bitcoin and other cryptocurrencies to operate without the need for a central authority by distributing their operations over a network of computers. This not only lowers risk, but it also removes a lot of the transaction and processing fees. [9]

- **Healthcare:** Healthcare providers can use blockchain to maintain their patients' medical records in a safe manner. When a medical record is created and signed, it can be stored on the blockchain, giving patients confirmation and assurance that the record cannot be altered. These personal health records might be encrypted and saved on the blockchain using a private key, guaranteeing that only certain people have access to them.

  Due to its sensitive data and nature, the health care industry can benefit from blockchain in a number of areas, including genomics, supply chain, and electronic health records. [9]

- **Voting:** A modern vote can use the blockchain, since every blockchain wallet have a unique address , and all operations and transactions data are available to all participants , that will prevents fraud, forgery and false election. [9]

- **Gaming:** With the use of non-fungible tokens, blockchain technology can transform the digital goods found inside video games—such as collectibles, weapons, and cosmetic skins—into real-world assets that can be traded like stocks or bonds (NFTs). [9]

1.11 Conclusion

This chapter gives an overview about blockchain in general including history , structure , types adventages, disadventages and fields application , the next chapter will take a closer look into blockchain uses for healthcare Sectore.

Chapter 2

Blockchain applications in Healthcare

2.1 Introduction

The past chapter was talking mainly about the blockchain technology about the history , components , functionality and use cases of it. This chapter will specify the healthcare sector use case ,It will talk about the utility of blockchain in healthcare.

2.2 Applications of blockchains in health Sector

In health care sector there is multiple use cases for the blockchain the figure 2.1 shows some of it. this part the chapter will explain some of the applications of the blockchain technology in healthcare sector

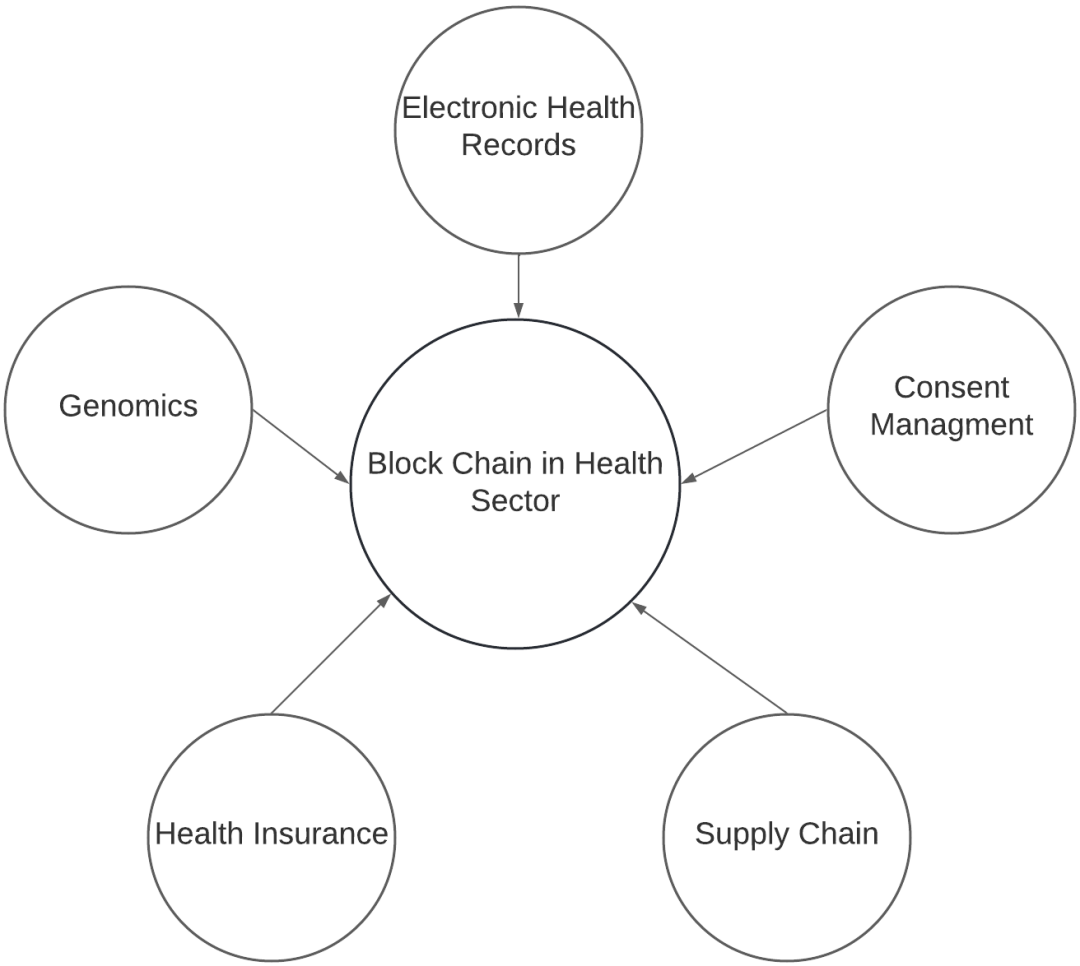

Figure 2.1: Blockchain use cases in health sector

2.2.1 Electronic Health Records (EHR)

The most obvious idea of implementing blockchain health field is the electronic health records, because storing patient data safely in variety locations , but without the full access to the shared patient database. One of the most important blockchain

characteristics that can Electronic Health Records take the advantage of is the immutability, which prevents data manipulation. In addition, we will eliminate a lot of paper work every time a patient visits a healthcare provider ,since all of the patient's medical information will be easily accessible by it regardless of place or time. [19]

2.2.2 Supply Chain

In simple words the supply chain is the activities an accompany provides to let a services or items to the consumer in medical case it would be the drugs, devices and medical services.

Blockchain provide a number of solutions for supply chain use case which face the medical industry today, it allows the transactions tracking , as well as it can increase overall efficiency and the removal of unnecessary intermediaries.

Also Blockchainoffers the ability to keep the immutability of records of available drugs, services and devices , which help to avoid bying expired or counterfeit products. Additionally, due to the accurate tracking of transactions between the provider and the supplier, blockchain offers the potential to reduce costs and the opportunity to approach a hospital or medical provider in the case of a problem. [11]

2.2.3 Health Insurance

The immutability of the blockchain guarantees the prevention of soft insurance fraud, smart contracts can also be used in this case to remove the chance of making false claims. [18]

2.2.4 Genomics

Blockchain has the potential to lower the cost of the data owner's revenue and the analytical costs when it comes to genomics and healthcare analysis. This is because one of the fundamental features of blockchain is peer-to-peer transaction, which also makes it faster and more efficiently than relying on an intermediary. Additionally, unlike central systems, the blockchain is an immutable ledger because it was created with all users cooperation and stored in a distributed system. This allowe transaction histories to be more visible and traceable and prevent the fraud or illegal activity even better than central systems. [5]

2.2.5 Consent Management

Consent is the permission for using an individual's personal information, be it for research or marketing and analysis purposes, in this case the relevant information is the patient's health information. [10] The main steps in consent management involve:

1. Collection of consent;

2. Storage of consent;
3. Use of collected consent and data.

The question now is : how can the blockchain be used for consent management? The following comprises an effective consent management system:

- Data transparency
- Secure storage
- Accurate data management

Due to the fact that all transactions and movements on the blockchain will be logged, users will be able to trace their own information and determine whether it is involved with any particular transaction or movement. Also Blockchain's immutability is both its most definitive and crucial component. The consent of a user cannot be changed once a record has been added to the network.

2.3 Comparison between the classic solution and the blockchain solution

A data structure that maintains track of the network's transactions is at the heart of blockchain technology. The immutability of the stored records is a unique property that sets it apart from previous technology.It employs consensus and cryptographic approaches to achieve immutability.

This technology is also known as "Distributed Ledger Technology (DLT)" since the data is kept in distributed nodes. As more experts and practitioners join the blockchain hype, some are voicing concerns about the fundamental differences between blockchain and traditional databases, as well as the actual value or potential of blockchain. This section compares the blockchain to the traditional approach in order to show that the blockchain can be more beneficial in this area. [3]

- Health-care is fundamentally very complex and sensitive sector. Adaptation of technology is always very slow due to legislative requirements. [3]
- However, interoperability and collaboration are very important in this sector for service delivery and innovation. Blockchain can be used to enable interoperability and collaboration without compromising the security of the health care providers. [3]
- Applying blockchain in health sector without rigorous research and usability test could be catastrophic. For example, doctor's access may get delayed due to scalability issue of blockchain in a critical moment, which may cause bad consequences. [3]

the table 1 will summarize the comparison between database and blockchain in general

Issue	BlockChain	Central Database	Advantage
Trust Building	Can operate without any trusted party	Need a central trusted party	Blockchain
Confidentiality of Data	(by default) All nodes have visibility of the data	It restricts access to authorized person	Database
Robustness/Fault Tolerance	Data is distributed among nodes	data is stored in central database	Blockchain
Performance	Takes time to reach consensus (e.g., 10 mins for Bitcoin)	Immediate execution/update	Database
Redundancy	(by default)Each participating node has the latest copy	Only the central party has copy	Blockchain
Security	(by default) Use cryptographic measures	uses traditional access control	Blockchain

Table 2.1: Comparison of different between database and blockchain

2.4 Related works

2.4.1 ClinicAppChain: A Low-Cost Blockchain Hyperledger Solution for Healthcare

2.4.1.1 Goals

The objective of this study is to design a global healthcare architecture, using PoS and that allows patient control the access to their health information. this should also be in accordance with the 2018 EU data protection regulation. [14]

2.4.1.2 Advantage

- by using an ecosystems MedicalChain and Hyperledger Blockchain as bases they found a strategically solution to combines on-chain and off-chain storage and uses no cryptocurrency, aiming for robustness and scalability.
- in this paper they propose a more balanced approach by keeping everything on-chain, except for large media files. As just media is off-chain, no written information (e.g., diagnostics) is compromised , which reduce the cost of storing large files and raise protection of the critical informations.
- Instead of crypto-currencies, Hyperledger Fabric uses Byzantine Fault Tolerance algorithm to avoid the expensive hardware and electricity costs, by distributing consensus in 3 types of peers: Endorsement (i.e., users credentials),

Validation (i.e., validate transactions), and Ordering (i.e., to sort validated transactions inside blocks).

2.4.1.3 Results

The results of this article are assuming the practicality of the project matches the theory behind it.
However, the article shows that blockchain-based digital platform applications are the next step in the healthcare field by solving the blockchain specific problems like the scalability and the robustness, by using several technical and strategies while reducing the cost of file storage as much as possible to secure the cost and the protection.

2.4.2 Electronic health records in a Blockchain: A systematic review

2.4.2.1 Goals

"This paper reviews some projects and researches of blockchain in the domain of healthcare records and collect then present the information provided in that researches using a systematic literature review methodology in order to support and ease the understanding of this distributed ledger technology." [13]

2.4.2.2 important points

this work presented an systematic review of blockchain and This research has highlighted several important points we mention of them:

- **Challenges and open questions that are related to EHRs and Blockchain**
  1. Interoperability: lack of open standards, trust between all parties, and data integration.
  2. Scalability: limit on the "block size" and storage capacity
  3. Identification: personally identifiable information (PII), global unique identity, and authenticity.
  4. Infrastructure costs: deployment, infrastructure support of Blockchain, and power consumption.
  5. User experience: how patients interact with their health data.
- **Blockchain principles in healthcare**
  1. Immutability: data integrity and authenticity.
  2. Cryptography: privacy and anonymity.
  3. Decentralization: network operates on a user-touser (or peer-to-peer) basis and every full node has a Blockchain full copy.

4. Transparency: an EHR may be open to viewing while ensuring the patient's anonymity (via cryptography).
5. Auditability: systematic examination of the blocks and the network with the objective of determining whether an operation is correct or not (according to the consistency rules).
6. Nonrepudiation: proof of the integrity and origin.

2.4.2.3 Results

"In this study, a systematic literature review regarding EHRs within a Blockchain was conducted, with the objective of identifying and discussing the main issues, challenges, and possible benefits from Blockchain adoption in the healthcare field. The application of Blockchain has exceeded the scope of the field of economics and we have highlighted Blockchain's potential for the healthcare area, while also revealing that it still highly depends on the acceptance of the new technology within the healthcare ecosystem. Analyzing the results that were obtained from the literature review, we conclude that Blockchain technology might be a future suitable solution for common problems in the healthcare field, such as EHR interoperability, establishing sharing trust between healthcare providers, auditability, privacy, and granting of health data access control by patients, which would enable them to choose whom they want to trust and with whom to share their medical records." [13]

2.5 Conclusion

This chapter presents an overview about blockchain in the medical health sector in addition to its uses and the big advancement that it is providing to this field, on top of that it compared the blockchain with traditional solutions,The next chapter will presents the design and conception phase which is the theoretical part of the application.

Chapter 3

Application design

3.1 Introduction

The previous chapter described the use cases of block-chain in the healthcare sector and compared it to more classic solutions, This chapter will look into the architecture of the application and the overall design in aim of building a completed application that will be best suited to solve the problems that has been previously recounted.

3.1.1 Problem

In E-health sector the biggest challenge could be faced is sharing sensitive patient data with the Stakeholders (laboratories, doctors, hospitals, insurance, patients),while at the same time insuring its privacy and safe keeping.

3.1.2 objective

The goal of this work is to propose and implement a new simple web application based on blockchain technology using smart contracts to store electronic health records, In the aim to do so, There are several steps that will be explained in the rest of the chapter like identifying the parties, processes, actors who are participating and System architectures...etc.
This application will provide the ability to manage health records safely while ensuring that private information goes directly to the concerned actors (health workers and information owners), and this will be possible thanks to the application architecture and the Blockchain characteristics.

3.2 Smart contract

A smart contract is used to define the business logic, it will effectively serve as the backend of the dApp. To put it simply, it contains all the functionality that is responsible for either writing data to the Blockchain, or reading from it.
Keeping in mind the sensitivity of the manipulated information, the smart contract must also provide security measures and privacy management, while at the same time keeping the code as compact and efficient as possible, which is critical for a smart contract being that it is deployed and used in a Blockchain with PoW consensus.

3.3 Global Architecture

3.3.1 Global System Architecture

A private Blockchain will be used as the main architecture, only verified nodes will be allowed to participate and hold a copy of the Blockchain.

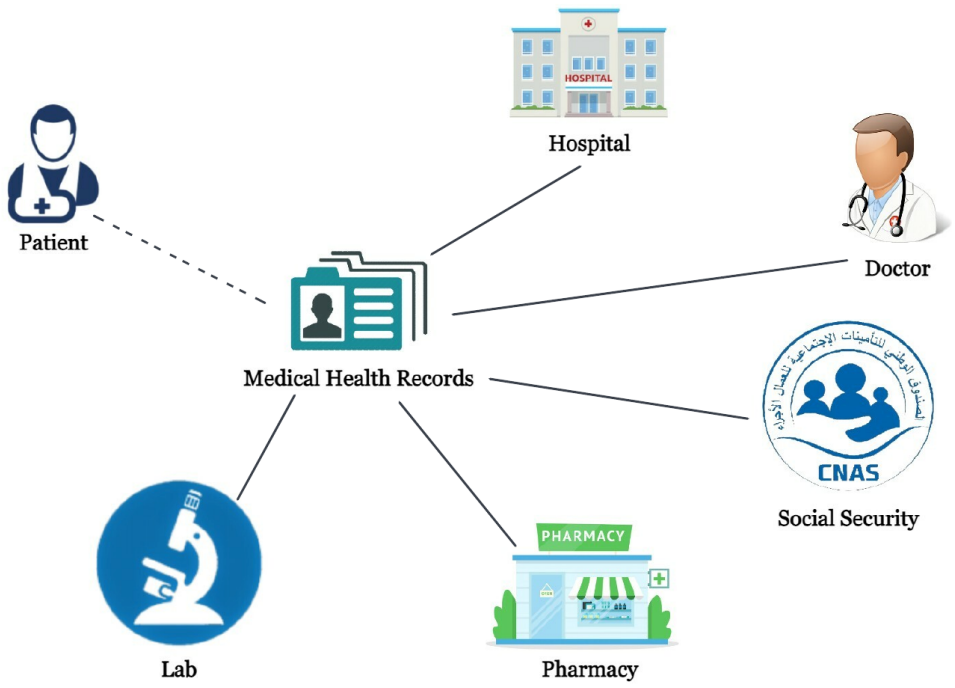

Figure 3.1: Global System Architecture.

3.3.2 Global Network Architecture

The figure 3.2 displays the Global network architecture, which shows the relation that allows the front-end clients to utilize the block-chain back-end through the smart contract, in this process the smart contract acts as a controller.

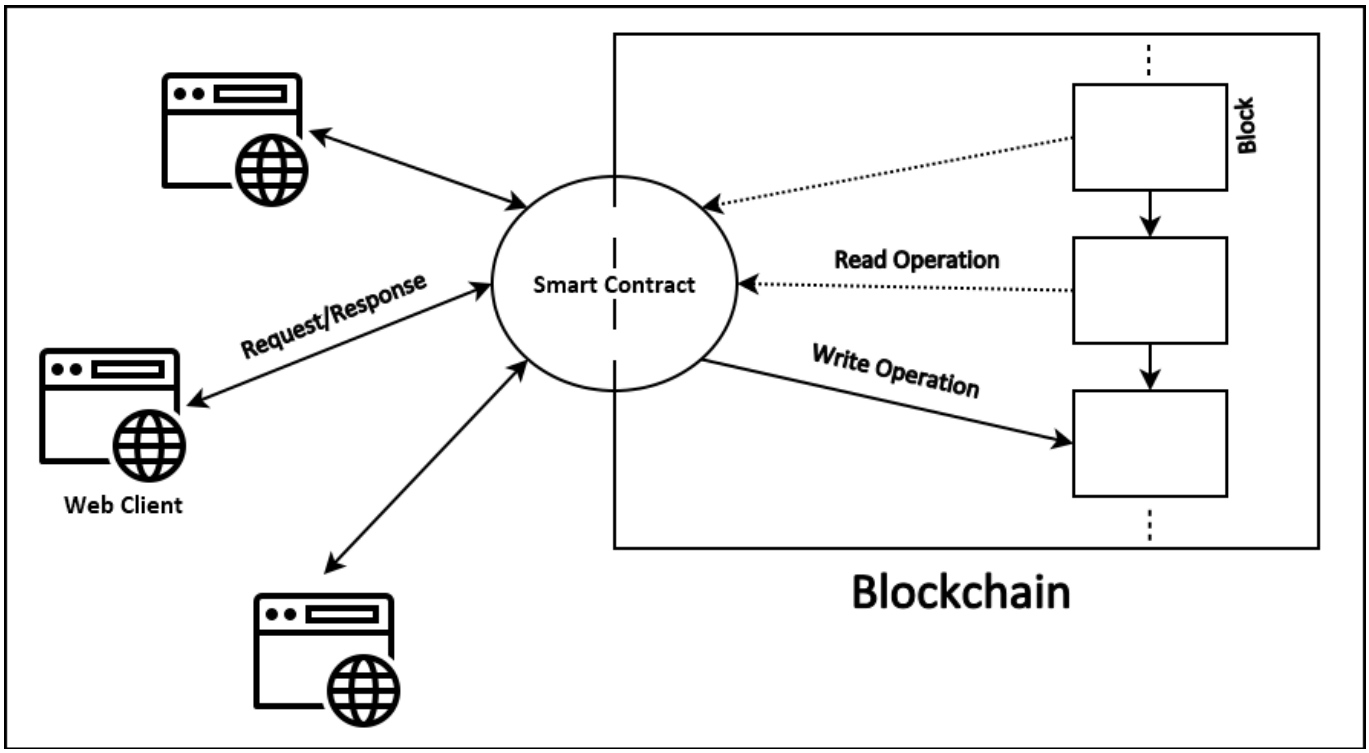

Figure 3.2: Global Network Architecture.

3.3.3 Detailed Network Architecture

The figure 3.3 displays the detailed network architecture, which shows more details about the relation that allows the Front-end to interact with the Back-end through smart contracts, and the use of Truffle in compiling and deploying the smart contracts.

After compiling the smart contracts using Truffle and its Solidity compiler (Solc), they then are also deployed using Truffle, which stores them as machine code on the blockchain itself(by means of a blockchain transaction). The smart contract now contained within the blockchain itself(Back-end), can be interacted with using the JavaScript Web3.js library(Front-end).

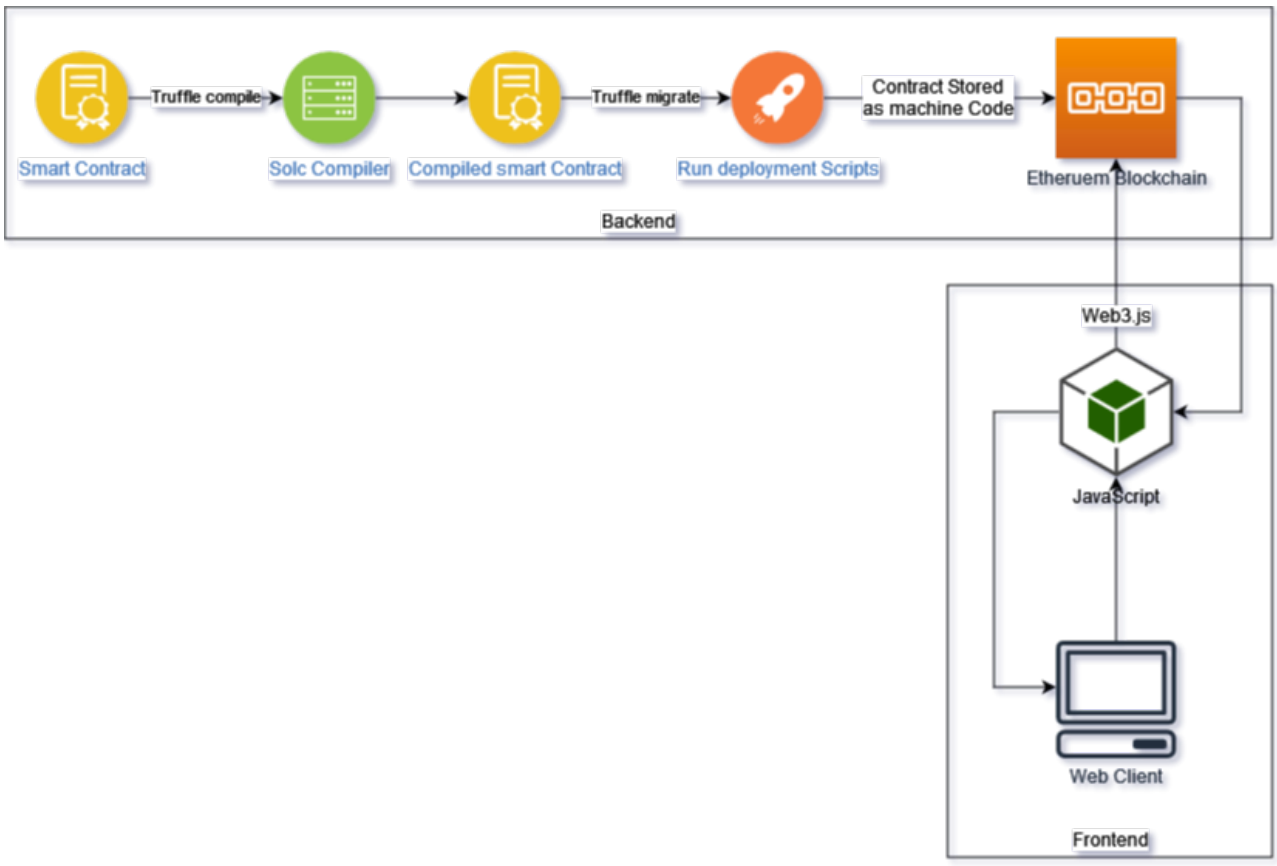

Figure 3.3: Detailed Network Architecture.

3.3.4 Choosing document storage method

In a medical application, there are two main types of information:
light information: These can be either identifying information, or just general medical information regarding the patient, in either case, this type does not consume much space, and therefore can be stored on the blockchain itself.

heavy information: Which is the documents, these can be images, videos, or even text files and many more, unlike the first type, this information is exponentially more storage consuming, storing these bigger files on the blockchain can be very expensive and also inefficient, therefore they must be stored externally.

Seeing as storing documents is the main use of this application, the storage method needs to be secure, fast and also convenient to use. There are couple suggestions to solve this problem which are:

1. **IPFS Server:** The InterPlanetary File System is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. in order to use this service, a vast network of users is required, which is difficult to implement, unless if using a public IPFS server.

2. **SQL server:** The classical solution is also an option, an SQL server, to save the files, this method is easier to implement, and has its disadvantageous, but the biggest draw back is the centralisation of file storage.

3.3.4.1 Storage method Architecture with IPFS server

Out of the two solutions proposed, Using an IPFS server is the most suitable method to store files on this application, because it further grounds the notion of decentralisation and is used much more frequently in these circumstances, On the other hand an SQL server will bring back the main problem, which is the files all being under one authority, this then leads to a loss of the transparency element.

Document Upload:

In this method the User(Health Worker) enters the document information(title, date...), and also selects the file to be uploaded, after confirming and submitting, this file is then uploaded by the Web Client to The IPFS server using The JavaScript library IPFS.JS, the server then returns the hash, which is effectively an Id representing the file, finally this file hash along with the information submitted previously, is sent to the blockchain to be stored. the figure 3.4 describes the storage method.

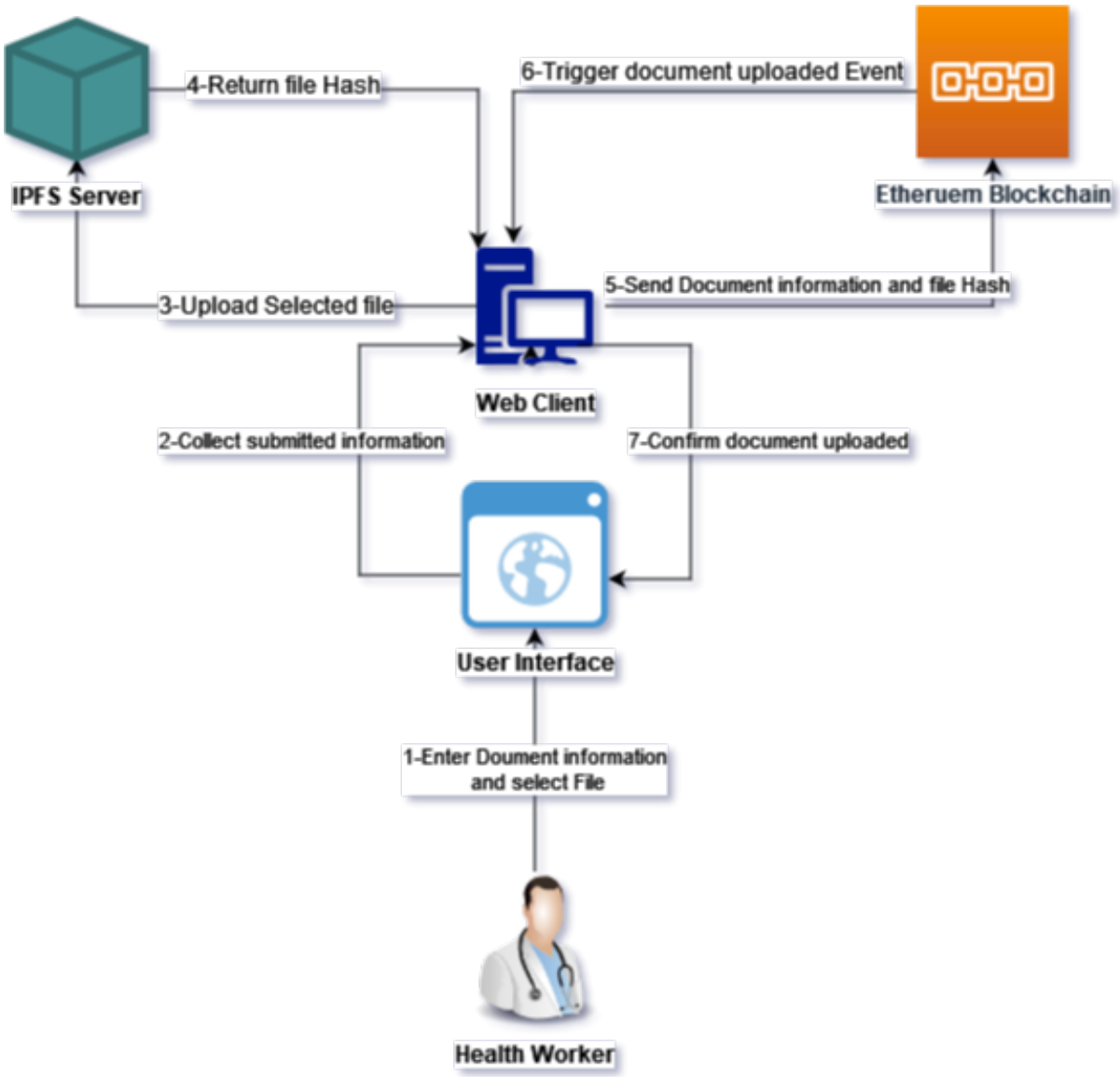

Figure 3.4: Document Upload Method

Document Download:

The User(Health Worker) selects the document to be shown, the Web Client then gets the file hash which is stored in the blockchain, this hash is then used to request the file from The IPFS server using The JavaScript library IPFS.JS, the server then returns the file, finally this file is then displayed to the user along with the document information. the figure 3.5 describes the storage method.

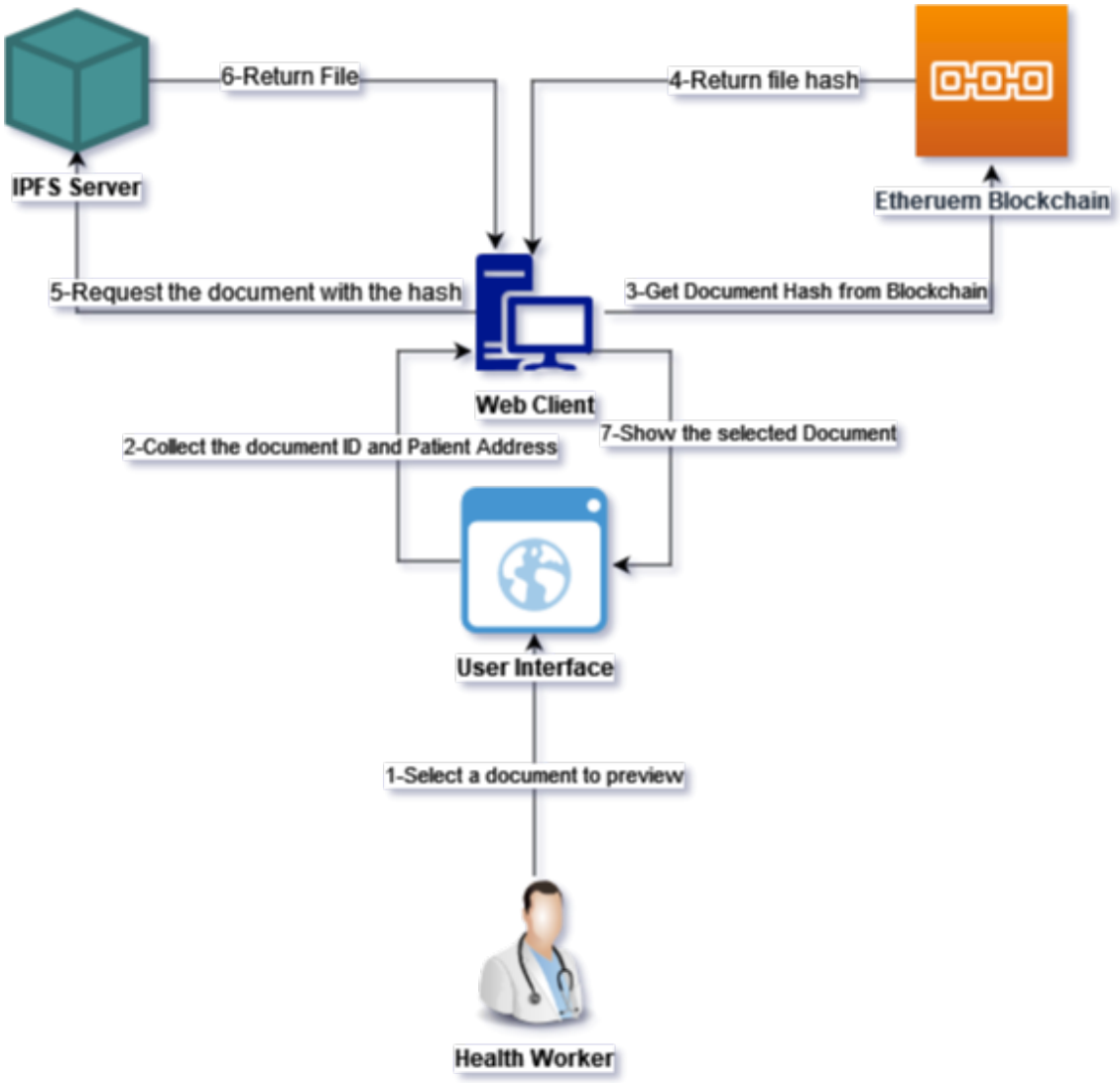

Figure 3.5: Document Storage Method

3.3.5 System actors

the suggested Actors who's going to interact with the application going to be:

1. **Admin Owner:** System organizer and he who gives the access permissions.
2. **Admin:** System supervisor ranked under the Admin owner, can do most of what the admin owner can do, but some operations are only limited to the Admin owner.
3. **Patients:** The system is made up of several patients who are treated by the attending healthcare workers of this network
4. **Healthcare Workers:**
    - **Doctor**: The system consists of several referring doctors, each doctor can add medical information or documents to the patient's record.
    - **Hospital:** Adding medical information, can view or retrieve data.
    - **Laboratory worker:** Documents the patient's laboratory exam results.
    - **Social Security:** Verification of treatments and payment settlements.
    - **Pharmacy:** Documents delivered medications and treatments.

3.3.6 Global use case

The figure 3.6 represents the general use case diagram of the application, it shows every system user and the actions they can enact.

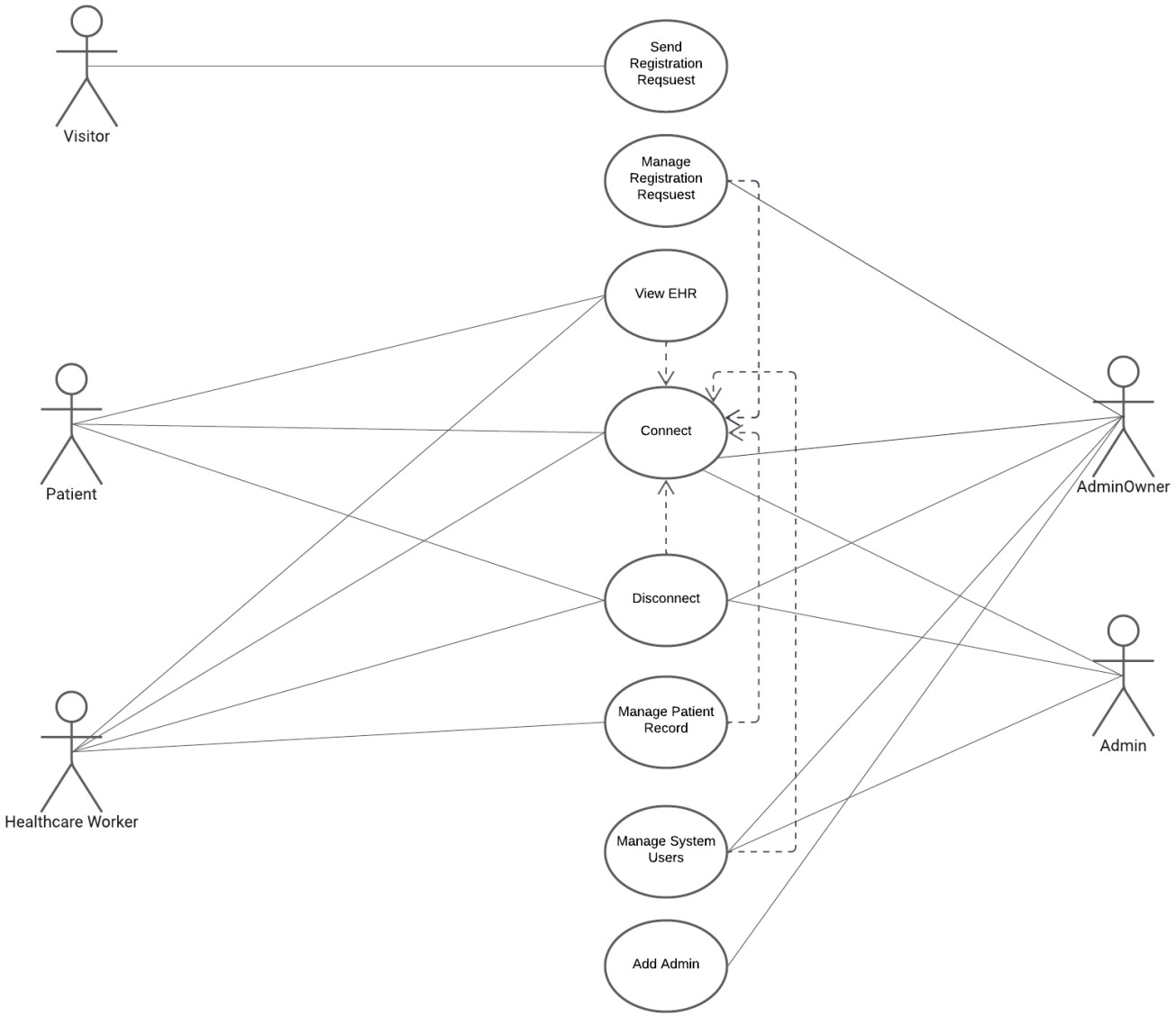

Figure 3.6: Global Use Case Diagram.

3.4 Detailed view

3.4.1 Use Case diagrams

3.4.1.1 Admin Owner (Administrator)

The Admin Owner will be able to manage things such as create accounts of users, other Admins (supervisors) and the health workers, who are able to interact with patients and their health records on top of that he will be able to reject or refuse registration requests of the visitors.
But because of the sensitive nature of health records, the admin himself will not be able to view any medical information concerning the patient. The following 3.7 figure will present the use case diagram of the Admin Owner.

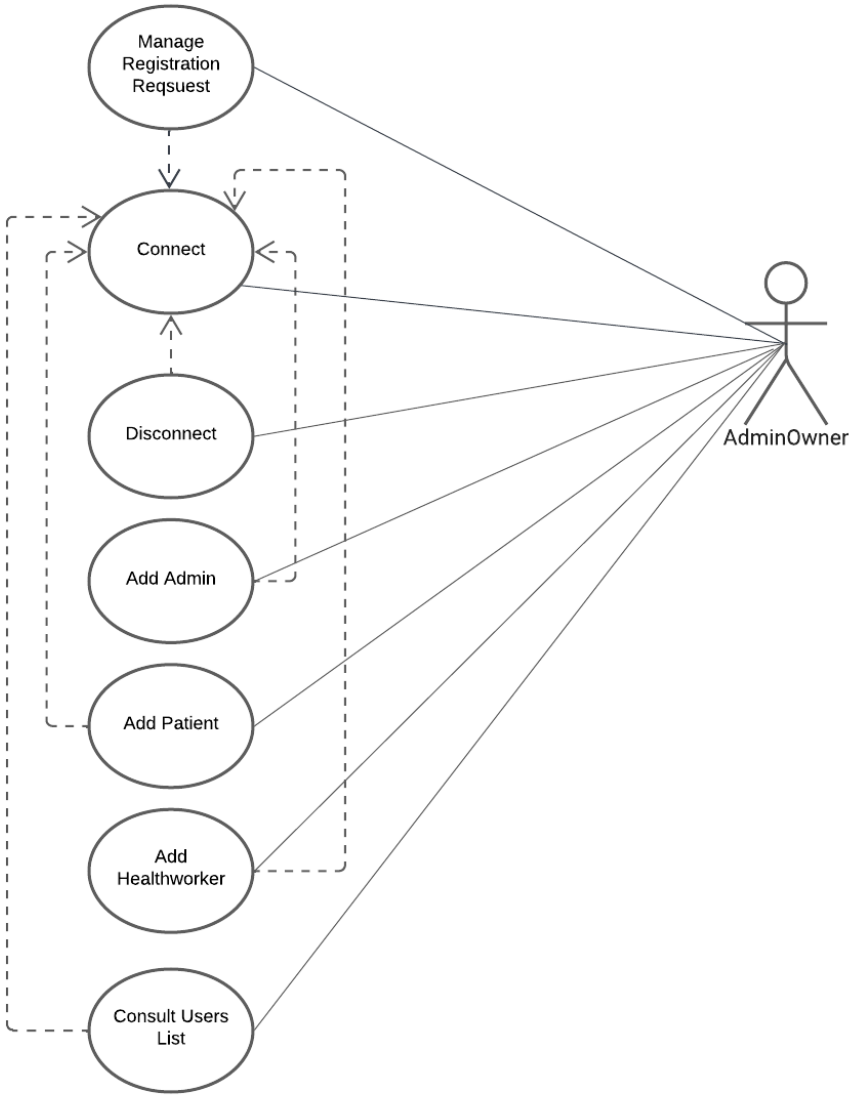

Figure 3.7: Admin Owner Use Case Diagram.

3.4.1.2 Admin (Supervisor)

The Admin will be mostly have same access as the Admin owner but he will be more like a supervisor on the application he won't be able to accept or reject visitors registrations request and he won't be able to create users with Admin role, except that he can do all what the Admin Owner can do The following 3.8 figure will present the use case diagram of the Admin.

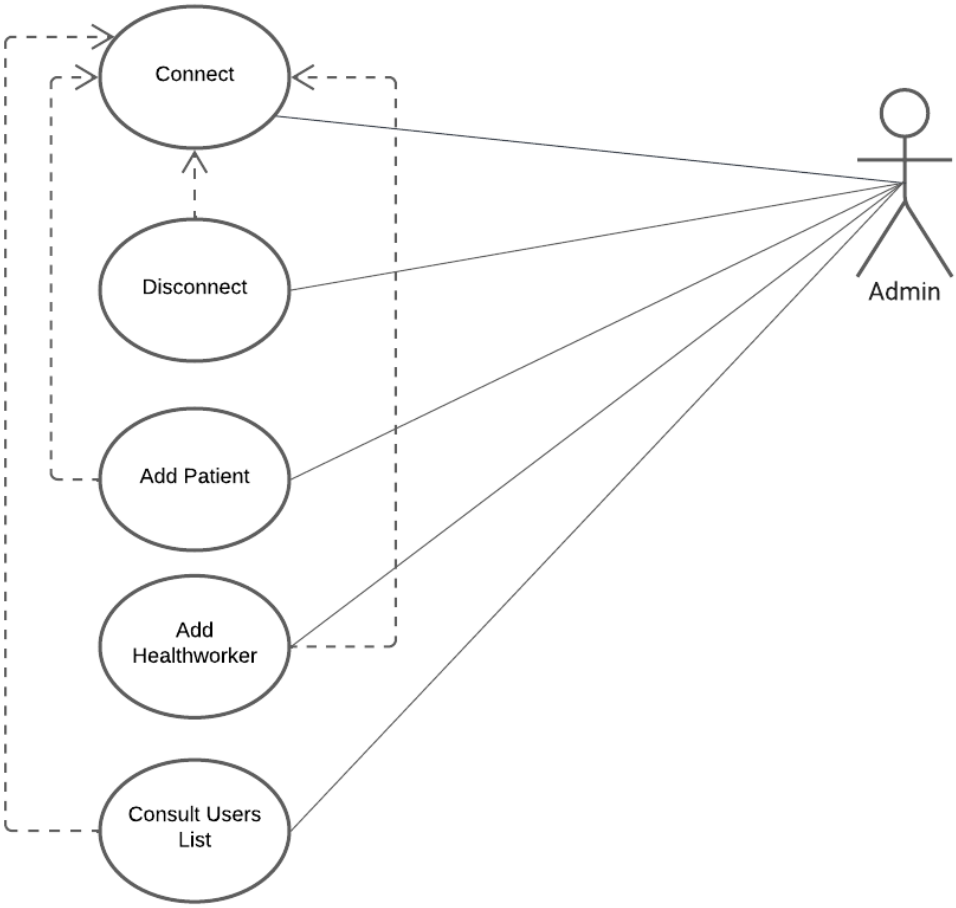

Figure 3.8: Admin Use Case Diagram.

3.4.1.3 Visitor

The visitor is not really an actor in the application, so to become a part of it he have to join by sending registration request. The next 3.9 figure will present the Visitor use case diagram to give more clear picture.

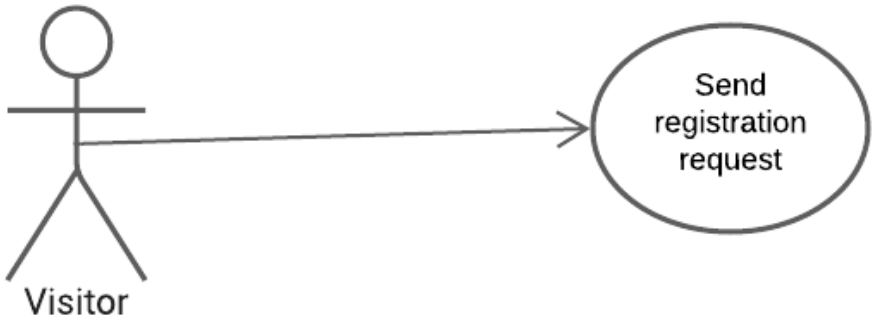

Figure 3.9: Visitor Use Case Diagram.

3.4.1.4 Patient

The patient will be able to complete simple tasks, such as consulting his own records and edit his own contact information. The next 3.10 figure will present the patient use case diagram to give more clear picture.

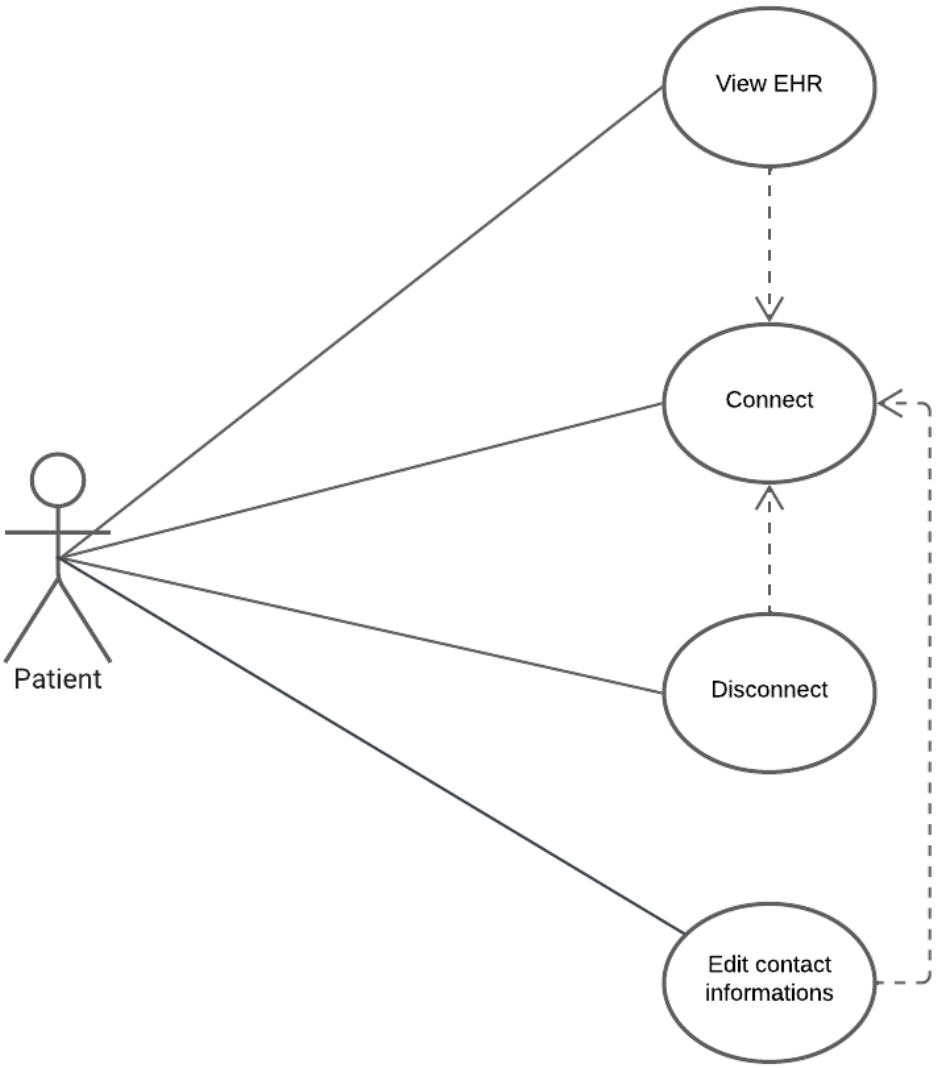

Figure 3.10: Patient Use Case Diagram.

3.4.1.5 Health professionals

The Healthcare worker is the user that can manage the records of the patients. he will able to consult, add and modify on the health records of patients to other health workers. The 3.11 figure next will show the use case diagram of the healthcare worker in the app.

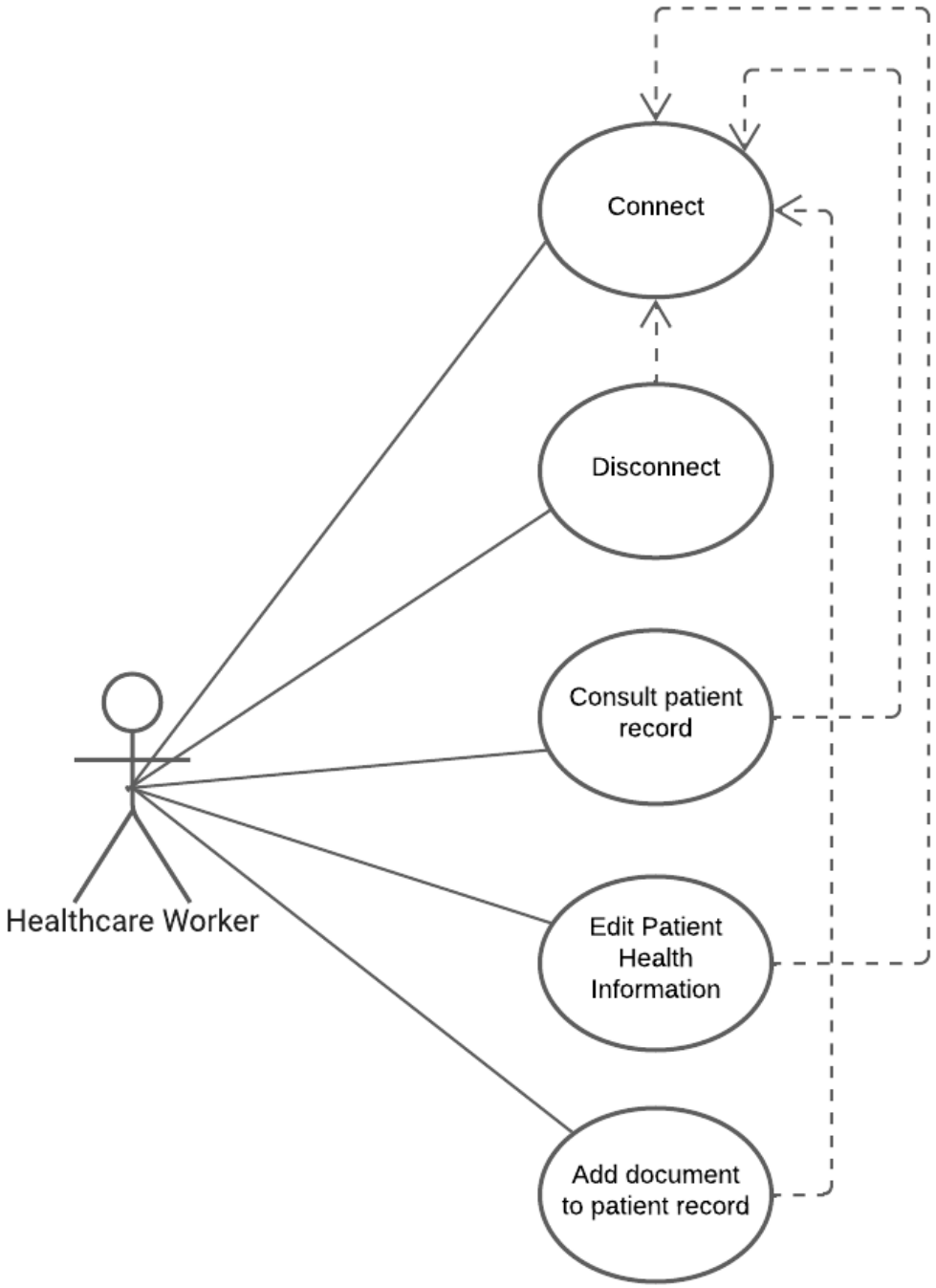

Figure 3.11: Health Worker worker Use Case Diagram.

3.4.2 Sequence Diagram

3.4.2.1 visitor registration request

The figure 3.12 shows the sequence diagram of the way visitor can send a registration request to join to the system and how the system will deal with it.

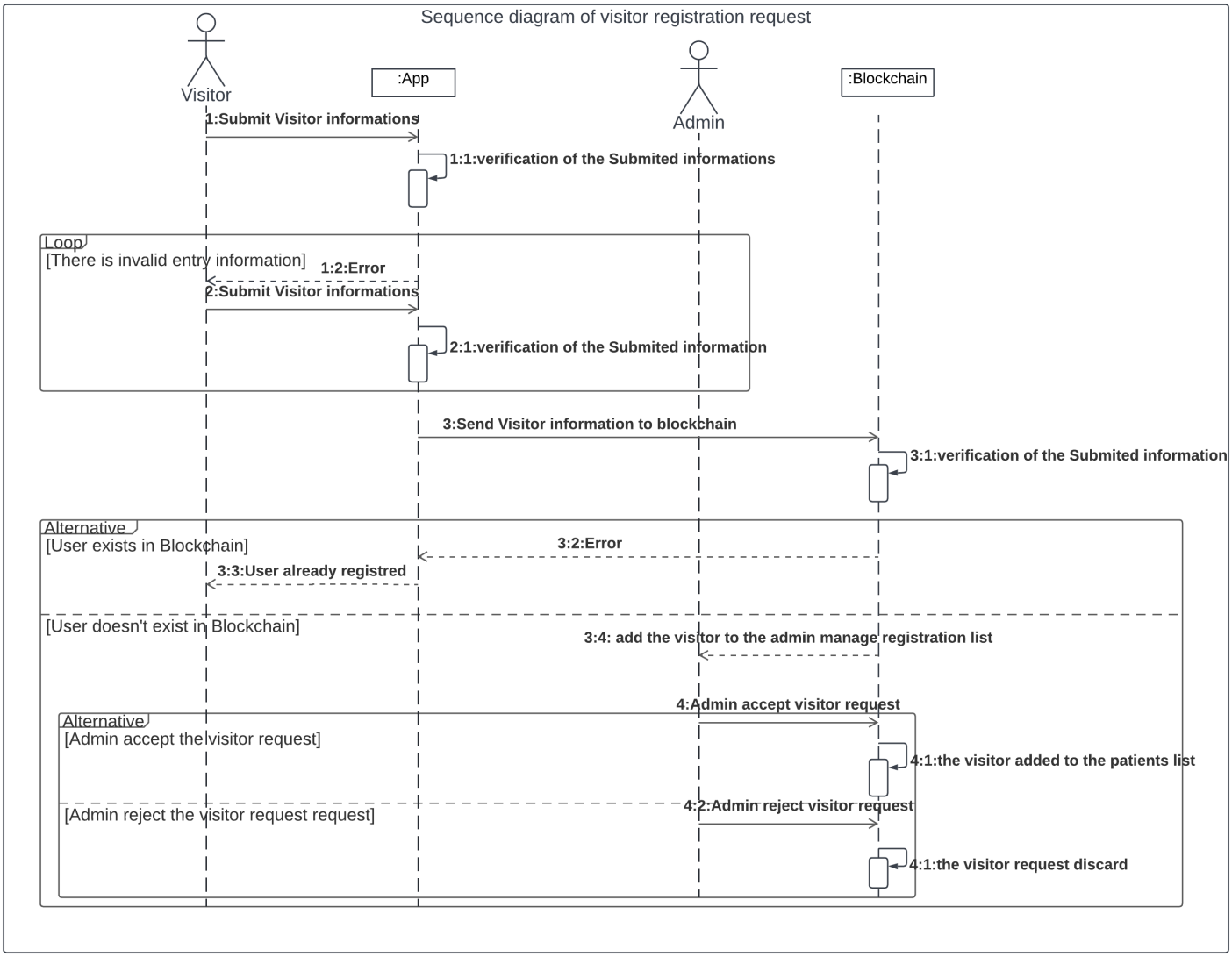

Figure 3.12: Sequence diagram of visitor registration request.

3.4.2.2 Authentication with Meta Mask

The figure 3.13 shows the sequence diagram of the most basic process in the application which is the login, in the application MetaMask is used as an authentication tool.

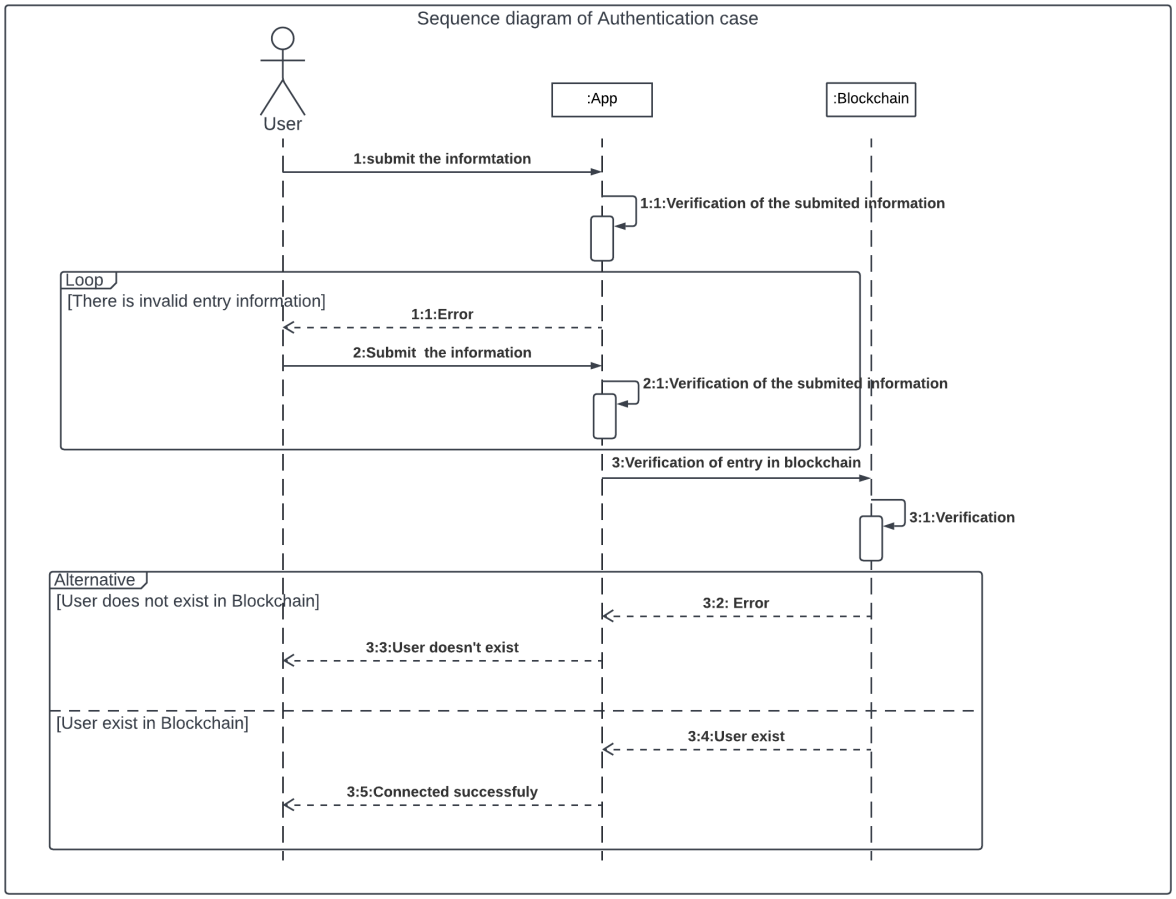

Figure 3.13: Sequence Diagram of Authentication with Meta Mask.

3.4.2.3 Admin Register User

the figure 3.14 represents the process there is used to register a new user of any role by the admin.

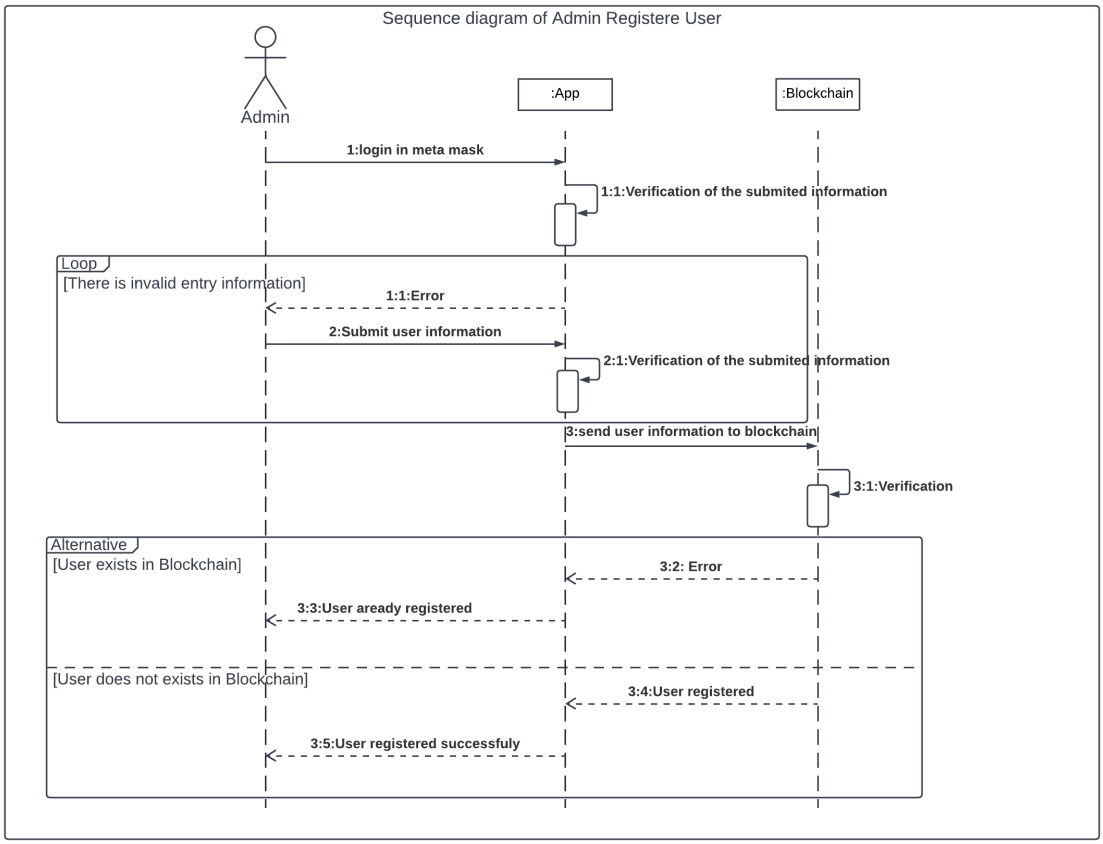

Figure 3.14: Sequence diagram of Admin Register User.

3.4.2.4 Patient Consult Record

the figure 3.15 represents the process the patient takes in order to view his own record.

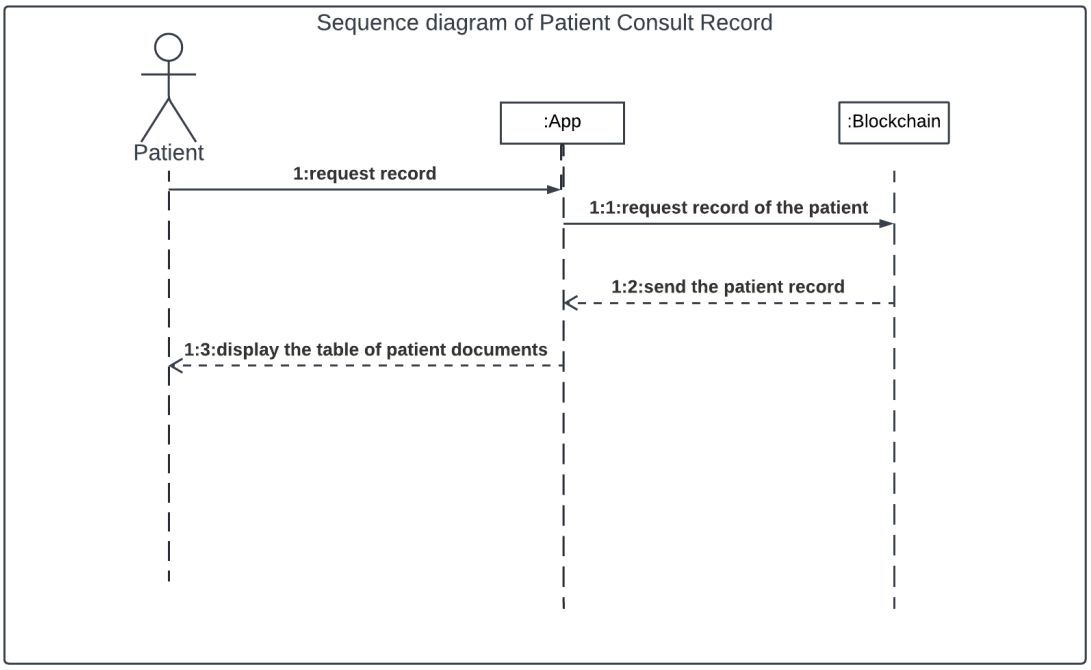

Figure 3.15: Sequence diagram of Patient Consult Record.

3.4.2.5 User edit contact information

the figure 3.16 represents the process the patient follows to edit his contact information.

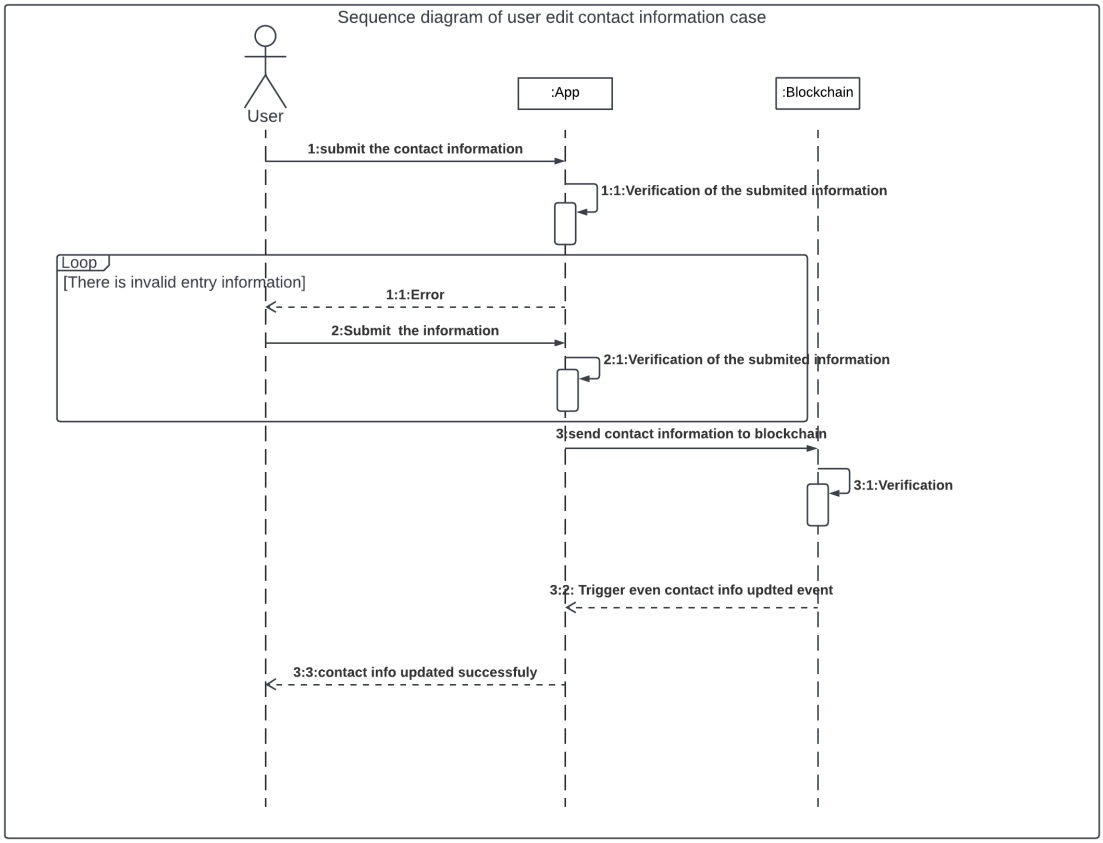

Figure 3.16: Sequence diagram of user edit contact information.

3.4.2.6 Health Worker Consult Record

the figure 3.17 represents the process the Health Worker takes in order to view a certain patient record.

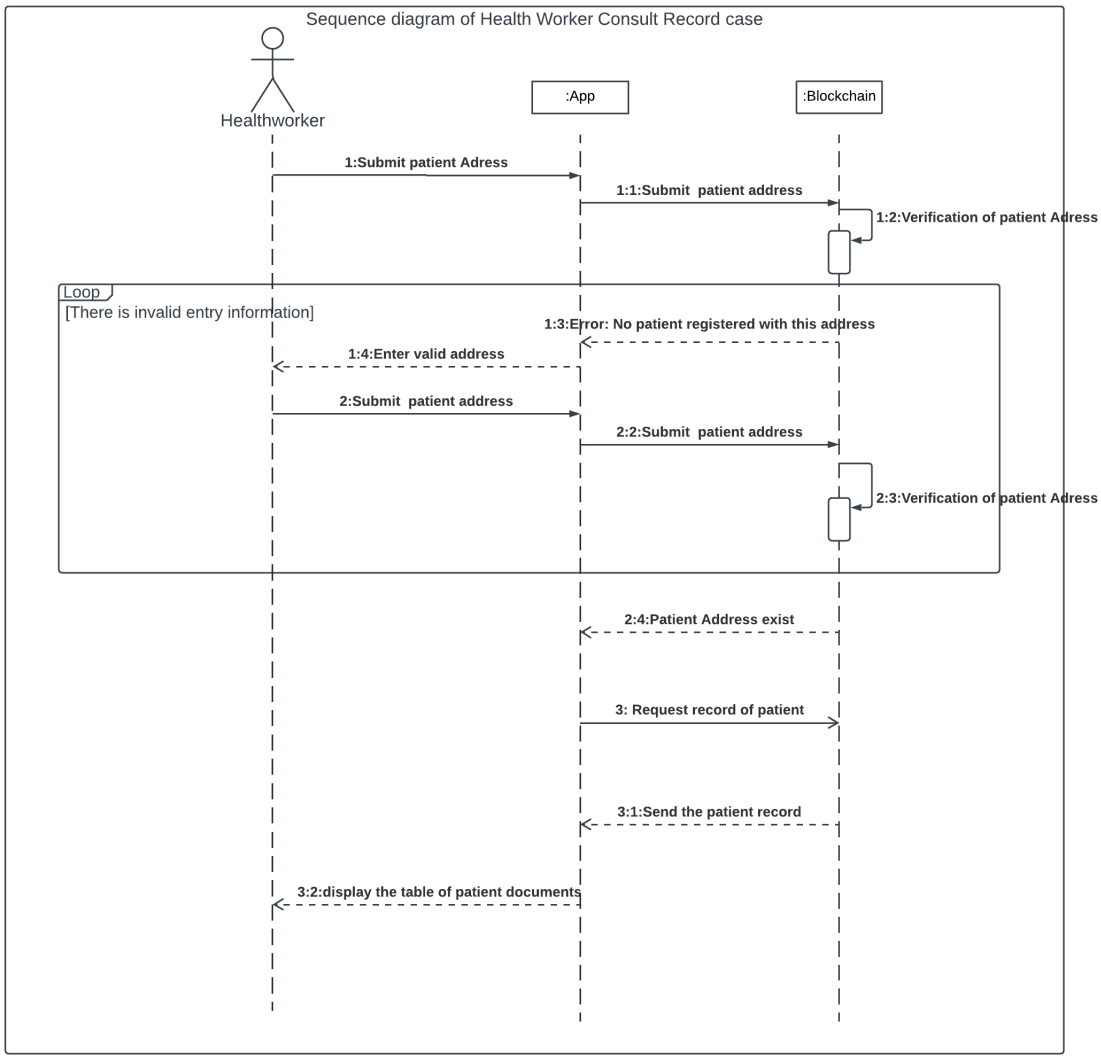

Figure 3.17: Sequence diagram of Health Worker Consult Record.

3.4.2.7 Health Worker add document

the figure 3.18 shows the operations the Health Worker performs in order to add a document to a patient record.

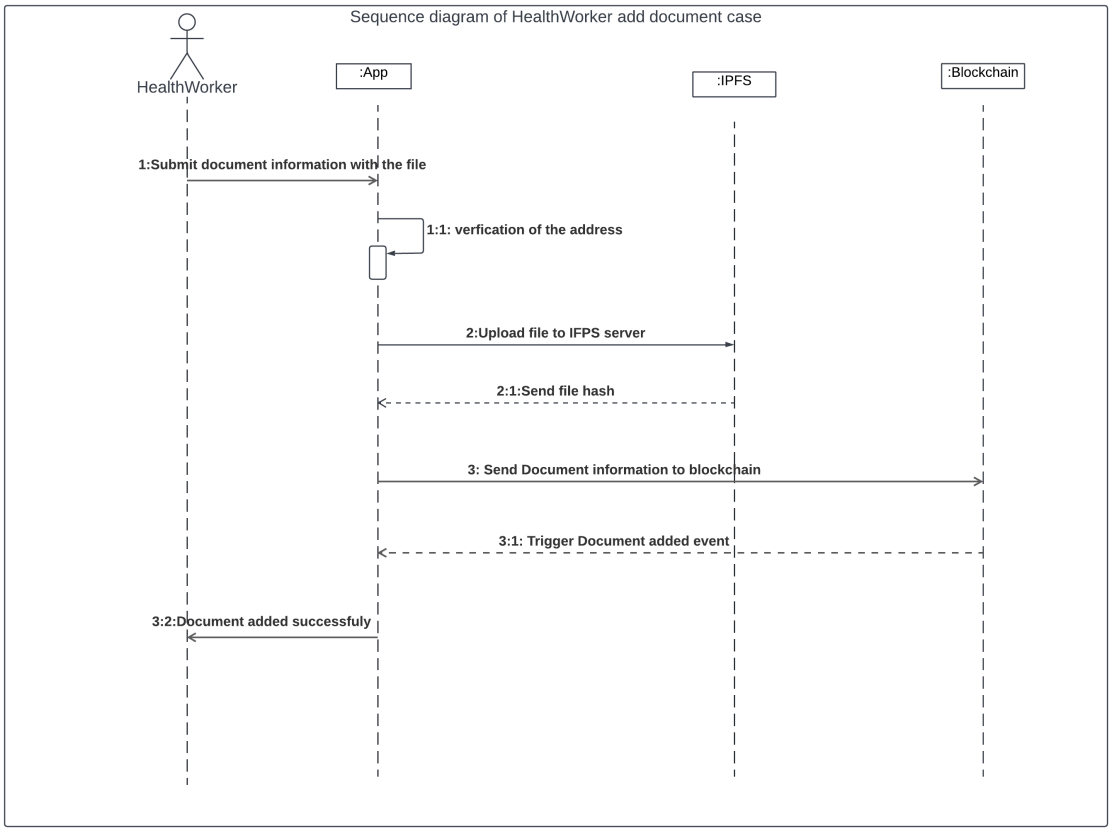

Figure 3.18: Sequence diagram of Health Worker add document.

3.5 Conclusion

This chapter reviewed the global and detailed architecture of the application, it includes UML diagrams for the use case and sequence diagrams, on top of that it presented the illustrations of the application structure and network, which gives the reader a deep understanding on the application.
The next chapter will present the tools used in the realization and the implementation of the system, by providing the technology stack used in the development, and some screenshots of the realized application.

Chapter 4

Realization and implementation of The applications

4.1 Introduction

The previous chapter displayed the theoretical side of the application, the logic and the diagrams behind the project, This chapter is dedicated to the realization of the application and will present the implementation of the system, the technology stack used, the development tools and some screenshots of the realized application.

4.2 The Hardware

First of all this application was realized using two systems, than collected to one machine , the specification of each system is :

4.2.1 The first system specification

- Cpu : Ryzen 5 4650G 3.7 Ghz
- Ram : 16GB DDR4
- Operating System : Windows 11

4.2.2 The Second system specification

- Cpu : Ryzen 5 4650G 3.7 Ghz
- Ram : 16GB DDR4
- Operating System : Windows 10

4.3 Software and programming languages used for Backend

The work was separated into back-end and front-end, to make it easier and more organised, multiple programming languages were used to realize the application.

During the early development stages, choosing between the possible approaches to accomplish the application proved to be challenging, from the different front-end frameworks, to the multiple storage options, finding the right blend of methods can be overwhelming.

4.3.1 Visual Studio Code

A source code editor that can be used with a variety of programming languages including Java, JavaScript, Node js, and C++ with the support of a very big number of helpful extensions. It is cross-platform, open source and free, It is used to develop the front-end as well as the back-end.

Figure 4.1: Visual Studio Code Logo

4.3.2 Truffle

Truffle Suite is a development environment based on Ethereum Blockchain, used to develop DApps (Distributed Applications). Truffle is a one-stop solution for building DApps: Compiling Contracts, Deploying Contracts, Injecting it into a web app as well as Testing, Truffle has been to compile and deploy the smart contracts to the blockchain.

Figure 4.2: Truffle Logo

4.3.3 Ganache

Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. Ganache can be used across the entire development cycle; enabling development, deploying, and testing a dApps in a safe and deterministic environment. Ganache is used to provide a private Ethereum blockchain, which is than used to deploy and interact with the Smart Contracts.

Figure 4.3: Ganache Logo

4.3.4 Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js is used to provide the dependencies, as well as running the Application front-end.

Figure 4.4: Node.js Logo

The first dependency will be needed is going to be the Node Package Manager (NPM) and Interplanetary File System (IPFS), which comes with Node.js.

- **npm** is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well. NPM is used to Adapt packages of code for your apps, or incorporate packages as they are, Download standalone tools you can use right away, Manage multiple versions of code and code dependencies ... and much more.
- **IPFS** is a peer-to-peer (p2p) storage network. Content is accessible through peers located anywhere in the world, that might relay information, store it, or do both. IPFS knows how to find what you ask for using its content address rather than its location.

4.4 Softwares and programmation languages used for front-end

4.4.1 Choosing the way to build the interface

The first thing that a general visitor notices about an Application is the design, therefore, the front end of an application is essential to the user experience, when it

comes to the front end the following factors are very important:

- Loading speed.
- Browser compatibility.
- Functionality Features.
- Media Delivery.
- Ease of use.

taking into account these criteria there are three solutions:

1. **React JS library :** React is considered one of the biggest frameworks at the moment, it is based on JavaScript, React makes it possible to develop a single page application, it is also Component-Based, on top of all that it has a big community which will help during the work.The figure 4.5 shows React.js logo.

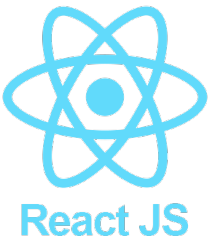

Figure 4.5: React.js Logo

2. **Vue JS framework :** this framework shares a lot of similarities with React JS, both of them are based on JavaScript, and uses the concept of Components, the main difference is that React requires solid JavaScript skills, while Vue.js is more oriented to novice developers. Similar to React, Vue.js enables writing with JSX, but the components are written in HTML templates, and for that Vue JS could be a considered a good medium solution between native and React.The figure 4.6 shows Vue.js logo.

Figure 4.6: Vue.js Logo

3. **native languages HTML5** Using HTML, CSS and JavaScript, this solution will be convenient and straightforward, considering the familiarity of these languages, The coding language also does not require any additional software or plug-ins to run and that was the approved solution in the work. The figure 4.7 shows the three native languages logo.

Figure 4.7: HTML, CSS, JS Logos

4.4.2 Sass

SASS stands for "short for syntactically awesome style sheets" is a preprocessor scripting language that is interpreted or compiled into Cascading Style Sheets (CSS).Basically SASS is an extension of CSS that enables you to use things like variables, nested rules, inline imports and more. It also helps to keep things organised and allows you to create style sheets faster on top of that Sass is compatible with all versions of CSS. The figure 4.7 shows SASS logo.

Figure 4.8: SASS Logo

4.5 MetaMask

MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications. The figure 4.9 shows MetaMask logo.

Figure 4.9: SASS MetaMask

4.6 Application description

The goal in this system is to design a blockchain application for controlling, storing and sharing patient electronic medical records easily between healthcare professionals in a secure and efficient manner. It also makes it possible to provide medical information (medical history, results of laboratory analyses, imaging, current treatment, etc.).

4.6.1 Preparing the environment

This section presents the description of the general steps needed in order to build a dApp.

4.6.1.1 Installing requirements

node.js: Can be downloaded and installed from the official Web Site.

Truffle: Can be installed with npm using the command:

```
npm install -g truffle
```

Figure 4.10: Installing Truffle

Ganache: Can be downloaded and installed from the official Web Site.

4.6.1.2 Writing the Smart Contracts

Smart Contracts represent the brain of every dApp, they implement the expected behavior, given the expected input. A Smart Contract must be concise, efficient but, also guarantee the expected outputs.

The figure 4.11 shows an example of a simple Smart Contract, the contract name is *TestContract*, it contains a global variable myString which is a string, and two functions, one sets the value of myString to its input value, while the second returns the value of myString

```
pragma solidity 0.4.16;

contract TestContract {

  string private myString = "foo";

  function getString() constant returns (string) {
      return myString;
  }

  function setString (string _string) {
      myString = _string;
  }
}
```

Figure 4.11: Simple Contract Example

4.6.1.3 Running the Blockchain

You can quickly fire up a personal Ethereum blockchain using Ganache, by choosing the quick-start option on the start page.

Figure 4.12: Running an Ethereum blockchain

4.6.1.4 Deploying the Smart Contracts

After Writing a smart contract, the next step is to deploy it to the blockchain, but before that,compiling it is necessary, Fortunately, truffle condenses these two

procedures into one command.:
Note:deploying a smart contract is represented in the blockchain as a transaction.

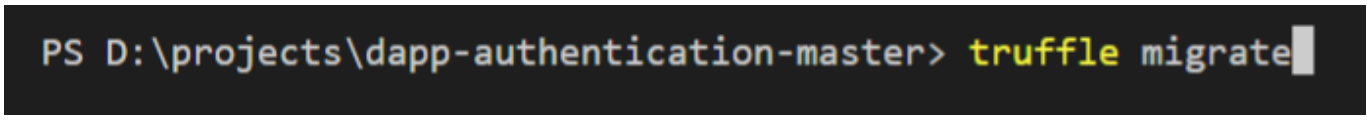

Figure 4.13: Deploying Smart Contracts

4.6.1.5 Testing the Smart Contracts

Testing the smart contracts is a crucial step it guarantees that the smart contract behaves as intended, and saves the developer from issues down the line, that way he can deploy his smart contract confidently.

Truffle comes standard with an automated testing framework to make testing your contracts a breeze, it uses the Mocha testing framework and Chai for assertions to provide you with a solid framework from which to write your JavaScript tests.

The figure 4.14 is an example of a simple smart contract test file,
TestContract.test.js tests the contract previously seen *TestContract*.

test > TestContract.test.js > ...

```
1  const TestContract = artifacts.require('./TestContract.sol')
2
3  contract('TestContract', (accounts) => {
4      before(async () => {
5          this.TestContract = await TestContract.deployed()
6      })
7
8      it('deploys successfully', async () => {
9          const address = await this.TestContract.address
10         assert.notEqual(address, 0x0)
11         assert.notEqual(address, '')
12         assert.notEqual(address, null)
13         assert.notEqual(address, undefined)
14     })
15
16     it('gets string successfully', async () => {
17         const myString = await this.TestContract.getString()
18         assert.equal(myString, "foo")
19     })
20
21     it('sets string successfully', async () => {
22         result = await this.TestContract.setString("someString")
23         const myString = await this.TestContract.getString()
24         assert.equal(myString, "someString")
25     })
26 })
```

Figure 4.14: testing Smart Contracts

This file shows:
• Firstly the import of the smart contract by using `artifacts.require()`.
• Then adding tests inside the contract function.
• The `before()` function is executed before running the tests.
• each test is defined with the `it()` where it contains two inputs, the first is a string which represents the test name, and the second is the actual function containing the test.

These tests can be run using the command:

```
PS D:\projects\test> truffle test
```

Figure 4.15: testing Smart Contracts

This runs all the test files contained in the folder test.

Bellow is the output of running `truffle test` for the file *TestContract.test.js*

```
Aures@DESKTOP-0KLP5E0 MINGW64 /d/projects/test
$ truffle test

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\TestContract.sol
> Artifacts written to C:\Users\Aures\AppData\Local\Temp\test--620-7DCuTwuQlv2z
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


  Contract: TestContract
    √ deploys successfully
    √ gets string successfully (99ms)
    √ sets string successfully (1033ms)


  3 passing (1s)
```

Figure 4.16: testing Smart Contracts

4.6.1.6 Interacting with The dApp

Now that the smart contract is deployed, Interacting with it from the front-end is possible, JavaScript is used for this purpose, the Web3.js library allows the web client to call the smart contract functions. the figure shows an example of calling the function *getString()* from the contract that previously seen *TestContract*.

```
myString = await TestContract.methods.getString.call()
```

Figure 4.17: JS function call example

4.7 The System Smart contracts

The application is built using two smart contracts, *Auth.sol* and *RegistrationReq.sol*, the latter is responsible only for managing the registration requests, while the first is the engine of the dApp. These two were divided for two reasons:

Security: because anyone can send a registration request, the system can then be potentially exploited, which led to the logic behind keeping the registration requests in a different contract.

Gas concerns: in an Ethereum blockchain, that uses the consensus process, transactions that consume more space in a block pay more gas, however the gas price of a single transaction has a limit, if exceeded the transaction will fail, this includes the transaction of deploying the smart contract, this means that, if a contract is too large, its deployment will fail.

the following figures 4.18, 4.19 show a part of each smart contract:

contracts > Auth.sol

```
1  pragma solidity >=0.4.0;
2
3  contract Auth {
4      address ownerAddress;
5      uint256 docCount = 0;
6      uint256 public patientCount = 0;
7      uint256 public doctorCount = 0;
8      uint256 public AdminCount = 0;
9
10     constructor() public {
11         ownerAddress = msg.sender;
12         users[ownerAddress].addr = ownerAddress;
13         users[ownerAddress].role = UserRole.AdminOwner;
14         users[ownerAddress].firstName = "Admin";
15         users[ownerAddress].lastName = "Owner";
16     }
17
18     modifier onlyOwner() {
19         require(msg.sender == ownerAddress);
20         _;
21     }
22
23     modifier onlyAdmin() {
24         require(
25             users[msg.sender].role == UserRole.Admin ||
26                 users[msg.sender].role == UserRole.AdminOwner
27         );
28         _;
29     }
30
31     modifier notAdmin() {
32         require(users[msg.sender].role != UserRole.Admin);
```

Figure 4.18: Auth.sol preview

contracts > RegistrationReq.sol

```
1   pragma solidity >=0.4.0 ;
2
3   contract RegistrationReq {
4
5       uint public registrationRequestsCount = 0;
6
7       mapping(address => RegistrationRequest) public registrationRequests;
8       mapping(uint => address) public requestAddresses;
9       mapping(address => uint) public requestIDs;
10      mapping(address => bool) public deletedRequest;
11
12
13      struct RegistrationRequest{
14          address addr;
15          string firstName;
16          string lastName;
17          string email;
18          string phoneNum;
19          string DOB;
20          string homeAddress;
21          string gender;
22      }
23
24      event RequestSent(
25          address addr,
26          string email
27      );
28
29      function AddRegistrationRequest(
30          string memory _firstName,
31          string memory _lastName,
32          string memory _email,
```

Figure 4.19: RegistrationReq.sol preview

4.7.0.1 Testing the System Contracts

Two test files were used, one for each contract,*Auth.test.js* *RegistrationReq.test.js*, they are represented by the figures 4.20 and 4.21 respectively. these figures show a snippet of each test file.

test > Auth.test.js > ...

```
1   const { assert } = require("chai")
2
3   const Auth = artifacts.require('./Auth.sol')
4
5   contract('Auth', (accounts) => {
6       before(async () => {
7           this.Auth = await Auth.deployed()
8       })
9
10      it('deploys successfully', async () => {
11          const address = await this.Auth.address
12          assert.notEqual(address, 0x0)
13          assert.notEqual(address, '')
14          assert.notEqual(address, null)
15          assert.notEqual(address, undefined)
16      })
17
18      it('initializes successfully', async () => {
19          const patientCount = await this.Auth.patientCount()
20          const doctorCount = await this.Auth.doctorCount()
21          const AdminCount = await this.Auth.AdminCount()
22          assert.equal(patientCount, 0)
23          assert.equal(doctorCount, 0)
24          assert.equal(AdminCount, 0)
```

Figure 4.20: Auth.test.js

test > RegistrationReq.test.js > ...

```
1   const { assert } = require("chai")
2
3   const RegistrationReq = artifacts.require('./RegistrationReq.sol')
4
5   contract('RegistrationReq', (accounts) => {
6       before(async () => {
7           this.RegistrationReq = await RegistrationReq.deployed()
8       })
9
10      it('deploys successfully', async () => {
11          const address = await this.RegistrationReq.address
12          assert.notEqual(address, 0x0)
13          assert.notEqual(address, '')
14          assert.notEqual(address, null)
15          assert.notEqual(address, undefined)
16      })
17
18      it('initializes successfully', async () => {
19          const registrationRequestsCount = await this.RegistrationReq.registrationRequestsCount()
20          assert.equal(registrationRequestsCount, 0)
```

Figure 4.21: RegistrationReq.test.js

Bellow 4.22 is the output of running `truffle test` for these files.

```
$ truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Compiling .\contracts\Auth.sol
> Compiling .\contracts\Auth.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\RegistrationReq.sol
> Artifacts written to C:\Users\Aures\AppData\Local\Temp\test--6652-BH7ZKIHspKMn
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


  Contract: Auth
    √ deploys successfully
    √ initializes successfully (244ms)
    √ registers patient (1391ms)
    √ registers admin (1346ms)
    √ registers health worker (1630ms)
    √ add document (4417ms)

  Contract: RegistrationReq
    √ deploys successfully
    √ initializes successfully (71ms)
    √ sends Registration Request (779ms)
    √ deletes registration request (1241ms)


  10 passing (11s)
```

Figure 4.22: Truffle test output

4.8 The System Interfaces

This part, presents some interfaces, starting with the homepage.

4.8.1 Homepage

Represents the main page of the site, it is the first page a visitor sees, it offers the user to login using MetaMask, whom is then redirected to his part of the application appropriately (admin, patient, health-worker, anonymous/unregistered), the figure 4.23 shows the Homepage interface.

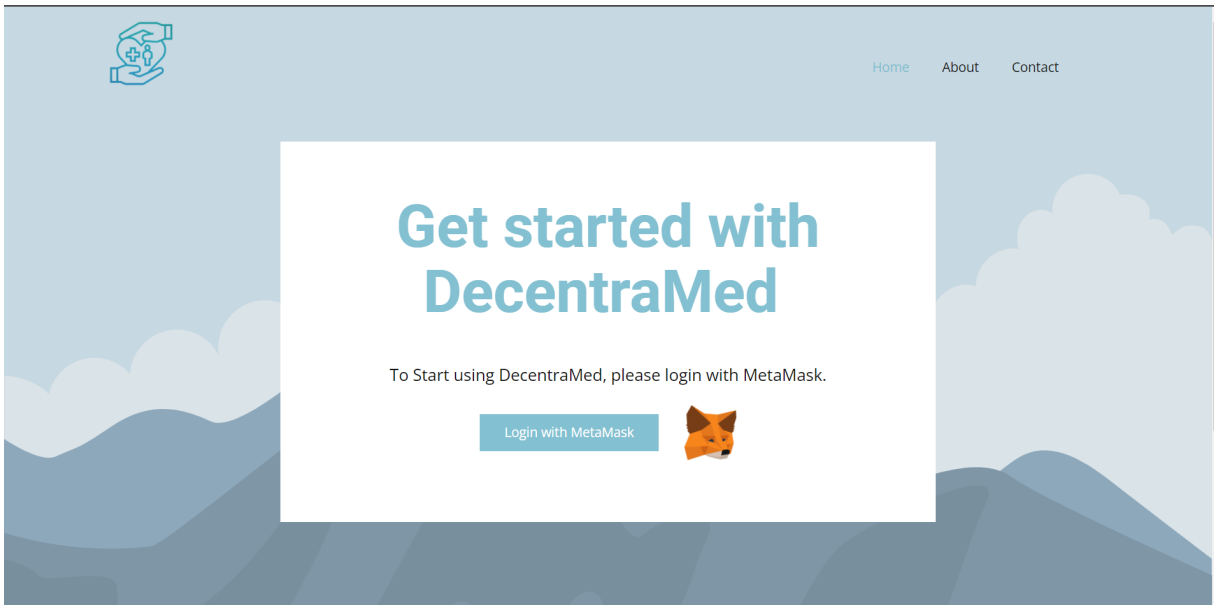

Figure 4.23: Homepage interface.

4.8.2 Visitor Send Request Portal

When a user uses an unregistered address to login with MetaMask, he's then redirected to this page, where he can send a registration request, which then can be accepted or rejected by the AdminOwner, the figure 4.24 introduce the Visitor Send Request Portal interface.

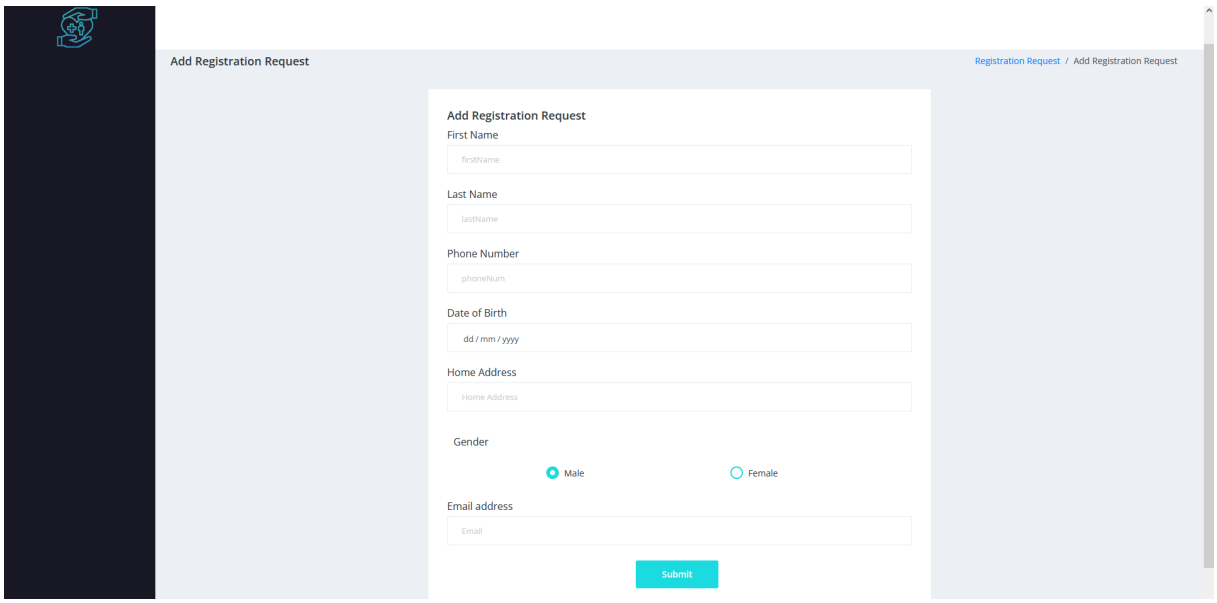

Figure 4.24: Visitor Send Request Portal interface.

4.8.3 Admin Owner Dashboard

This is the Application owner(Admin Owner) page, which shows that he has the access to add a patient, Health worker or admin, and viewing their user lists, plus he can accept or reject the visitors requests, the figure 4.25 displays the Homepage interface.

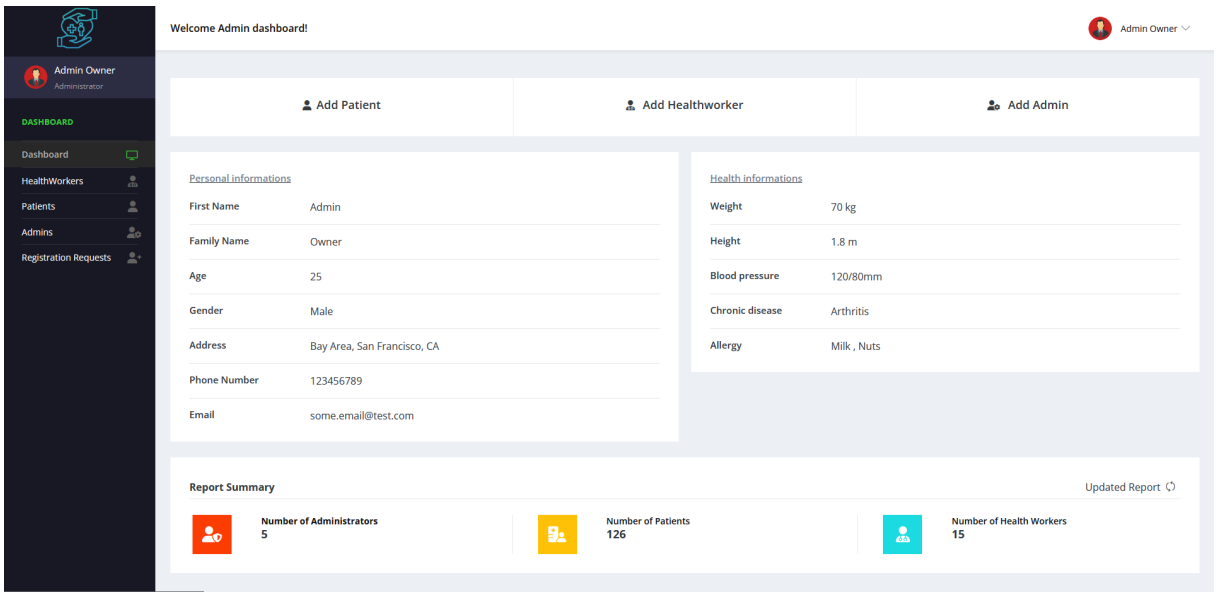

Figure 4.25: Admin Owner Dashboard interface.

4.8.4 Admin Dashboard

The Admin can do most things the AdminOwner can do except registering new admins and managing visitors requests, the figure 4.26 presents the Homepage interface.

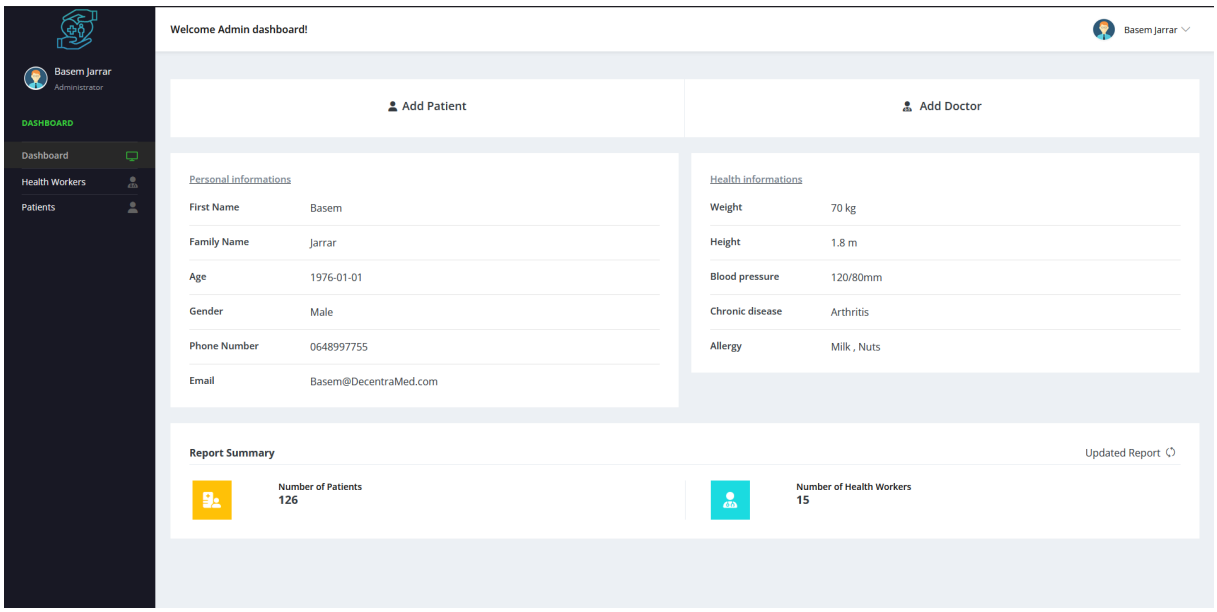

Figure 4.26: Admin Dashboard interface.

4.8.5 Add Patient Page

Can be accessed by an Admin or the AdminOwner, it allows them to register a Patient to the system, the interface is similar to adding a Health worker or an Admin, the patient addition interface posed on the figure 4.27.

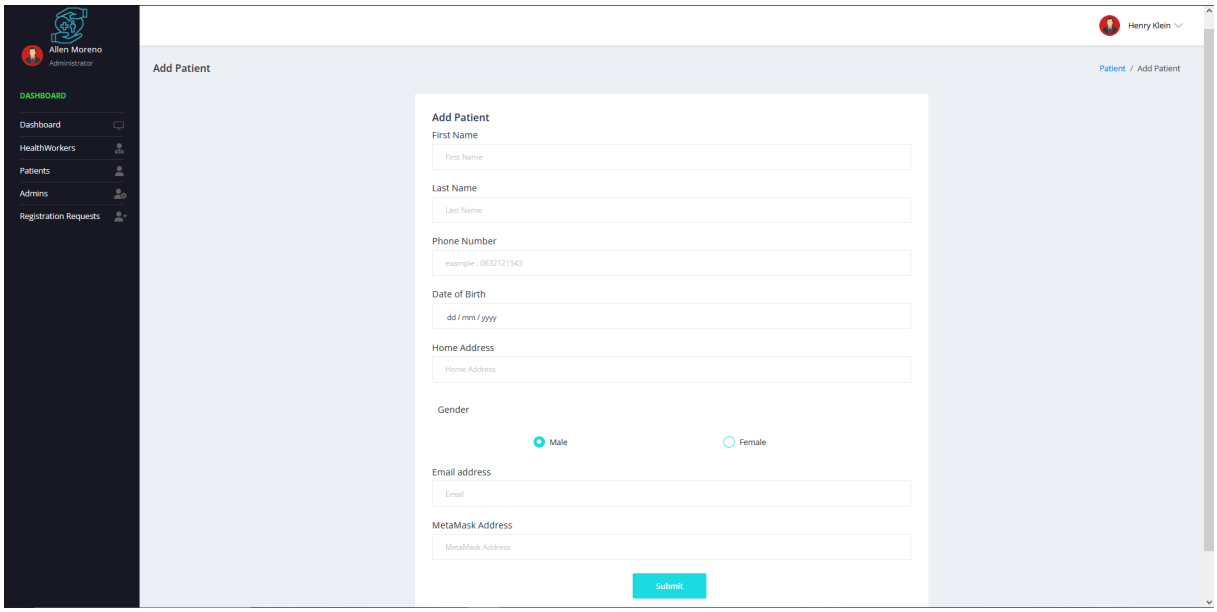

Figure 4.27: Add patient interface interface.

4.8.6 Patient Dashboard

Patient's Dashboard, where he can view his own personal and health information, and also can consult his records and change his contact info, figure 4.28 introduce Patient Dashboard interface.

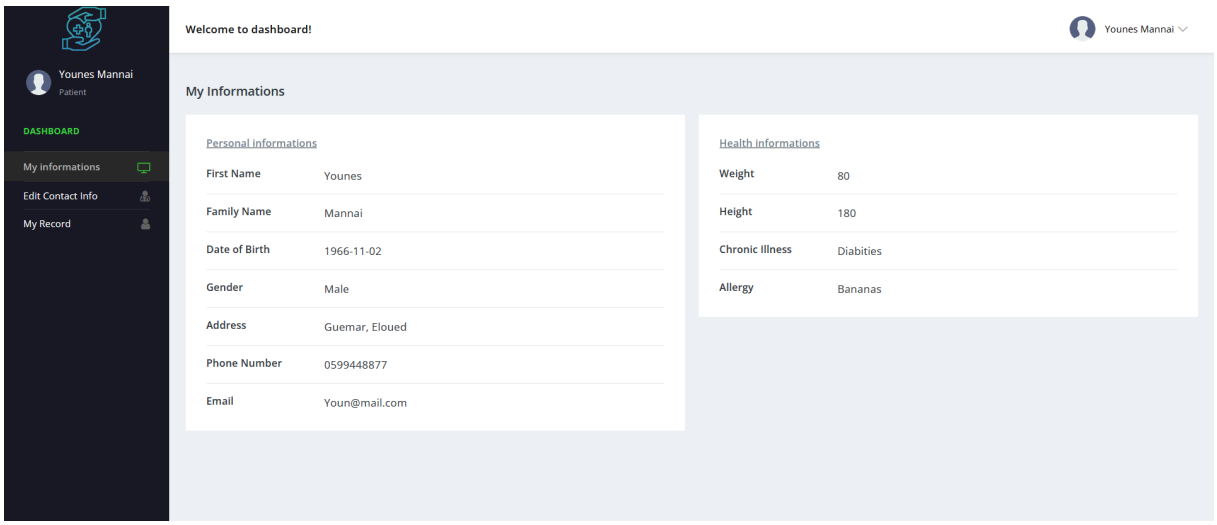

Figure 4.28: Patient Dashboard interface.

4.8.7 Patient Record Page

This page is only for the patient, it allows him to view his own record, figure 4.29 shows the patient addition interface.

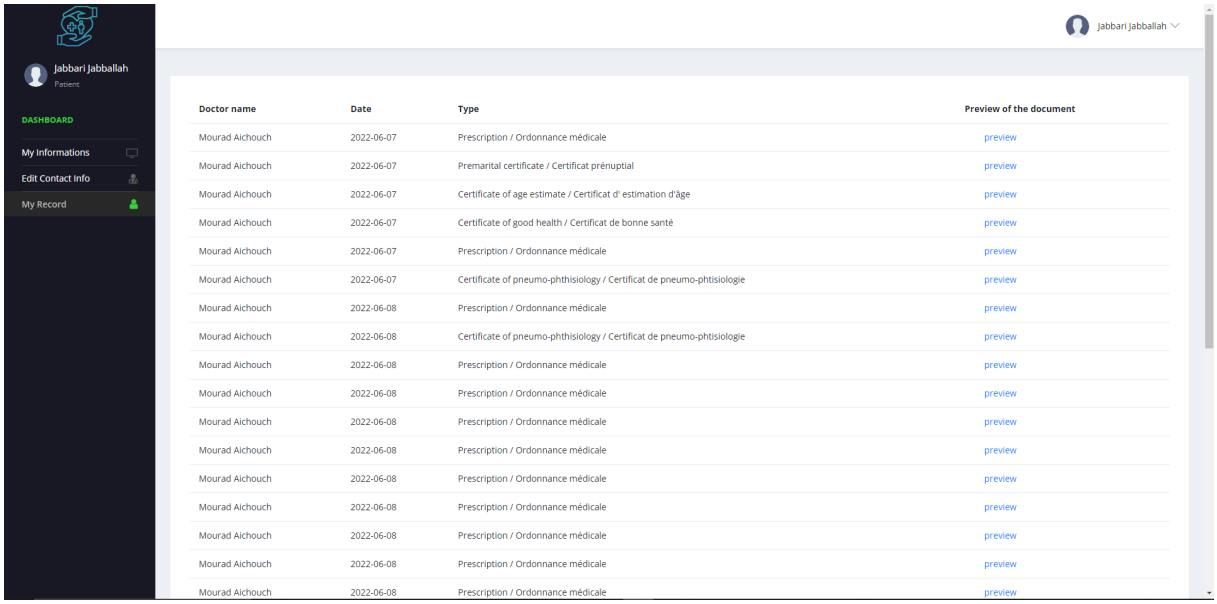

Figure 4.29: Patient view record interface.

4.8.8 Health worker Dashboard

Health worker Dashboard displays his own personal information and change his contact info, he can also consult a patient's information, figure 4.30 presents Health Worker Dashboard interface.

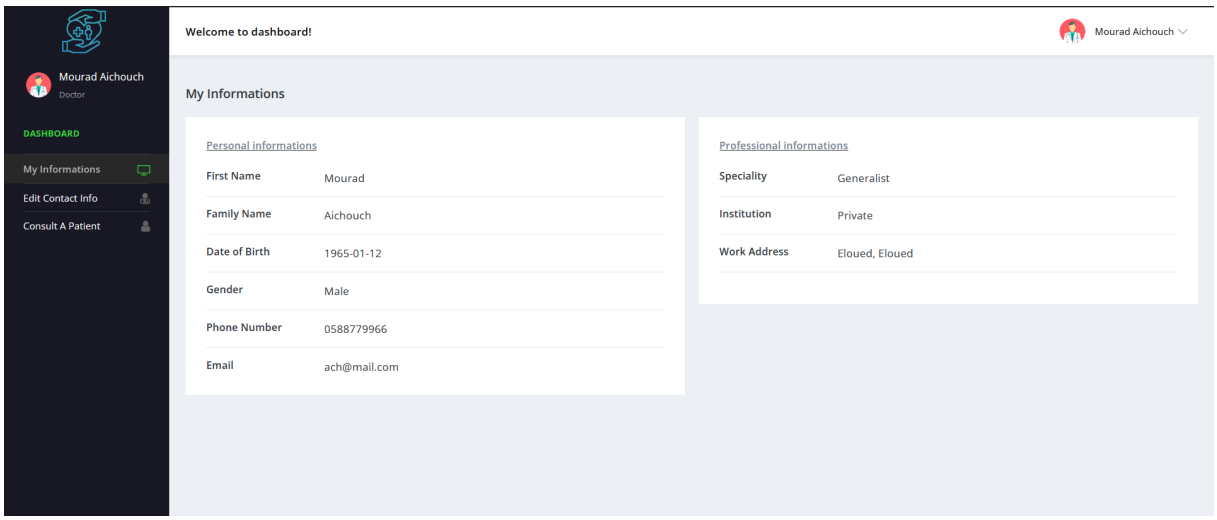

Figure 4.30: Health Worker Dashboard interface.

4.8.9 Consult Patient Record Page

Can only be accessed by Health workers, this page allows consultation of any patient's health record, providing the health worker has access to the patient's address, The figure 4.31 illustrates the consult Patient Record interface.

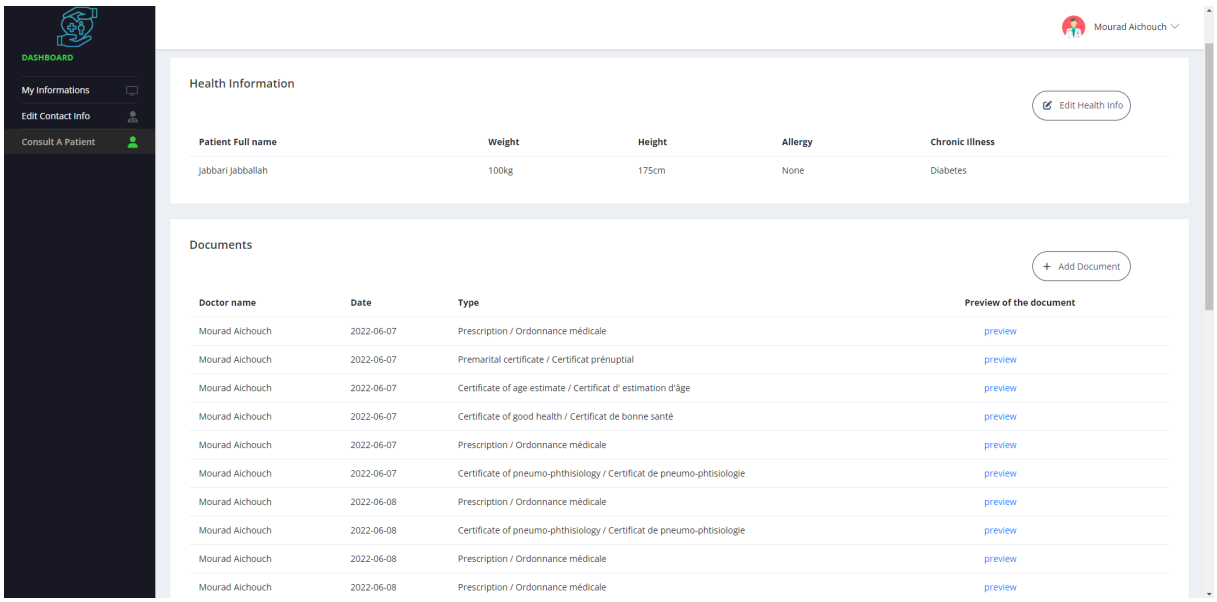

Figure 4.31: Consult Patient Record interface.

4.8.10 Add document patient record Page

can only be accessed from the Consult Patient Record Page, therefor only a Health Worker with the patient's address can access it, it allows the Health worker to add a document into the patient's health record, The figure 4.32 displays the Add document patient record.

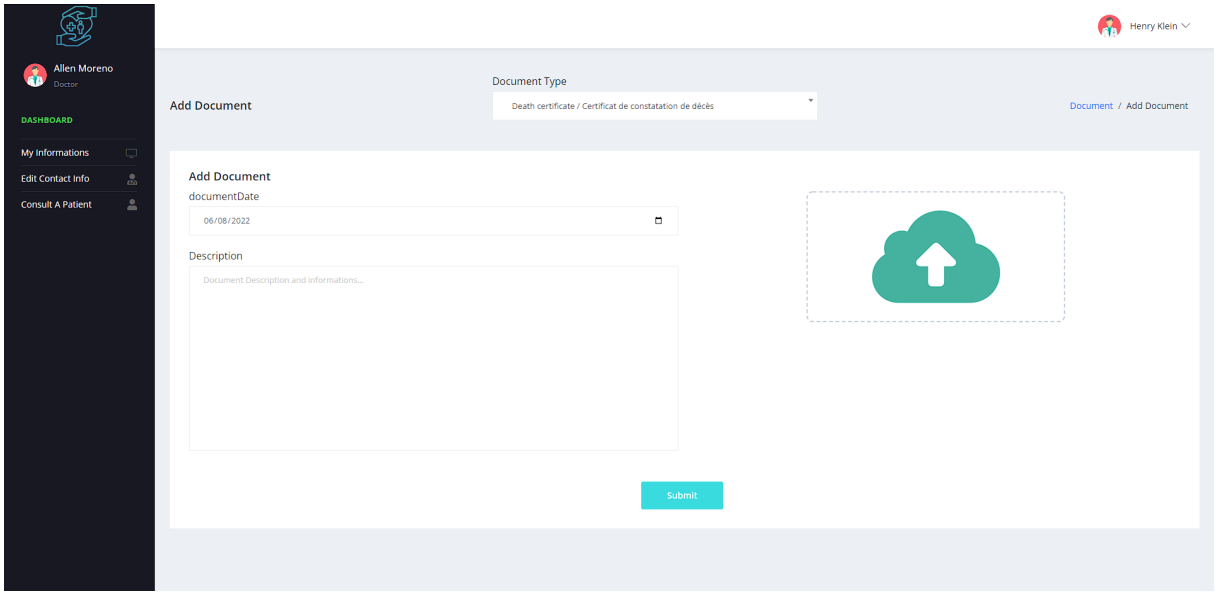

Figure 4.32: Add document patient record interface.

If the health worker prefers to write a prescription rather than submit a document, he can change the document type to prescription form.

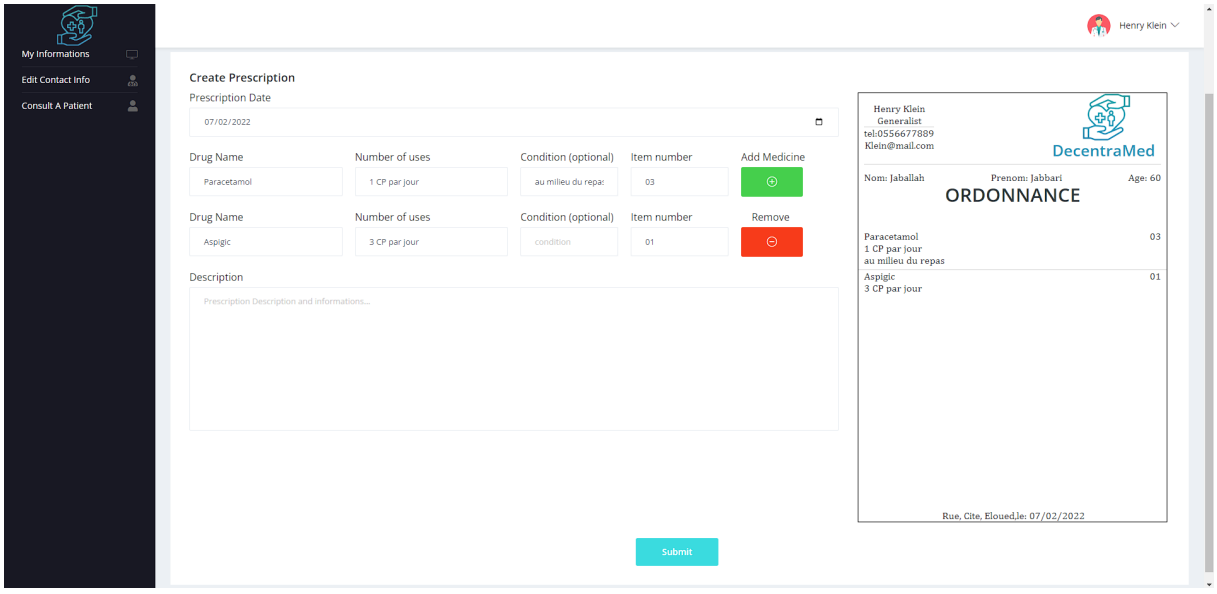

Figure 4.33: Add prescription patient record interface.

4.9 Conclusion

This chapter presents and explains the implementation of the website and its different features that enable users to interact with it with more privacy.

General Conclusion:

The goal in this thesis was to design and develop a new Electronic Health Record based on Blockchain technology.

The Dapp was built using smart contracts that store electronic health records with the help of an IPFS server as a document storage method. it makes use of the blockchain characteristics to guarantee the immutability of the patients record.

The work started by introducing Blockchain, explaining its characteristics, structure and types, then it presented it's advantages as well as it's disadvantages.

The second chapter discussed the medical field, the need, and the advantages that blockchain could provide into this field, because of the sensitive nature of the information in this field, including real life examples and use cases that have been tested.

The third chapter presents the software design phase, by collecting user needs, and then finishing by a detailed design of the software.

The Fourth chapter involved the implementation details and the presentation of the main user interfaces.

Perspectives: the projects could be improved in various ways, and can have limitless enhancements added, such as:

- Mining data collected by the website using AI to understand medical records and disease behavior.
- Add a scannable code(a QR code or by using the chifa card) that can added to identify patients, this way only by having the physical card can you modify patient info.
- Mining data collected by the website using the AI to understand medical records and disease behavior.

Bibliography

[1] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[2] Dave Bayer, Stuart Haber, and W Scott Stornetta. Improving the efficiency and reliability of digital time-stamping. In *Sequences Ii*, pages 329–334. Springer, 1993.

[3] Mohammad Jabed Morshed Chowdhury, Alan Colman, Muhammad Ashad Kabir, Jun Han, and Paul Sarda. Blockchain versus database: a critical analysis. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1348–1353. IEEE, 2018.

[4] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71, 2016.

[5] Beyhan Adanur Dedeturk, Ahmet Soran, and Burcu Bakir-Gungor. Blockchain for genomics and healthcare: a literature review, current status, classification and open issues. *PeerJ*, 9:e12130, 2021.

[6] JAKE FRANKENFIELD. Block (bitcoin block), 2022.

[7] Suyash Gupta, Sajjad Rahnama, Jelle Hellings, and Mohammad Sadoghi. Resilientdb: Global scale resilient blockchain fabric. *arXiv preprint arXiv:2002.00160*, 2020.

[8] Suyash Gupta and Mohammad Sadoghi. Blockchain transaction processing. *arXiv preprint arXiv:2107.11592*, 2(6-10):71, 2021.

[9] ADAM HAYES. Blockchain fields, 2022.

[10] Prasanth Kakarlapudi and Qusay Mahmoud. A systematic review of blockchain for consent management. *Healthcare*, 9:137, 02 2021.

[11] Skylar Kenney. The case for leveraging blockchain to improve the global health supply chain, 2021.

[12] D Massessi. Blockchain public/private key cryptography in a nutshell, 2018.

[13] André Henrique Mayer, Cristiano André da Costa, and Rodrigo da Rosa Righi. Electronic health records in a blockchain: A systematic review. *Health informatics journal*, 26(2):1273–1288, 2020.

[14] Daniel-Jesus Munoz, Denisa-Andreea Constantinescu, Rafael Asenjo, and Lidia Fuentes. Clinicappchain: A low-cost blockchain hyperledger solution for healthcare. In *International Congress on Blockchain and Applications*, pages 36–44. Springer, 2019.

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[16] Narayanan, Bonneau Arvind, Felten Josephn, Miller Edward, Goldfeder Andrew, and Steven. Blockchains: The great chain of being sure about things. *The Economist.*, 2016.

[17] AINO Nordgren, ELLEN Weckström, MINNA Martikainen, and OTHMAR M Lehner. Blockchain in the fields of finance and accounting: a disruptive technology or an overhyped phenomenon. *ACRN Journal of Finance and Risk Perspectives*, 8:47–58, 2019.

[18] Ryu Doojin Park Daehyeon. Blockchain in health insurance: Sharing medical information and preventing insurance fraud. *Korean J Financ Stud*, 48(4):417–447, 2019.

[19] Igor Radanović and Robert Likić. Opportunities for use of blockchain technology in medicine. *Applied health economics and health policy*, 16(5):583–590, 2018.

[20] Roshan Raj. Advantages and disadvantages of blockchain, 2021.

[21] Maik Schmeling. What is libra? understanding facebook's currency. Technical report, SAFE Policy Letter, 2019.

[22] David Schwartz, Noah Youngs, Arthur Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5(8):151, 2014.

[23] Alan Sherman, Farid Javani, Haibin Zhang, and Enis Golaszewski. On the origins and variations of blockchain technologies. *IEEE Security & Privacy*, 17:72–77, 01 2019.

[24] Edgar Mondragón Tenorio. Advantages and disadvantages of blockchain, 2021.

[25] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.