



SECURE SHARING OF MEDICAL RECORDS USING IPFS

R. Sathya¹, Ganesh Govind J², Mohamed Athif M³, Mohammed Zaid E⁴, Naveen Kumar SV⁵
SRM TRP Engineering College, Trichy

ABSTRACT

Health care professionals are required to maintain accurate health records of patients. Nowadays, health care providers use electronic health records (EHRs) as a key to the implementation of these goals and the delivery of quality care. However, there are technical and legal hurdles that prevent the adoption of these systems, such as concerns about performance and privacy issues. MedLink is built based on consortium blockchain technology. Every peer node hosts an instance of the distributed ledger consisting of EHRs and an example of chain code regulating the permissions of participants. We propose a permissioned blockchain-based system for EHR data sharing and integration. Each hospital will provide a blockchain node integrated with its own EHR system to form the blockchain network. A web-based interface will be used for patients and doctors to initiate EHR sharing transactions. We take a hybrid data management approach, where only management metadata will be stored on the chain. Actual EHR data, on the other hand, will be encrypted and stored in an IPFS cloud-based storage. The system uses public key infrastructure-based asymmetric encryption and digital signatures to secure shared EHR data.

1.1 CRYPTOGRAPHY

CHAPTER 1 INTRODUCTION

Cryptography is a technique for safeguarding data and correspondences using codes, so just those for whom the data is planned can peruse and handle it.

In software engineering, cryptography alludes to get data and correspondence procedures from numerical ideas and a bunch of rule-based computations called calculations, to change messages in manners that are difficult to translate. These deterministic calculations are utilized for cryptographic key age, advanced marking, confirmation to safeguard information security, web perusing on the web and secret interchanges, for example, charge card exchanges and email.

Secrecy. The data can't be perceived by anybody for whom it was accidental.

Integrity. The data can't be changed away or travel among source and expected collectors without the modification being recognized.

Non-disavowal. The maker/source of the data can't deny at a later stage their goals in the creation or transmission of the data.

Verification. The source and recipient can affirm each other's character and the beginning/objective of the data.

1.2 ENCRYPTION AND DECRYPTION

Encryption in network safety is the transformation of information from a meaningful organization into an encoded design. Encoded information must be perused or handled after it's been decoded.

Encryption is the essential structure block of information security. It is the least difficult and most significant method for guaranteeing a PC framework's data can't be taken

and perused by somebody who needs to involve it for vindictive purposes.

Information security encryption is broadly utilized by individual clients and enormous enterprises to safeguard client data sent between a program and a server. That data could incorporate everything from installment information to individual data.

Decoding is a Cyber Security method that makes it more challenging for programmers to block and peruse the data they're not permitted to do. It is changing scrambled or encoded information or text back to unique plain configuration individuals can undoubtedly peruse and comprehend from PC applications. This is the converse of encryption, which requires coding information to make it garbled for all, however just those with matching Decryption keys can understand it.

Despite the fact that encryption safeguards the information, beneficiaries should have the right Decryption or interpreting devices to get to the first subtleties. What Decryption does is decode the information, which should be possible physically, consequently, utilizing the best Decryption programming, extraordinary keys, passwords, or codes. This deciphers incomprehensible or unintelligible information into unique message records, email messages, pictures, client information, and registries that clients and PC frameworks can peruse and decipher.

1.3 BLOCKCHAIN TECHNOLOGY

Blockchain technology is a structure that stores transactional records, also known as the block, of the public in several databases, known as the “chain,” in a network connected through peer-to-peer nodes. Typically, this storage is referred to as a ‘digital ledger.’

Every transaction in this ledger is authorized by the digital signature of the owner, which authenticates the transaction and safeguards it from tampering. Hence, the information the digital ledger contains is highly secure.

In simpler words, the digital ledger is like a Google spreadsheet shared among numerous computers in a network, in which, the transactional records are stored based on actual purchases. The fascinating angle is that anybody can see the data, but they can't corrupt it.

Record keeping of data and transactions are a crucial part of the business. Often, this information is handled in house or passed through a third party like brokers, bankers, or lawyers increasing time, cost, or both on the business. Fortunately, Blockchain avoids this long process and facilitates the faster movement of the transaction, thereby saving both time and money.

131 HOW DOES THIS TECHNOLOGY WORK?

Blockchain is a combination of three leading technologies:

- Cryptographic keys
- A peer-to-peer network containing a shared ledger
- A means of computing, to store the transactions and records of the network

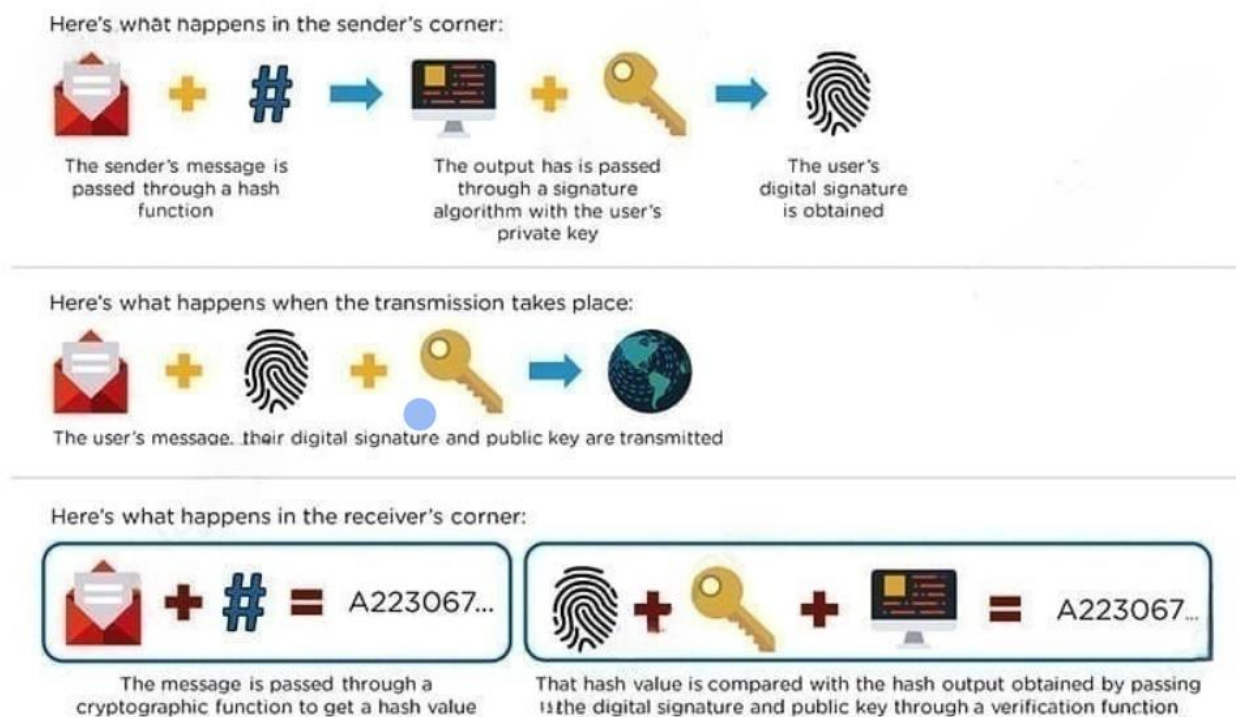
Cryptography keys consist of two keys – A private key and a public key. These keys help in performing successful transactions between two parties. Each individual has these two keys, which they use to produce a secure digital identity reference. This secured identity is the most important aspect of Blockchain technology. In the world of cryptocurrency, this identity is referred to as a 'digital signature' and is used for authorizing and controlling transactions.

The digital signature is merged with the peer-to-peer network; a large number of individuals who act as authorities use the digital signature in order to reach a consensus on transactions, among other issues. When they authorize a deal, it is certified by a mathematical verification, which results in a successful secured transaction between the two network-connected parties. So to sum it up, Blockchain users employ cryptography keys to perform different types of digital interactions over the peer-to-peer network.

132 HASH ENCRYPTPTIONS

Blockchain uses cryptography (see definition of "cryptography" above) to ensure that all the data in the blocks is kept secure from unauthorized access and is not altered. Blockchain uses SHA-256 for encryption. SHA-256 is one of the strongest hash functions available. This cryptographic hash algorithm generates an almost unique 256-bit signature for a text. Blockchain also uses digital signatures to validate users.

Each user has a public and private key. The public key is used to identify the user uniquely, and the private key gives the user access to everything in the account. In the process from the sender's side, the sender's message is passed through a hash function; then, the output is passed through a signature algorithm with the user's private key, then the user's digital signature is obtained. In the transmission, the user's message, digital signature, and public key are transmitted.



As mentioned, each block in a blockchain uses SHA-256 to encrypt and therefore secure the data. Every block has four fields:

- Previous hash—this field stores the hash of the previous block in the Blockchain
- Transaction details—this field contains information regarding several transactions
- Nonce—this field contains a random value (the nonce value) whose sole purpose is to act as a variate for the hash value
- Hash address—this field contains the unique identification of the block; it is a hex value of 64 characters, both letters, and numbers, obtained by using the SHA-256 algorithm

The first three values (previous hash, transaction details, and nonce) are passed through a hashing function to produce the fourth value, the hash address of that particular block.

133 PROOF OF WORK

In a Blockchain, each block consists of 4 main headers.

- Previous Hash: This hash address locates the previous block.
- Transaction Details: Details of all the transactions that need to occur.
- Nonce: An arbitrary number given in cryptography to differentiate the block's hash address.
- Hash Address of the Block: All of the above (i.e., preceding hash, transaction details, and nonce) are transmitted through a hashing algorithm. This gives an output containing a 256-bit, 64 character length value, which is called the unique 'hash address.' Consequently, it is referred to as the hash of the block.
- Numerous people around the world try to figure out the right hash value to meet a pre-determined condition using computational algorithms. The transaction completes when the predetermined condition is met. To put it more plainly, Blockchain miners attempt to solve a mathematical puzzle, which is referred to as a proof of work problem. Whoever solves it first gets a reward.

1.4 ETHEREUM AND ETHHASH

Originally conceived by Vitalik Buterin, Ethereum can be characterized as being an open technology platform upon which developers can build and launch decentralized applications (or dapps) using smart contract technology. Ethereum's native asset is known as ether, it serves as a fee for computations that are executed on the network. That fee is calculated in gas, but is paid for in ether. Ether also functions as an incentive mechanism for cryptocurrency miners to secure the network. Underpinning the entire protocol is the Ethereum algorithm for proof of work mining, known as Ethash.

Proof of work mining functions as a tool that is used to secure distributed networks and process blocks of transactions on the blockchain. Proof of work mining involves taking data from a block header to form an input, and repeatedly hashing that input using a cryptographic hashing algorithm. This produces an output of a fixed length, which represents the hash value. Miners will hash variations of the input data by including a different nonce each time the input data is entered into the algorithm. In the case of Ethereum, the algorithm used for this process is Ethash.

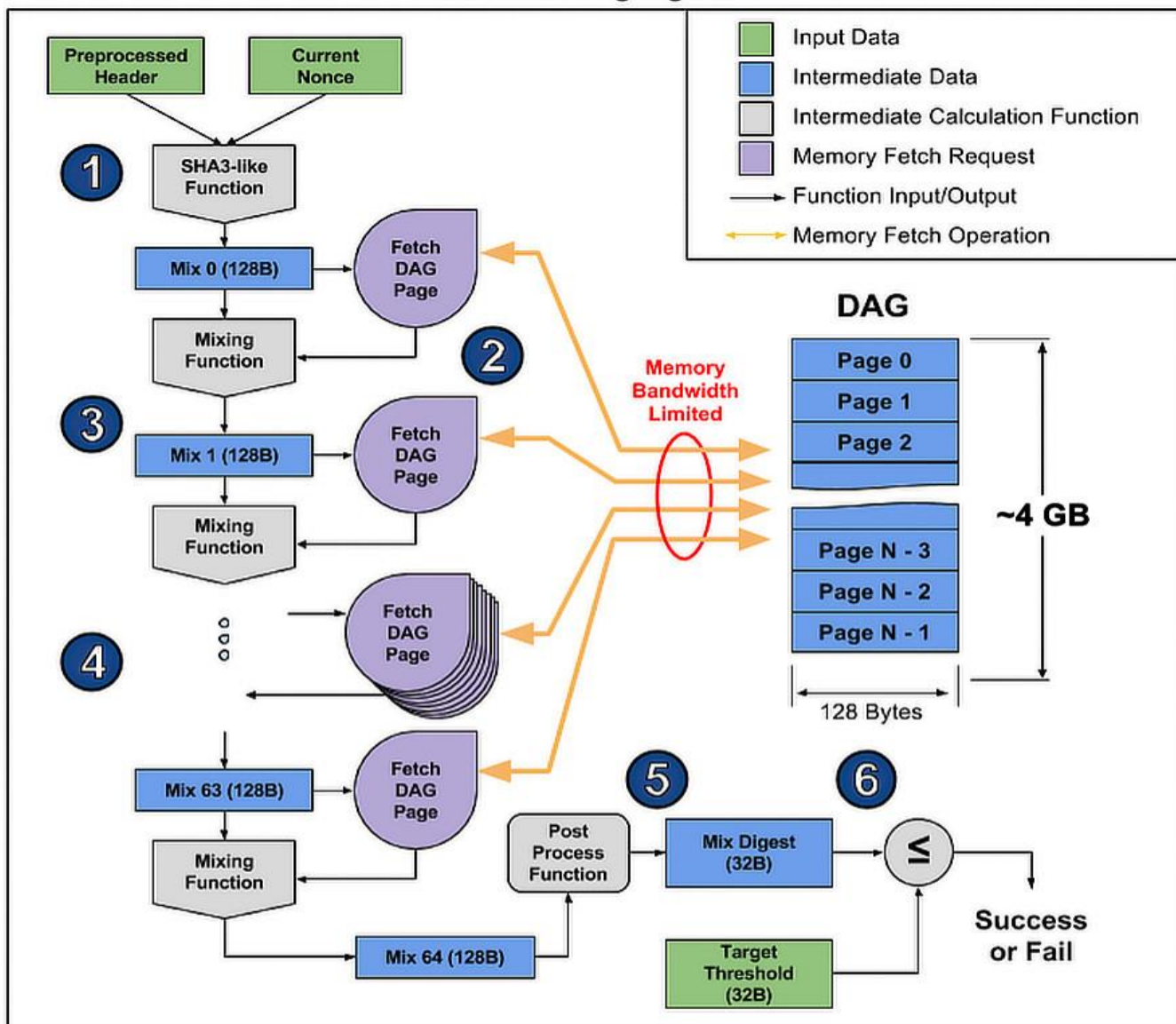
Ethash is a proof of work algorithm that is a modified version of a precursor algorithm known as Dagger-Hashimoto. With Ethash, the output formed in the hashing process must result in a hash value that is below a certain threshold. This concept is known as difficulty, and it involves the Ethereum network increasing and decreasing the threshold in order to control the rate at which blocks are mined on the network. If the rate at which blocks are found increases, then the network will automatically increase the difficulty i.e. it will lower the network threshold so that the number of valid hashes capable of being found also decreases. Conversely, if the rate of discovered blocks decreases, the network threshold will increase to produce a higher number of correct hash values that could be found. With the Ethereum algorithm, the difficulty dynamically adjusts such that, on average, one block is produced by the network every 12 seconds.

A miner that successfully discovers a block that can be added to the blockchain receives:

- A static block reward consisting of 3 ether.

- All of the gas spent within the block, in other words, all of the gas that has been consumed by the execution of all the transactions that are contained within the block. This gas cost is awarded to the miner's account as part of the proof of work mining process.
- An extra reward for including uncles as part of the block. Uncles that are included in a block by a proof of work miner receive 7/8 of the static block reward, which is 2.625 ether.

Ethash Hashing Algorithm

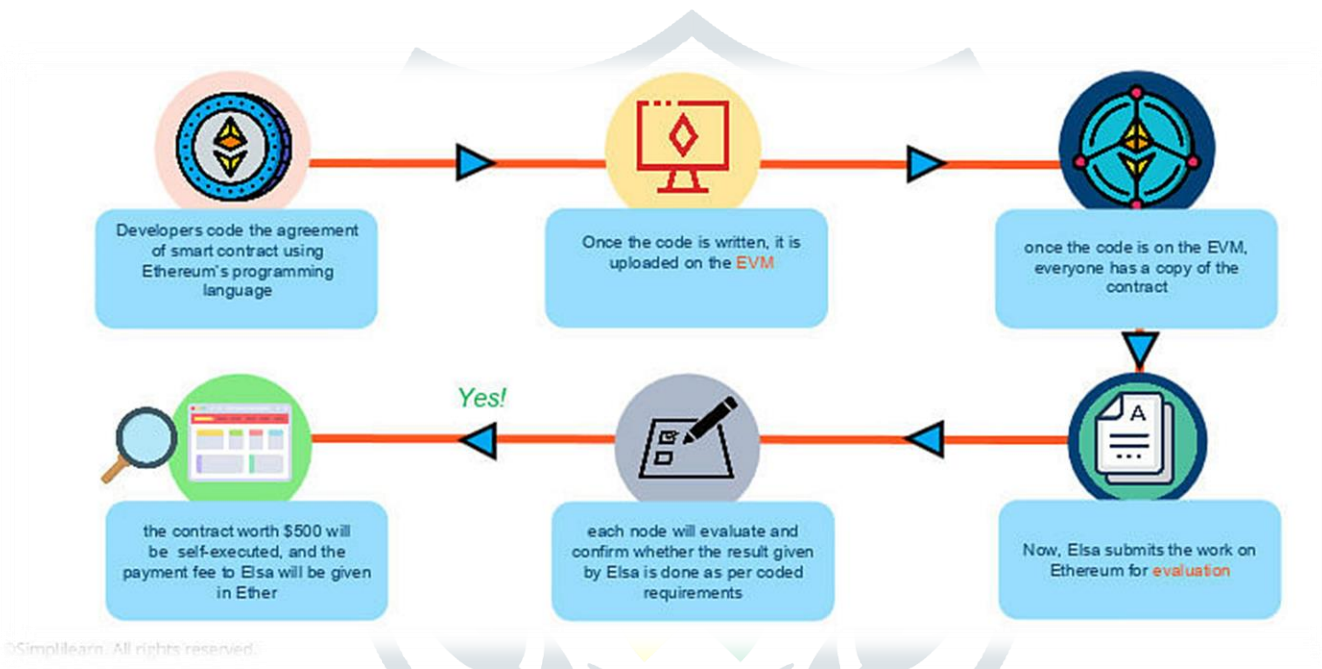


1.5 SMART CONTRACTS

Smart contracts are computer programs or protocols for automated transactions that are stored on a blockchain and run in response to meeting certain conditions. In other words, smart contracts automate the execution of agreements so that all participants can ascertain the outcome as soon as possible without the involvement of an intermediary or time delay.

- Smart contracts are self-executing contracts in which the contents of the buyer-seller agreement are inscribed directly into lines of code.
- According to Nick Szabo, an American computer scientist who devised a virtual currency called "Bit Gold" in 1998, Smart contracts are computerized transaction protocols that execute contract conditions.
- Using it makes the transactions traceable, transparent, and irreversible.

15.1 HOW DO SMART CONTRACTS WORK?



A smart contract is a sort of program that encodes business logic and operates on a dedicated virtual machine embedded in a blockchain or other distributed ledger.

Step 1: Business teams collaborate with developers to define their criteria for the smart contract's desired behavior in response to certain events or circumstances.

Step 2: Conditions such as payment authorization, shipment receipt, or a utility meter reading threshold are examples of simple events.

Step 3: More complex operations, such as determining the value of a derivative financial instrument, or automatically releasing an insurance payment, might be encoded using more sophisticated logic.

Step 4: The developers then use a smart contract writing platform to create and test the logic. After the application is written, it is sent to a separate team for security testing.

Step 5: An internal expert or a company that specializes in vetting smart contract security could be used.

Step 6: The contract is then deployed on an existing blockchain or other distributed ledger infrastructure once it has been authorized.

Step 7: The smart contract is configured to listen for event updates from an "oracle," which is effectively a cryptographically secure streaming data source, once it has been deployed.

Step 8: Once it obtains the necessary combination of events from one or more oracles, the smart contract executes.

152 BENEFITS OF SMART CONTRACTS

Accuracy, Speed, and Efficiency

- The contract is immediately executed when a condition is met.
- Because smart contracts are digital and automated, there is no paperwork to deal with, and
- No time was spent correcting errors that can occur when filling out documentation by hand.

Trust and Transparency

- There's no need to worry about information being tampered with for personal gain because there's no third party engaged and
- Encrypted transaction logs are exchanged among participants.

Security

- Because blockchain transaction records are encrypted, they are extremely difficult to hack.
- Furthermore, because each entry on a distributed ledger is linked to the entries before and after it, hackers would have to change the entire chain to change a single record.

Savings

- Smart contracts eliminate the need for intermediaries to conduct transactions, as well as the time delays and fees that come with them.

CHAPTER 2 LITERATURE SURVEY

2.1 Title: BSF-EHR: Blockchain Security Framework for Electronic Health Records of Patients, 2021

Authors: Ibrahim Abunadi and Ramasamy Lakshmana Kumar

In the current epoch of smart homes and cities, personal data such as patients' names, diseases and addresses are often violated. This is frequently associated with the safety of the electronic health records (EHRs) of patients. EHRs have numerous benefits worldwide, but at present, EHR information is subject to considerable security and privacy issues. This paper proposes a way to provide a secure solution to these issues. Previous sophisticated techniques dealing with the protection of EHRs usually make data inaccessible to patients. These techniques struggle to balance data confidentiality, patient demand and constant interaction with provider data. Blockchain technology solves the above problems since it distributes information in a transactional and decentralized manner. The usage of blockchain technology could help the health sector to balance the accessibility and privacy of EHRs. This paper proposes a blockchain security framework (BSF) to effectively and securely store and keep EHRs. It presents a safe and proficient means of acquiring medical information for doctors, patients and insurance agents while protecting the patient's data. This work aims to examine how our proposed framework meets the security needs of doctors, patients and third parties

and how the structure addresses safety and confidentiality concerns in the healthcare sector. Simulation outcomes show that this framework efficiently protects EHR data.

2.2 Title: Healthchain: A Privacy Protection System for Medical Data Based on Blockchain, 2021

Authors: Baocheng Wang and Zetao Li

With the dramatically increasing deployment of the Internet of Things (IoT), remote monitoring of health data to achieve intelligent healthcare has received great attention recently. However, due to the limited computing power and storage capacity of IoT devices, users' health data are generally stored in a centralized third party, such as the hospital database or cloud, and make users lose control of their health data, which can easily result in privacy leakage and single-point bottleneck. In this paper, we propose Healthchain, a large-scale health data privacy preserving scheme based on blockchain technology, where health data are encrypted to conduct fine-grained access control. Specifically, users can effectively revoke or add authorized doctors by leveraging user transactions for key management. Furthermore, by introducing Healthchain, both IoT data and doctor diagnosis cannot be deleted or tampered with so as to avoid medical disputes. Security analysis and experimental results show that the proposed Healthchain is applicable for smart healthcare system.

2.3 Title: A Machine learning-based privacy-preserving fair data trading in big data market, 2019

Authors: Zhao, Yanqi, Yong Yu, Yannan Li, Gang Han, and Xiaojiang Du

In the era of big data, the produced and collected data explode due to the emerging technologies and applications that pervade everywhere in our daily lives, including internet of things applications such as smart home, smart city, smart grid, e-commerce applications and social network. Big data market can carry out efficient data trading, which provides a way to share data and further enhances the utility of data. However, to realize effective data trading in big data market, several challenges need to be resolved. The first one is to verify the data availability for a data consumer. The second is privacy of a data provider who is unwilling to reveal his real identity to the data consumer. The third is the payment fairness between a data

provider and a data consumer with atomic exchange. In this paper, we address these challenges by proposing a new blockchain-based fair data trading protocol in big data market. The proposed protocol integrates ring signature, double-authentication-preventing signature and similarity learning to guarantee the availability of trading data, privacy of data providers and fairness between data providers and data consumers. We show the proposed protocol achieves the desirable security properties that a secure data trading protocol should have. The implementation results with Solidity smart contract demonstrate the validity of the proposed blockchain-based fair data trading protocol.

2.4 Title: Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage, 2018.

Authors: Shen, Wenting, Jing Qin, Jia Yu, Rong Hao, and Jiankun Hu.

With cloud storage services, users can remotely store their data to the cloud and realize the data sharing with others. Remote data integrity auditing is proposed to guarantee the integrity of the data stored in the cloud. In some common cloud storage systems such as the electronic health records system, the cloud file might contain some sensitive information. The sensitive information should not be exposed to others when the cloud file is shared. Encrypting the whole shared file can realize the sensitive information hiding, but will make this shared file unable to be used by others. How to realize data sharing with sensitive information hiding in remote data integrity auditing still has not been explored up to now. In order to address this problem, we propose a remote data integrity auditing scheme that realizes data sharing with sensitive information hiding in this paper. In this scheme, a sanitizer is used to sanitize the data blocks corresponding to the sensitive information of the file and transforms these data blocks' signatures into valid ones for the sanitized file. These signatures are used to verify the integrity of the sanitized file in the phase of integrity auditing. As a result, our scheme makes the file stored in the cloud able to be shared and used by others on the condition that the sensitive information is hidden, while the remote data integrity auditing is still able to be efficiently executed. Meanwhile, the proposed scheme is based on identity-based cryptography, which simplifies the complicated certificate management. The security analysis

and the performance evaluation show that the proposed scheme is secure and efficient.

2.5 Title: Blockchain-Based, Decentralised Access Control for IPFS, 2019

Authors: Mathis Steichen; Beltran Fiz; Robert Norvill; Wazen Shbair; Radu State

Large files cannot be efficiently stored on blockchain. On one hand side, the blockchain becomes bloated with data that has to be propagated within the blockchain network. On the other hand, since the blockchain is replicated on many nodes, a lot of storage space is required without serving an immediate purpose, especially if the node operator does not need to view every file that is stored on the blockchain. It furthermore leads to an increase in the price of operating blockchain nodes because more data needs to be processed, transferred and stored. IPFS is a file-sharing system that can be leveraged to more efficiently store and share large files. It relies on cryptographic hashes that can easily be stored on a blockchain. Nonetheless, IPFS does not permit users to share files with selected parties. This is necessary if sensitive or personal data needs to be shared. Therefore, this paper presents a modified version of the Interplanetary File system (IPFS) that leverages Ethereum smart contracts to provide access-controlled file sharing. The smart contract is used to maintain the access control list, while the modified IPFS software enforces it. For this, it interacts with the smart contract whenever a file is uploaded, downloaded or transferred. Using an experimental setup, the impact of the access controlled IPFS is analyzed and discussed.

2.6 Title: A First Look at Blockchain-based Decentralized Applications ,2019.

Authors: Kaidong Wu, Yun Ma, Gang Huang, Xuanzhe Liu

With the increasing popularity of blockchain technologies in recent years, blockchain-based decentralized applications (DApps for short in this paper) have been rapidly developed and widely adopted in many areas, being a hot topic in both academia and industry. Despite of the importance of DApps, we still have quite little understanding of DApps along with its

ecosystem. To bridge the knowledge gap, this paper presents the first comprehensive empirical study of blockchain-based DApps to date, based on an extensive dataset of 995 Ethereum DApps and 29,846,075 transaction logs over them. We make a descriptive analysis of the popularity of DApps, summarize the patterns of how DApps use smart contracts to access the underlying blockchain, and explore the worth-addressing issues of deploying and operating DApps. Based on the findings, we propose some implications for DApp users to select proper DApps, for DApp developers to improve the efficiency of DApps, and for blockchain vendors to enhance the support of DApps.

2.7 Title: Demystifying Cryptography behind Blockchains and a Vision for Post-Quantum Blockchains, 2020.

Authors: Auqib Hamid Lone, Roohie Naaz

One of the backbone technology of Blockchains is cryptography. In simpler terms cryptography is the mathematical art of secret writing, however it's applications are not limited to writing secret codes only. Essentially cryptography is used to achieve three main security goals namely confidentiality (in-formation must be hidden from the unintended users), integrity (information must be prevented from illicit modifications) and availability (information must be readily available to intended users at all times). In cryptography confidentiality is achieved by employing a technique called as Encryption, integrity is achieved by employing special one way functions called as Cryptographic Hash Functions and authentication (process for verifying identity of the sender) is achieved by employing a technique called as Digital Signatures. For better understanding about the working of Blockchain it is essential to understand cryptographic concepts on which Blockchain is built. This paper presents the brief overview of such cryptographic concepts. In particular this papers throws light on the cryptographic concepts used to build Bitcoin and Ethereum Blockchains. Furthermore this paper also throws light on the need and requirements of post-quantum cryptographic primitives for post-quantum Blockchains.

2.8 Title: Bootstrapping a blockchain based ecosystem for big data exchange, 2017.**Authors: Chen, Jinchuan, and Yunzhi Xue**

In recent years, data is becoming the most valuable asset. There are more and more data exchange markets on Internet. These markets help data owners publish their datasets and data consumers find appropriate services. However, different from traditional goods like clothes and food, data is a special commodity. For current data exchange markets, it is very hard to protect copyright and privacy. Moreover, maintaining data services requires special IT techniques, which is a difficult job for many organizations who own big datasets, such as hospitals, government departments, planetariums and banks. In this paper, we propose a decentralized solution for big data exchange. This solution aims at cultivating an ecosystem, inside which all participators can cooperate to exchange data in a peer-to-peer way. The core part of this solution is to utilize blockchain technology to record transaction logs and other important documents. Unlike existing data exchange markets, our solution does not need any third-parties. It also provides an convenient way for data owners to audit the use of data, in order to protect data copyright and privacy. We will explain the ecosystem, and discuss the technical challenges and corresponding solutions.

2.9 Title: Blockchain Interoperability: Towards a Sustainable Payment System,2022.**Authors: Divya Anand, Hani Moaiteq Aljahdali, Santos Gracia Villar**

The highly fragmented blockchain and cryptocurrency ecosystem necessitates interoperability mechanisms as a requirement for blockchain-technology acceptance. The immediate implication of interchain interoperability is automatic swapping between cryptocurrencies. We performed a systematic review of the existing literature on Blockchain interoperability and atomic cross-chain transactions. We investigated different blockchain interoperability approaches, including industrial solutions, categorized them and identified the key mechanisms used, and list several example projects for each category. We focused on the atomic transactions between blockchain, a process also known as atomic swap. Furthermore, we studied recent implementations along with architectural approaches for atomic swap and

deduced research issues and challenges in cross-chain interoperability and atomic swap. Atomic swap can instantly transfer tokens and significantly reduce the associated costs without using any centralized authority, and thus facilitates the development of a sustainable payment system for wider financial inclusion.

2.10 Title: Secure electronic health care (EHC) system using Blockchain technique, 2021

Authors: Ms. Nilima V. Pardakhe, Dr. V. M. Deshmukh

Information systems and computerization nowadays need very faster, secure & easier data analysis techniques. The modern healthcare systems are extremely complex, expensive and face problems concerning data privacy, security and integrity. However, better monitoring and management of electronic health records will reduce these issues regarding complexity and security. Blockchain with its decentralized and trustworthy nature has established massive potentials in healthcare sector. Primary problems in healthcare delivery include lack of data management, and how data can be made verifiable, immutable, and distributed. One of the key benefits of using blockchain technology in the healthcare database is because of its potential to update medical interoperability systems which provides better access to patient records, medication tracking, drug systems and hospital assets etc. Access to patient's medical histories is essential to correctly prescribe medication, with blockchain being able to dramatically enhance the healthcare services framework. In this work, various solutions to improve current healthcare system vulnerabilities using blockchain technology including frameworks and methods are addressed.

The main aim of the proposed research work is to design blockchain based framework for preserving & securing electronic medical healthcare data efficiently in comparison with conventional EHR systems. In addition, framework store medical prescription, laboratory report and emergency data which could encompass data regarding medical history of the patients, the medicine details in database for future use providing EHR system which includes scalable, secure and integral blockchain-based solution.

CHAPTER 3 PROBLEM DESCRIPTION

3.1 EXISTING METHOD

- In EHRs, all medical-related data are digitized and stored in a centralized server.
- Although cloud-based PHR management services are currently available, including Google Health, Apple Health, and Practice Fusion, these services arguably take data ownership out of the hands of the patient through third-party involvement.
- Third, patient-reported data do not always get recorded in a patient's EHRs since doctor-patient communication is not always possible, which impacts the quality of care. Therefore, a kind of patient-reporting mechanism is needed for precision medicine.
- After all, 11,581,616,452 records have been breached since 2005, and this has been reported through either government agencies or verifiable media sources. (<https://privacyrights.org/data-breaches>)

Disadvantages

- EHRs can get incorrect information if the EHR is not updated immediately when new information, such as when new test results come in.
- Selecting and setting up an EHR system and digitizing all paper records can take years.
- Maintaining an EHR system requires frequent updates. If your team doesn't keep up with this, your records can lose their accuracy and, as a result, their value.
- At 100%, you will need to optimize your EHR system over time to keep it running smoothly. This costs a lot of money and man-power.
- Other major challenges that EHR's face are interoperability and data privacy.

3.2 PROPOSED SYSTEM

- ‘MedLink’ is a DApp built on the Ethereum network using Solidity. Methods passed within the ‘Smart Contracts’ work as various functionalities within the GUI.
- The system consists of an admin whose wallet is registered first. Doctors can be created by the admin where each doctor has his/her own Private and Public Keys.
- Patients on the other hand will have their own unique wallets with a unique Public and Private Key.
- Patients can register themselves and then book an appointment based on the doctor of their preference/specialty.
- The doctors will get to know about any new appointments from their dashboard. Once the doctor consults the patient, he/she can provide details about the medication, tests and reports for a particular patient.
- All these transactions will be taken through the Ethereum network (Testnet) and the data uploaded will be distributed as caches via a peer-peer network of nodes called ‘IPFS’. (Domain: ipfs.co and local storage)

Advantages

- Users in a local network can communicate with each other, even if the Wide Area network is blocked for some reason.
- Fewer servers are required, as users viewing the content will help with the distribution by sharing it peer-to-peer.
- Content loads faster, but uses more bandwidth because you now need to contribute to the network.
- IPFS replacing HTTP would mean you can get a cache of pretty much the entire web with sufficient storage.
- Deduplication, because everything is addressed by a hash and integrity as files must match the appropriate hash.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 ARCHITECTURE

System architecture involves the high-level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability.

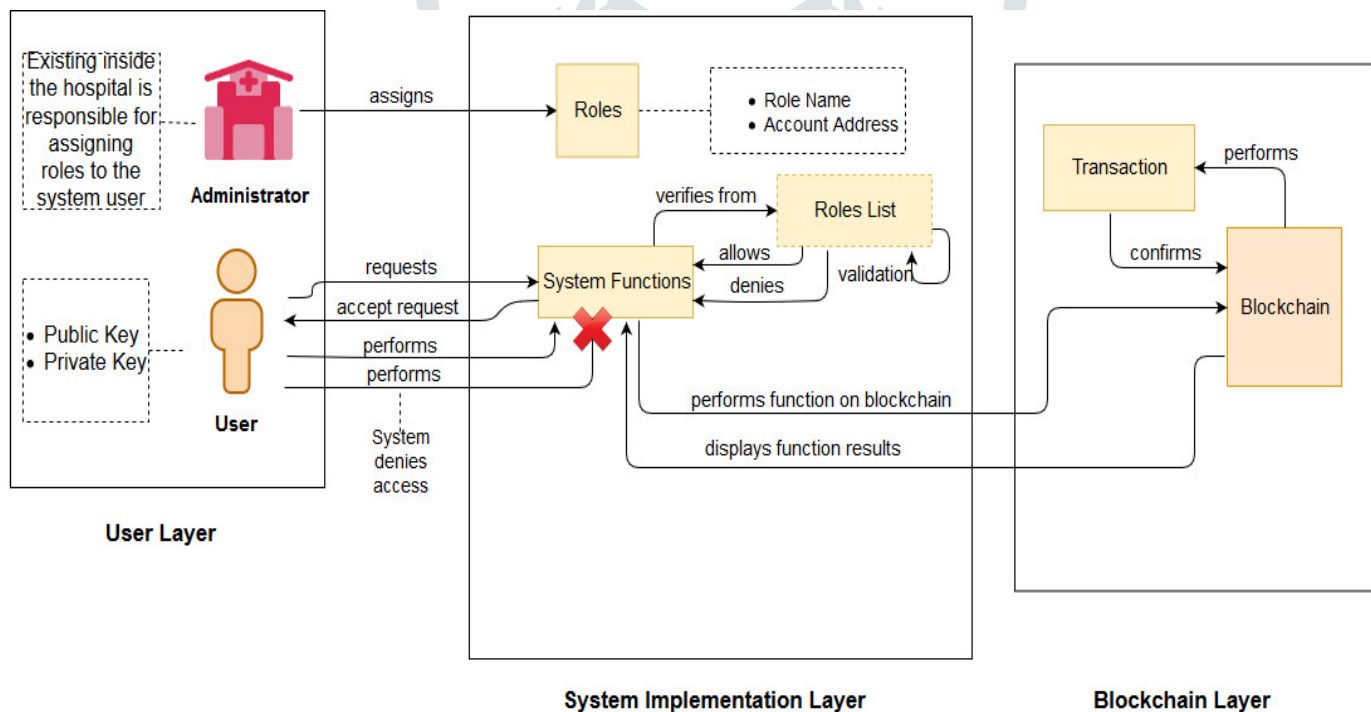


Fig 4.1 System Architecture

4.2 FLOW CHART

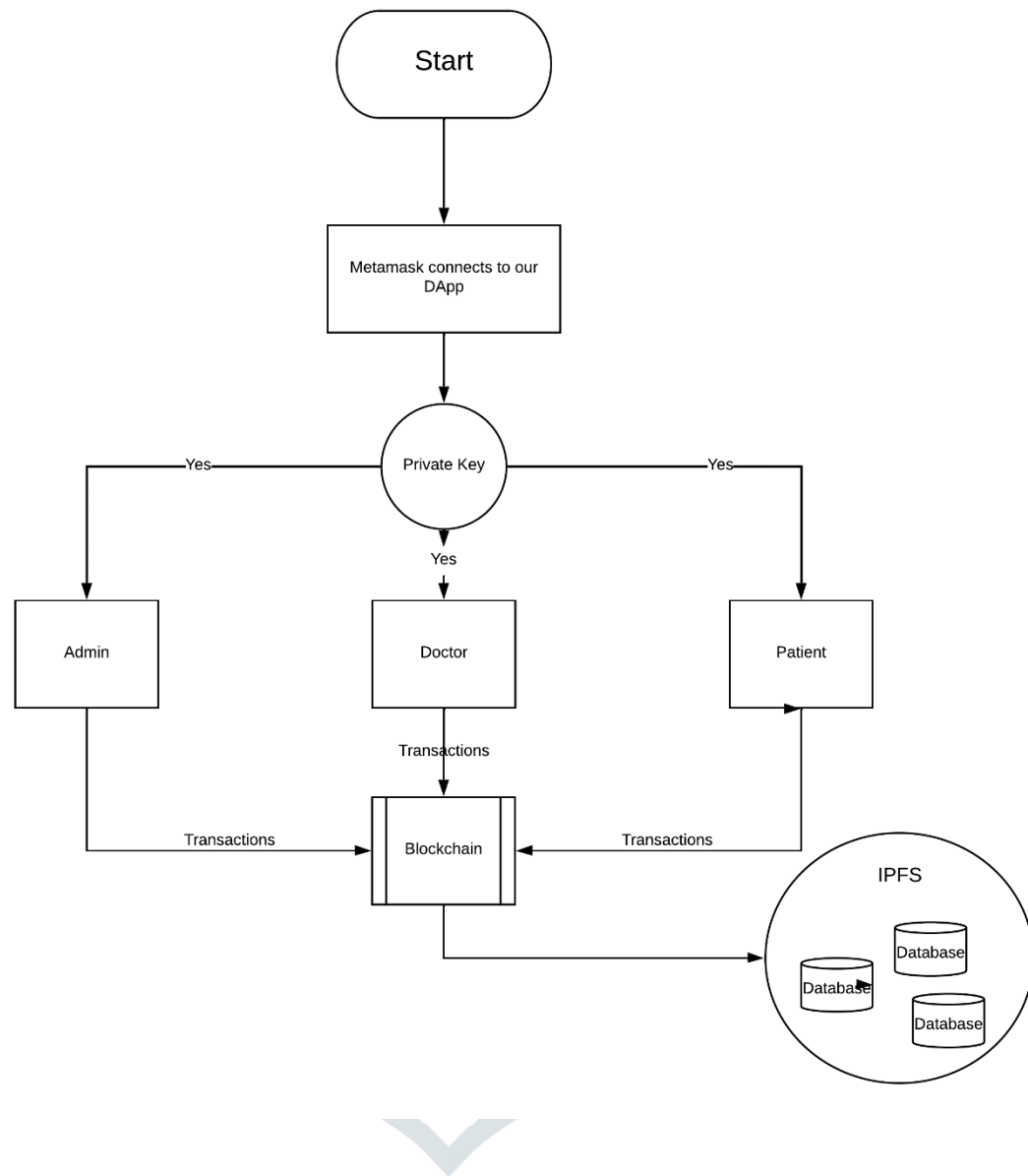


Fig 4.2 Flow Chart

4.3 UML DIAGRAM

4.3.1 Use Case Diagram

A use case is a list of steps, typically defining interactions between a role (known in Unified Modelling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human, an external system, or time. In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or

stakeholder goals. Use Case Diagram has actors like sender and receiver. Use cases show the activities handled by both sender and receiver.

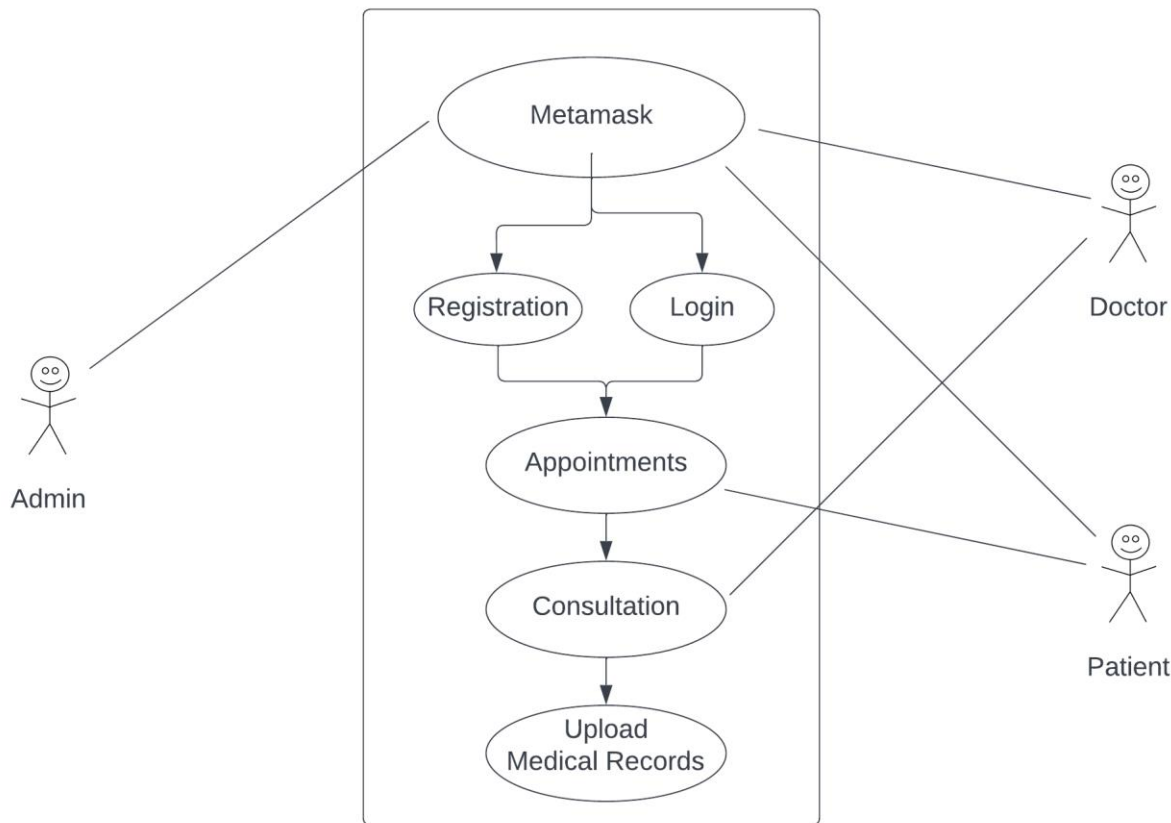


Fig 4.3.1 Use case Diagram

4.3.2 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes. Activity diagrams show the overall flow of control. The Activity diagram has an initial and final state. Then activities are mentioned between the states.

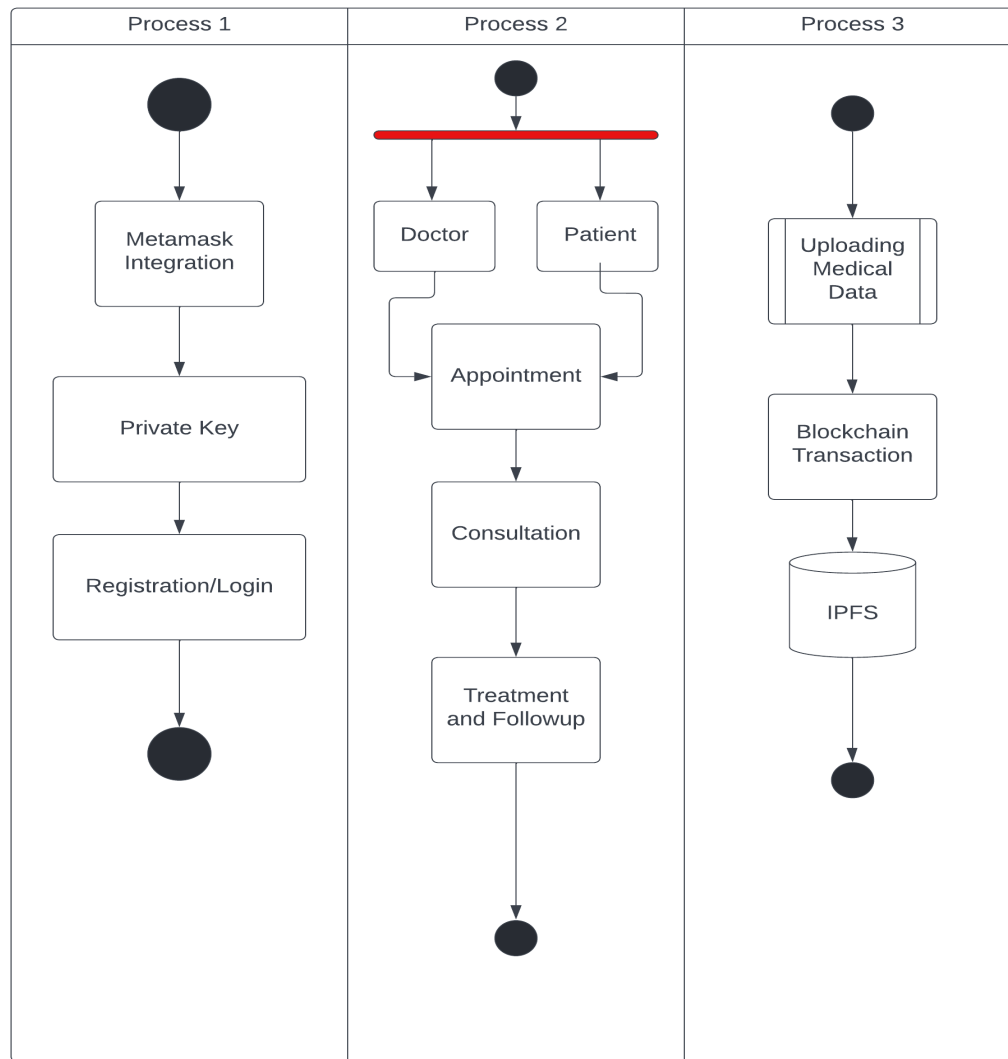


Fig 4.3.2 Activity Diagram

4.3.3 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. The Sequence diagram is sometimes called event trace diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines, different processes that live simultaneously and horizontal arrows. The messages exchanged between them. The Sequence diagram has three objects. The connection between the objects is mentioned using stimulus and self-stimulus.

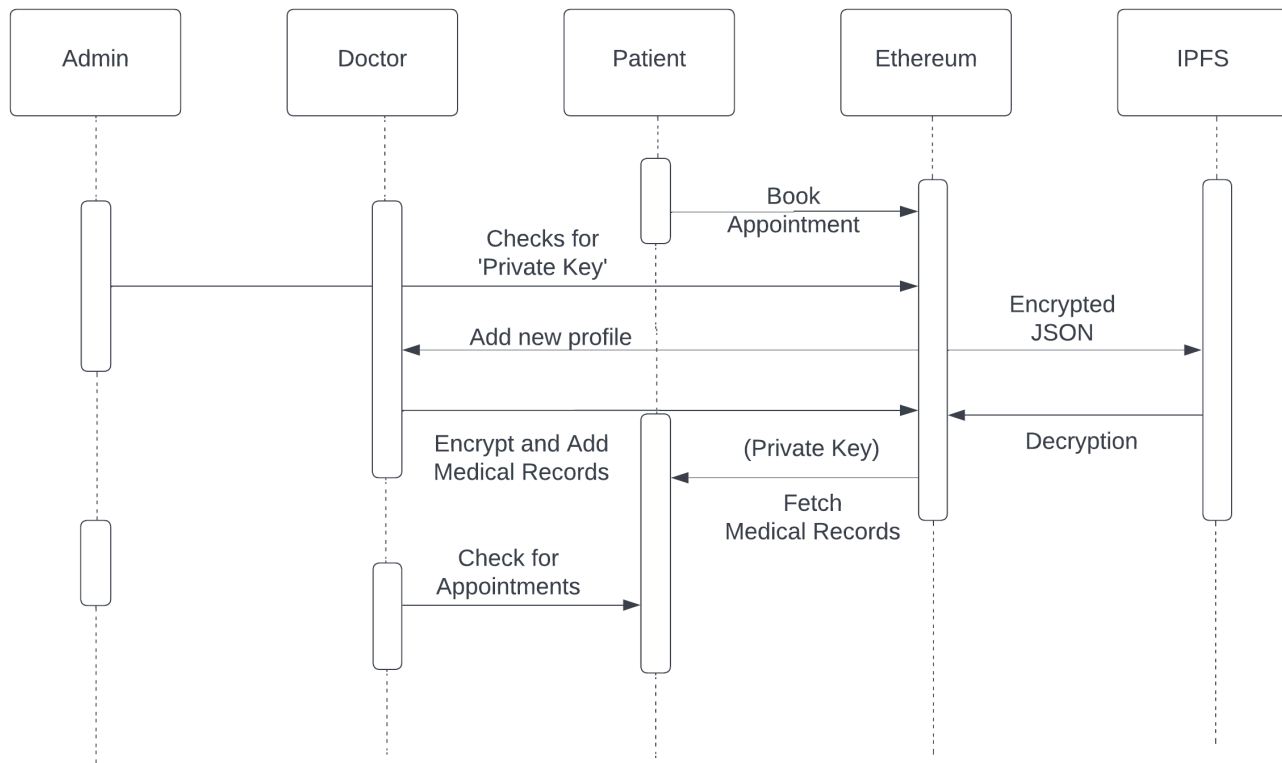


Fig 4.3.3 Sequence Diagram

4.3.4 COLLABORATION DIAGRAM

A collaboration diagram resembles a flowchart that portrays the roles, functionality, and behavior of individual objects as well as the overall operation of the system in real-time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing. The Collaboration diagram shows the collaborative connections between three objects like sender, receiver, and server. The Collaborative has self stimulus and also a connection between two objects.

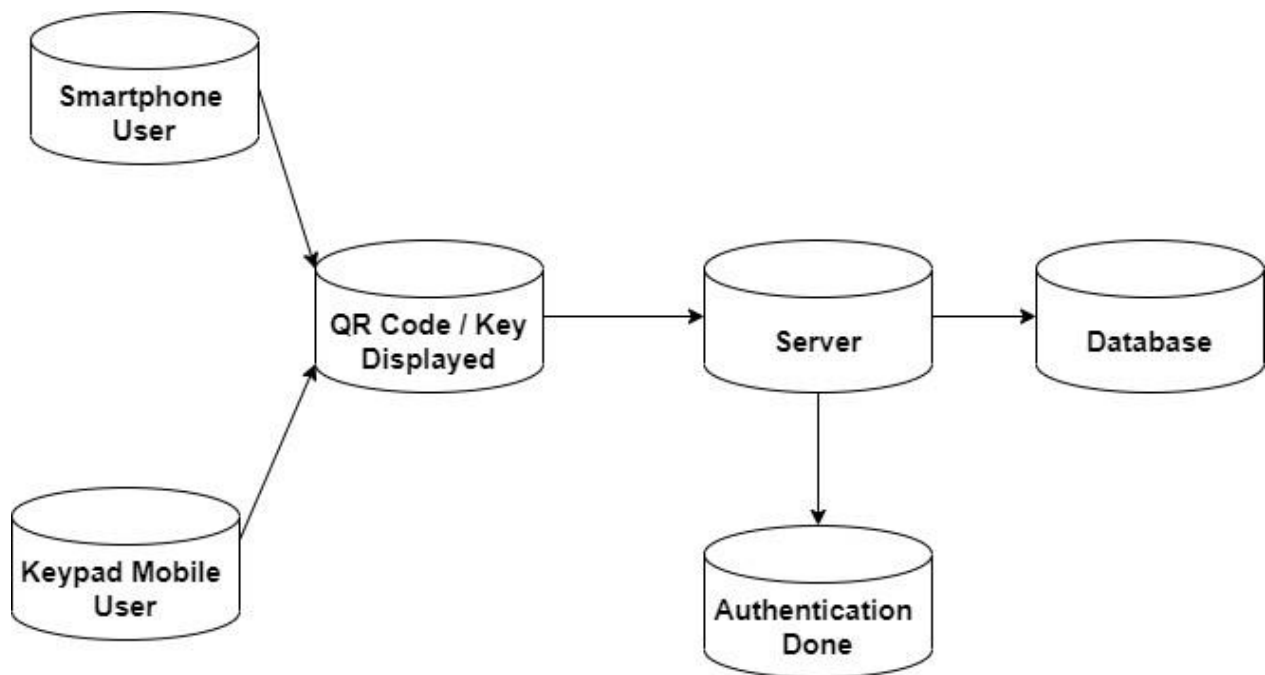


Fig 4.3.4 Collaboration Diagram

CHAPTER 5

SYSTEM CONFIGURATION

5.1 HARDWARE REQUIREMENTS

- Processor : Dual-core processor 2.6 – 3.0 GHZ
- RAM : 2 GB (Min.)
- Hard disk : 100 GB (Max.)
- Monitor : 15 inch color-monitor

5.2 SOFTWARE REQUIREMENTS

- Operating system: Windows 7 or later, Linux, Mac
- Front End: Angular JS, HTML, SCSS
- Back End: Node JS, Ganache (Blockchain), Truffle Framework,
- Metamask Browser extension (Connect Browser to Blockchain)
- IDE: VS Code
- Application: Web APP

CHAPTER 6 MODULES

MODULE LIST

- Web App Framework
- Blockchain Layer
- Smart Contract

MODULE DESCRIPTIONS

6.1 WEB APP FRAMEWORK

Contents

- GUI – Admin, Doctors and Patients
- Metamask Integration

Usage

- Admin, Doctor and Patient registration can be done using their respective Metamask wallets.
- Appointments can be requested and accepted by a patient and doctor respectively.
- Uploading reports and other medical documents

6.2 BLOCKCHAIN LAYER

Contents

- Blockchain Assets
- Governance Rules
- Network

Usage

- Transactions are treated as *assets* by the Ethereum network.
- Governance of the network is maintained using the ‘Proof of Work’ (PoW) mechanism.
- IPFS architecture is used to store data as peer-peer nodes.
- Acts as additional layer of Security

63 SMART CONTRACT

Contents

- Patient Records
- Roles

Usage

- Patient Record implements the functionality of the proposed framework.
- Roles is a smart contract library by Open Zeppelin used to create our own methods

CHAPTER 7 SOFTWARE TESTING

7.1 TESTING PROCESS

Software testing is a method of assessing the functionality of a software program. There are many different types of software testing but the two main categories are dynamic testing and static testing. Dynamic testing is an assessment that is conducted while the program is executed; static testing, on the other hand, is an examination of the program's code and associated documentation. Dynamic and static methods are often used together.

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and works towards the integration of the entire computer-based system. Nothing is complete without testing, as it is a vital success of the system.

- Testing Objectives:

There are several rules that can serve as testing objectives, they are

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case has a high probability of finding an undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also, testing demonstrates that software functions appear to be working according to the specification, that performance requirements appear

to have been met.

There are three ways to test a program

1. For Correctness
2. For Implementation efficiency
3. For Computational Complexity.

Tests for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

Tests used for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which reexamines the implementation phase of algorithm development. Tests for computational complexity amount to an experimental analysis of the complexity of an algorithm or an experimental comparison of two or more algorithms, which solve the same problem.

The data is entered in all forms separately and whenever an error occurred, it is corrected immediately. A quality team deputed by the management verified all the necessary documents and tested the Software while entering the data at all levels. The development process involves various types of testing. Each test type addresses a specific testing requirement. The most common types of testing involved in the development process are:

- Unit Test.
- System Test
- Integration Test
- Functional Test

7.2 UNIT TESTING

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called a unit test. The Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and

contains clearly defined inputs and expected results. Unit testing producing tests for the behavior of components (nodes and vertices) of a product to ensure their correct behavior prior to system integration.

7.3 FUNCTIONAL TESTING:

The Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification, and establishing confidence that a program does what it is supposed to do.

7.4 INTEGRATION TESTING:

In integration testing modules are combined and tested as a group. Modules are typically code modules, individual applications, source and destination applications on a network, etc. Integration Testing follows unit testing and precedes system testing. Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is released.

7.5 WHITE BOX TESTING:

Testing is based on an analysis of the internal workings and structure of a piece of software. This testing can be done using the percentage value of load and energy. The tester should know what exactly is done in the internal program. It includes techniques such as Branch Testing and Path Testing. White box testing is also known as Structural Testing and Glass Box Testing.

7.6 BLACK BOX TESTING:

In black-box testing, the tester has without knowledge of the internal workings of the item being tested. Tests are usually functional. This testing can be done by the user who has no knowledge of how the shortest path is found.

CHAPTER 8 CONCLUSION

AND FUTURE WORK

8.1 CONCLUSION

Our results show that the executions of smart contract functions are inexpensive, implying that blockchain technology could be cost-effective for EHR sharing while ensuring the privacy, confidentiality, and integrity of these shared EHRs. Several authors are widely proposed blockchain based systems in the healthcare field to secure the sharing of health data. Contrary to Zhao et al. [16], which saves all blockchain records, this article uses IPFS storage technology to relieve the strain on blockchain storage and meet real-world deployment requirements.

In this project we discussed how blockchain technology can 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. 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.

8.2 FUTURE WORK

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. We would also need to implement medical insurance module for patients and also an SMS integration to facilitate quicker response from doctors.

APPENDIX 1

SOURCE CODE

FRONT-END (HTML):

```

<app-navigation></app-navigation>
<div class="home container-fluid">
  <div class="overlay">
    <section class="her-wrap js-fullheight " data-section="home" data-stellar-background-
ratio="0.5">
      <div class="container">
        <div class="row no-gutters slider-text js-fullheight align-items-center justify-
content-start"
          data-scrollax-parent="true">
          <div class="content col-md-6">
            <div class="subh">
              <span class="subheading">Welcome to EHR 3.0</span>
              <h1 class="mb-4">We are here <br />for your Care</h1>
              <figure class="text-start">
                <blockquote class="blockquote">
                  <p>Your time is limited, so don't waste it living someone else's life. Don't
be trapped by
                  dogma - which is living with the results of other people's thinking.</p>
                </blockquote>
                <figcaption class="blockquote-footer">
                  Steve Jobs.
                </figcaption>
              </figure>
              <p class="mb-4"> </p>
              <p><a [routerLink]="['appointment']" class="btn btn-primary py-3 px-4 rounded-
pill">Make an appointment</a></p>
            </div>
          </div>
        </div>
      </div>
    </section>
  </div>
</div>

```

BACKEND:

```

from dataclasses import dataclass
from django.shortcuts import render
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from rest_framework.decorators import api_view

```

```

from api.models import Appointment, Doctor, Patient
from .serializers import AppointmentSerializer, DoctorSerializer, PatientSerializer
# Create your views here.

```

```

class DoctorView(APIView):
    def post(self, request):
        serializer = DoctorSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success", "data": serializer.data},
status=status.HTTP_200_OK)
        else:
            return Response({"status": "error", "data": serializer.errors},
status=status.HTTP_400_BAD_REQUEST)

    def get(self, request, id=None):
        if id:
            doctor = Doctor.objects.filter(docID=id)
            serializer = DoctorSerializer(doctor, many = True)
            return Response({"status": "success", "data": serializer.data},
status=status.HTTP_200_OK)
        doctors = Doctor.objects.all()
        serializer = DoctorSerializer(doctors, many=True)
        return Response({"status": "success", "data": serializer.data}, status.HTTP_200_OK)

    def delete(self, request, id=None):
        doctor = Doctor.objects.filter(docID=id)
        doctor.delete()

        return Response({"status": "success", "data": True})

class PatientView(APIView):
    def post(self, request):
        serializer = PatientSerializer(data=request.data)

        if serializer.is_valid():
            serializer.save()
            return Response({"status": "success", "data": serializer.data},
status=status.HTTP_200_OK)
        else:
            return Response({"status": "error", "data": serializer.errors},
status=status.HTTP_400_BAD_REQUEST)

class AppointmentView(APIView):
    def post(self, request):

        doctor = Doctor.objects.get(docID=request.data.get('docID'))
        patient = Patient.objects.get(patID=request.data.get('patID'))

        request.data._mutable = True

```



```

request.data['docName'] = doctor.fName + ' ' + doctor.lName
request.data['patName'] = patient.patName
request.data._mutable = False

serializer = AppointmentSerializer(data=request.data)
if serializer.is_valid():
    serializer.save()
    return Response({"status": "success", "data": serializer.data},
status=status.HTTP_200_OK)
else:
    return Response({"status": "error", "data": serializer.errors},
status=status.HTTP_400_BAD_REQUEST)

def get(self, request):
    appointments = Appointment.objects.all()
    serializer = AppointmentSerializer(appointments, many=True)
    return Response({"status": "success", "data": serializer.data}, status.HTTP_200_OK)

def put(self, request, id):
    appointment = Appointment.objects.get(id=id)
    appointment.status = True
    appointment.save()
    return Response({"status": "success", "data": appointment.status}, status.HTTP_200_OK)

@api_view(['GET'])
def getAppointmentDoc(self, id):

    appointment = Appointment.objects.filter(docID=id)
    serializer = AppointmentSerializer(appointment, many=True)
    return Response({"status": "success", "data": serializer.data}, status.HTTP_200_OK)

@api_view(['GET'])
def getAppointmentPat(self, id):

    appointment = Appointment.objects.filter(patID=id)
    serializer = AppointmentSerializer(appointment, many=True)
    return Response({"status": "success", "data": serializer.data}, status.HTTP_200_OK)

@api_view(['GET'])
def getCount(self):
    doctorCount = Doctor.objects.all().count()
    patientCount = Patient.objects.all().count()
    return Response({"status": "success", "docCount": doctorCount, "patCount": patientCount},
status.HTTP_200_OK)

@api_view(['GET'])
def clear(self):
    Doctor.objects.all().delete()
    Appointment.objects.alias().delete()

```

```
Patient.objects.alias().delete()
```

```
return Response({"status": "success"}, status.HTTP_200_OK)
```

SMART CONTRACT:

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity >= 0.4.21;
```

```
import './Roles.sol';
```

```
contract Contract{
```

```
    using Roles for Roles.Role;
```

```
    Roles.Role private admin;
```

```
    Roles.Role private doctor;
```

```
    Roles.Role private patient;
```

```
    struct Doctor{
        string drHash;
    }
```

```
    struct Patient{
        string patHash;
    }
```

```
    struct MedRec{
        string RecordHash;
    }
```

```
    mapping(address => Doctor) Doctors;
    mapping(address => Patient) Patients;
    mapping(address => MedRec) Records;
```

```
    address[] public Dr_ids;
    address[] public Patient_ids;
    string[] public RecordHashes;
```

```
    address accountId;
    address admin_id;
    address get_patient_id;
    address get_dr_id;
```

```
    constructor() {
        admin_id = msg.sender;
        admin.add(admin_id);
    }
```

```
//get Admin
```



```

function getAdmin() public view returns(address){
    return admin_id;
}

//Add Doctor

function addDoctor(address _newdr) public{
    require(admin.has(msg.sender), 'Only For Admin');
    doctor.add(_newdr);
}

function delDoctor(address docID) public {
    require(admin.has(msg.sender), 'Only For Admin');
    doctor.remove(docID);
}

// check is Doctor

function isDr(address id) public view returns(string memory){
    require(doctor.has(id), "Only for Doctors");
    return "1";
}

// Check is Patient

function isPat(address id) public view returns(string memory){
    require(patient.has(id), "Only for Doctors");
    return "1";
}

// Add patient

function addPatient(address _newpatient) external onlyAdmin() {
    patient.add(_newpatient);
}

// Get Patient Information => retrun pateint IPFS hash

function getPatInfo(address id)public view returns(string memory){
    return (Patients[id].patHash);
}

// Add patient Information to BlockChain

function addPatInfo(address pat_id, string memory _patInfoHash) public {
    Patient storage patInfo = Patients[pat_id];
    patInfo.patHash = _patInfoHash;
    Patient_ids.push(pat_id);

    patient.add(pat_id);
}

```

```

// Add Medical record to block chain

function addMedRecord(string memory _recHash, address _pat_id) public{
    require(doctor.has(msg.sender) == true, 'Only Doctor Can Do That');

    MedRec storage record = Records[_pat_id];
    record.RecordHash = _recHash;
    RecordHashes.push(_recHash);
}

// View Medical record return IPFS hash of record

function viewMedRec(address id)public view returns(string memory){
    return (Records[id].RecordHash);
}

/*
    Modifiers
*/

modifier onlyAdmin(){
    require(admin.has(msg.sender) == true, 'Only Admin Can Do That');
    _;
}
modifier onlyDoctor(){
    require(doctor.has(msg.sender) == true, 'Only Doctor Can Do That');
    _;
}
modifier onlyPatient(){
    require(patient.has(msg.sender) == true, 'Only Admin Can Do That');
    _;
}
}

```

ALGORITHM:

◆ 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 ( msg.sender == doctor || patient) then
        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 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:

```

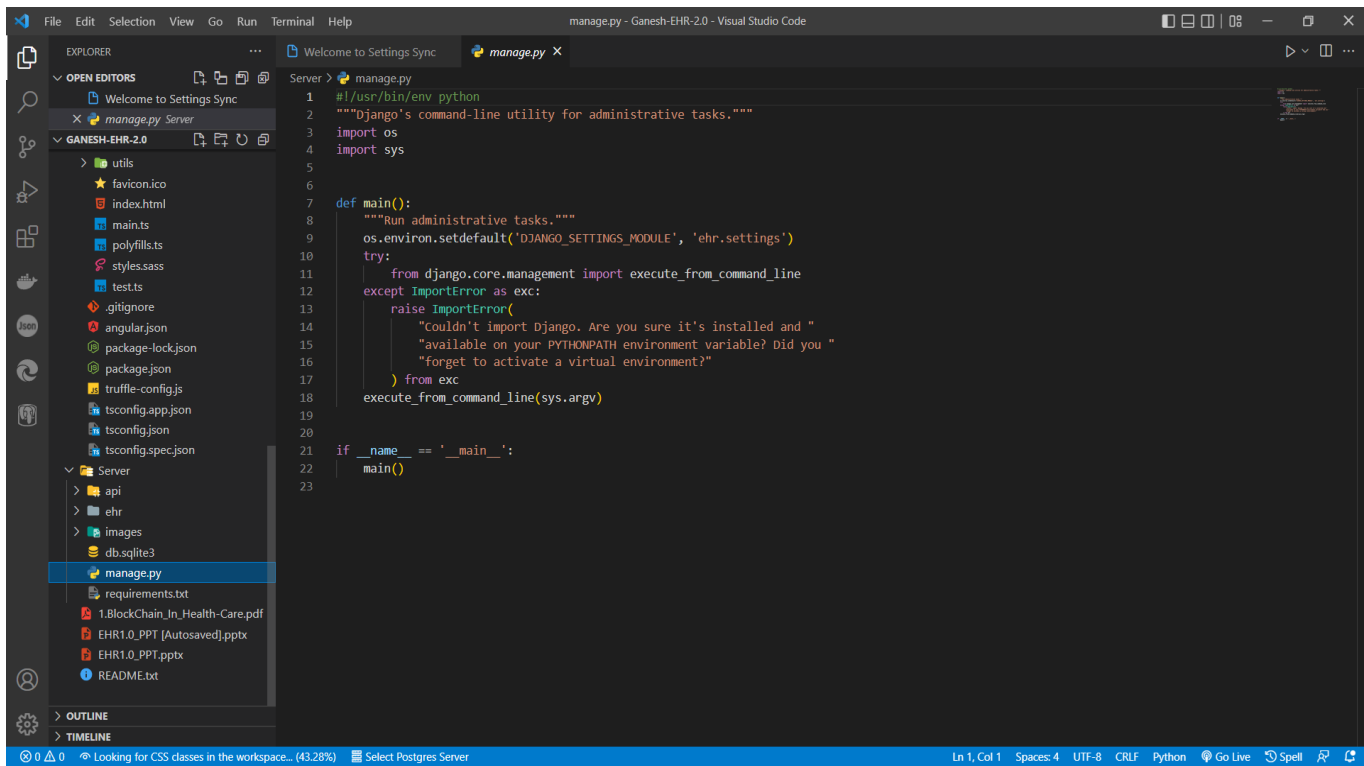
function Delete Patient Record ( patient id )
    if (msg.sender == doctor ) then
        if ( id == patient id ) then
            delete particular patient's record
            return success
        else return fail
    end if

```

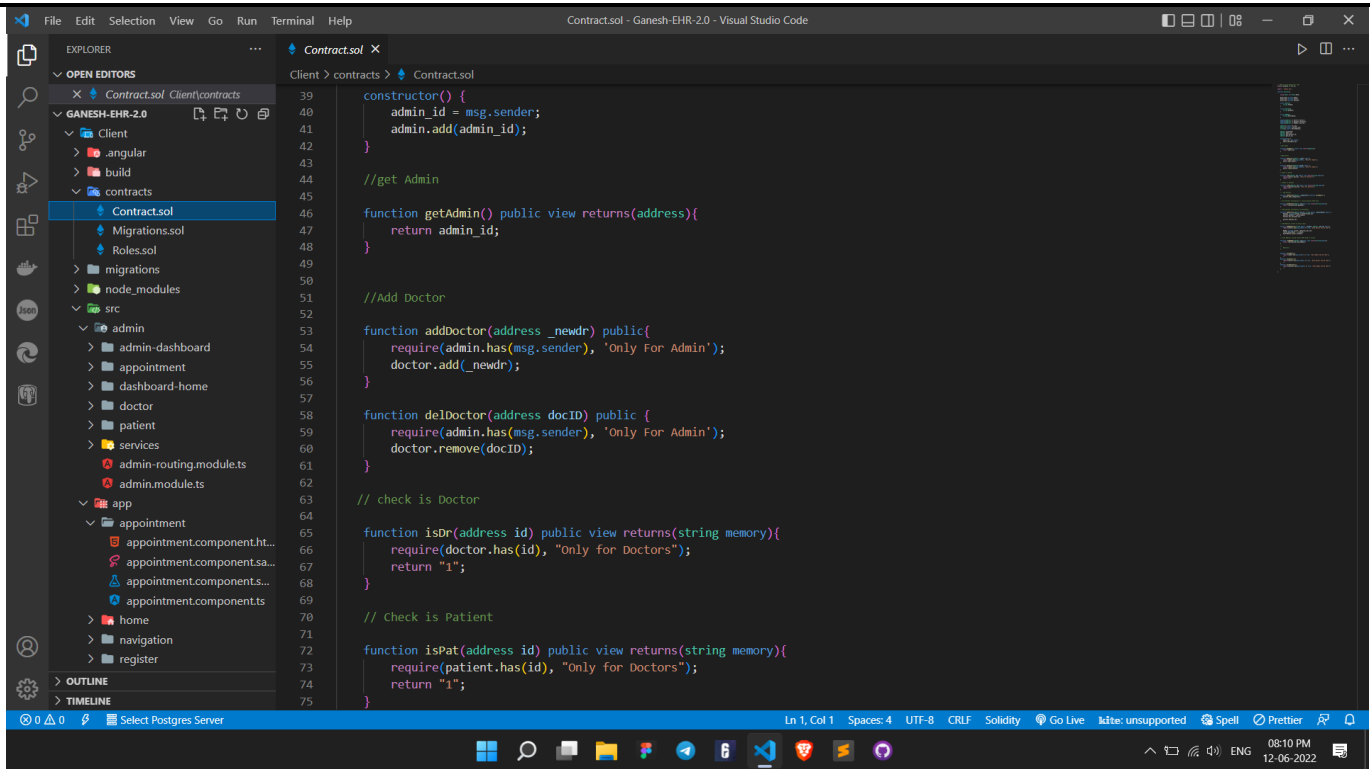
```

end if
else Abort session
end if
end function

```



Server using Django



```
39 constructor() {
40     admin_id = msg.sender;
41     admin.add(admin_id);
42 }
43
44 //get Admin
45
46 function getAdmin() public view returns(address){
47     return admin_id;
48 }
49
50 //Add Doctor
51
52 function addDoctor(address _newdr) public{
53     require(admin.has(msg.sender), 'Only For Admin');
54     doctor.add(_newdr);
55 }
56
57 function delDoctor(address docID) public {
58     require(admin.has(msg.sender), 'Only For Admin');
59     doctor.remove(docID);
60 }
61
62 // check is Doctor
63
64 function isDr(address id) public view returns(string memory){
65     require(doctor.has(id), "Only for Doctors");
66     return "1";
67 }
68
69 // Check is Patient
70
71 function isPat(address id) public view returns(string memory){
72     require(patient.has(id), "Only for Doctors");
73     return "1";
74 }
75
```

Contract.sol




```

// Add patient
function addPatient(address _newpatient) external onlyAdmin() {
    patient.add(_newpatient);
}

// Get Patient Information => retrun pateint IPFS hash
function getPatInfo(address id) public view returns(string memory){
    return (Patients[id].patHash);
}

// Add patient Information to BlockChain
function addPatInfo(address pat_id, string memory _patInfoHash) public {
    Patient storage patInfo = Patients[pat_id];
    patInfo.patHash = _patInfoHash;
    Patient_ids.push(pat_id);
    patient.add(pat_id);
}

// Add Medical record to block chain
function addMedRecord(string memory _rechash, address _pat_id) public{
    require(doctor.has(msg.sender) == true, 'Only Doctor Can Do That');

    MedRec storage record = Records[_pat_id];
    record.RecordHash = _rechash;
    RecordHashes.push(_rechash);
}

// View Medical record return IPFS hash of record
function viewMedRec(address id) public view returns(string memory){
    return (Records[id].RecordHash);
}

```

```

// SPDX-License-Identifier: MIT
pragma solidity >= 0.4.21;
import './Roles.sol';

contract Contract{
    using Roles for Roles.Role;

    Roles.Role private admin;
    Roles.Role private doctor;
    Roles.Role private patient;

    struct Doctor{
        string drHash;
    }

    struct Patient{
        string patHash;
    }

    struct MedRec{
        string RecordHash;
    }

    mapping(address => Doctor) Doctors;
    mapping(address => Patient) Patients;
    mapping(address => MedRec) Records;

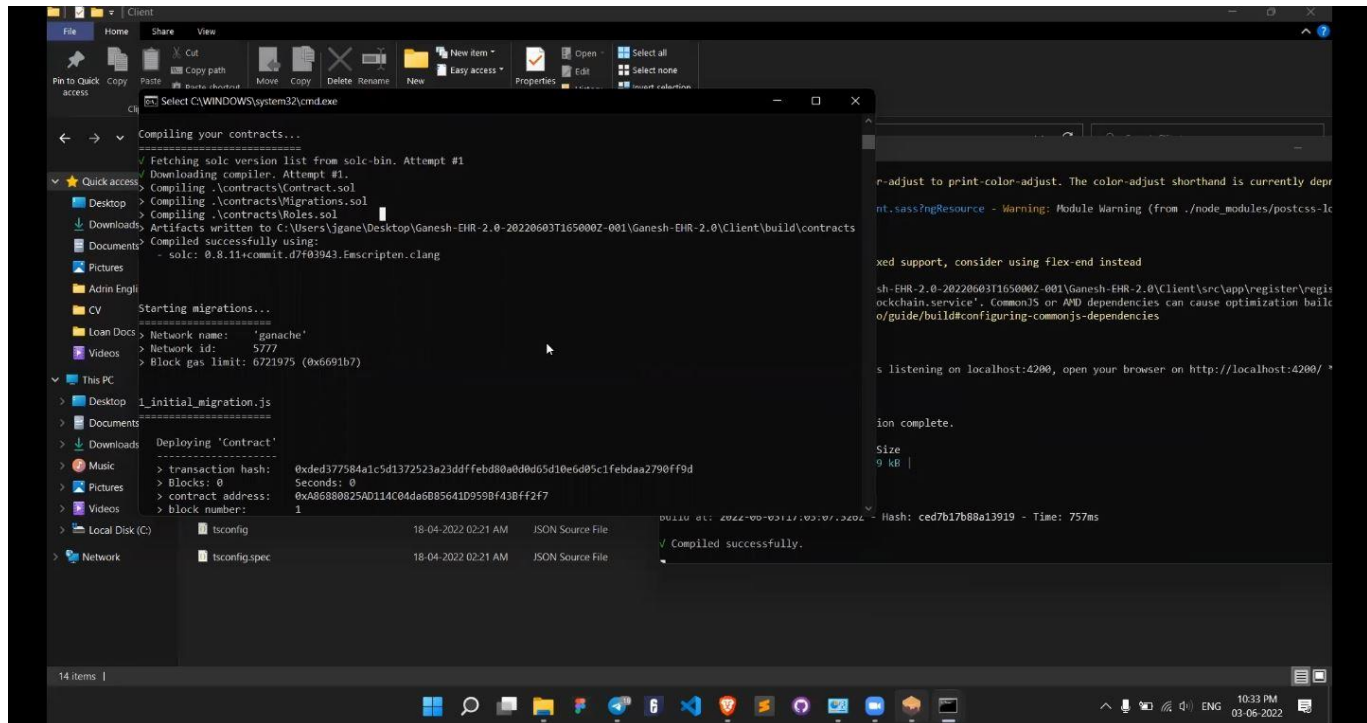
    address[] public Dr_ids;
    address[] public Patient_ids;
    string[] public RecordHashes;

    address accountId;
    address admin_id;
    address get_patient_id;
    address get_dr_id;
}

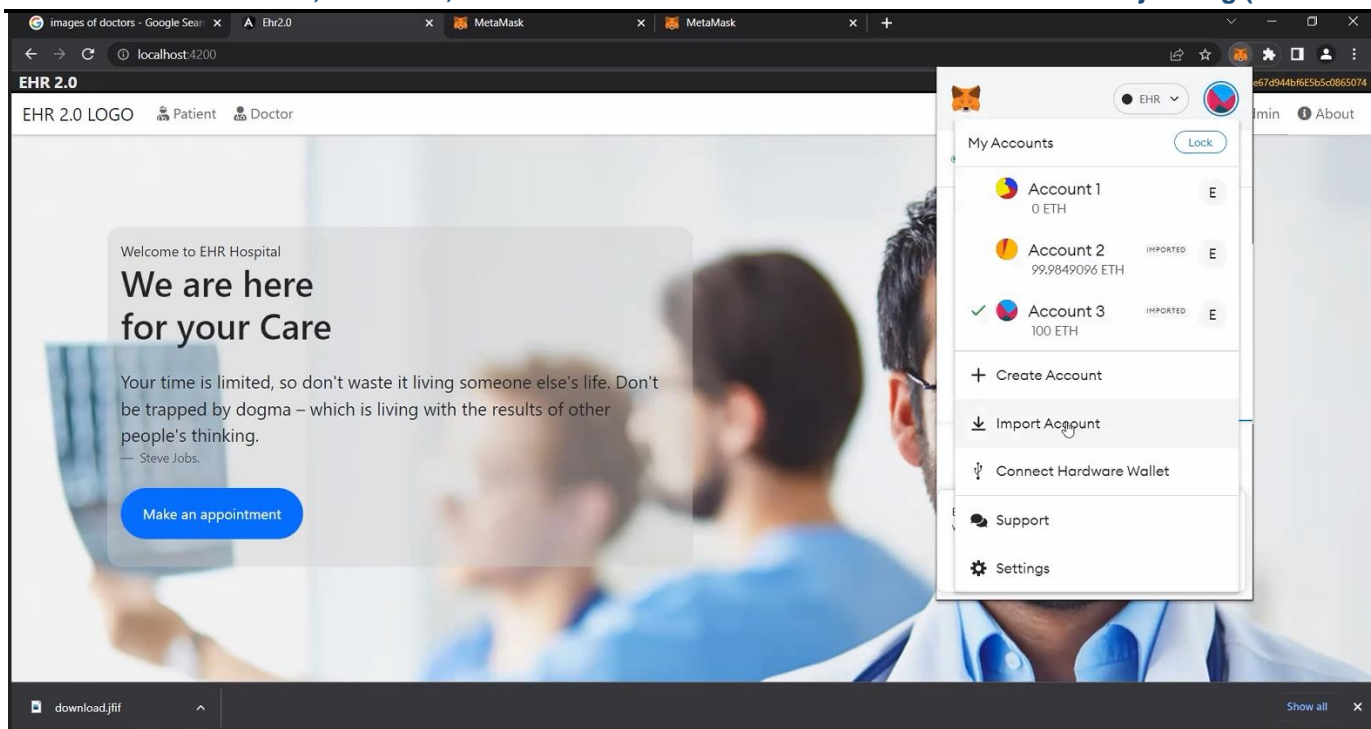
```

APPENDIX 2

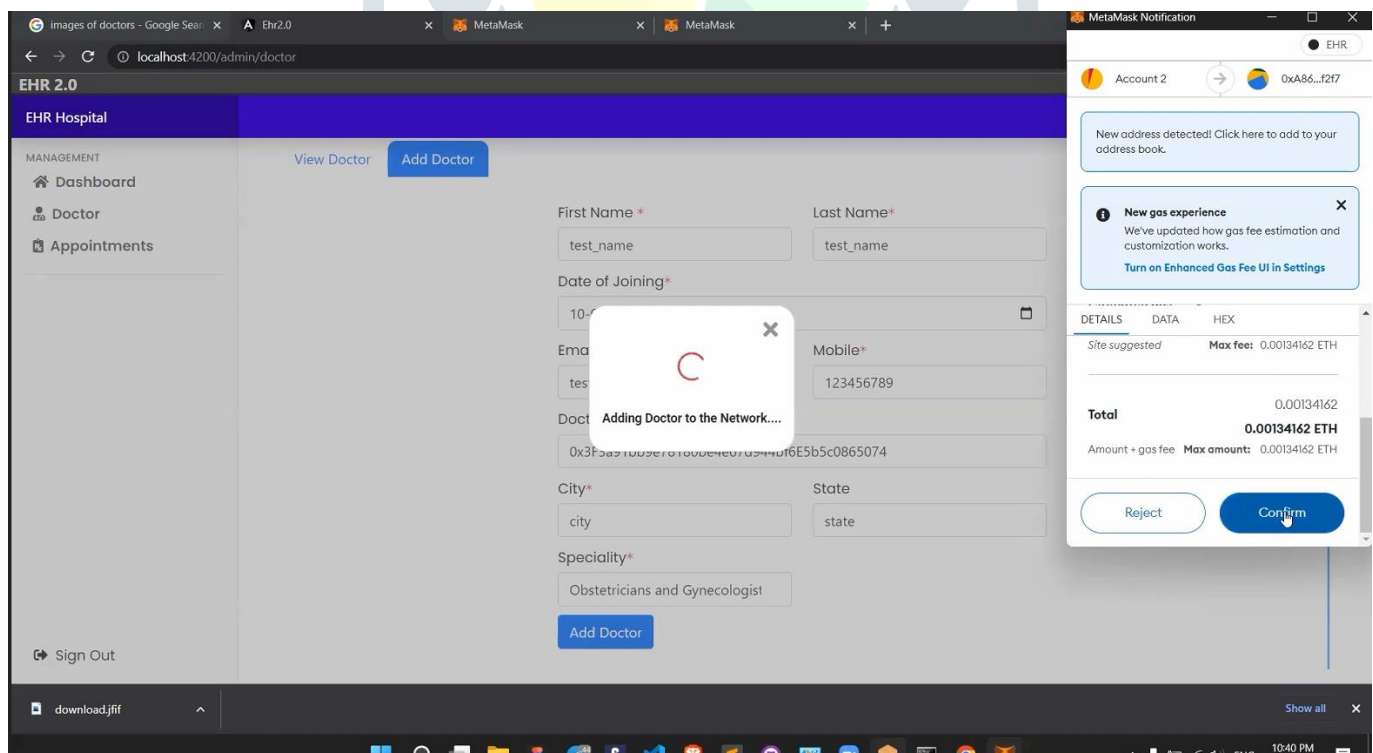
SCREENSHOTS



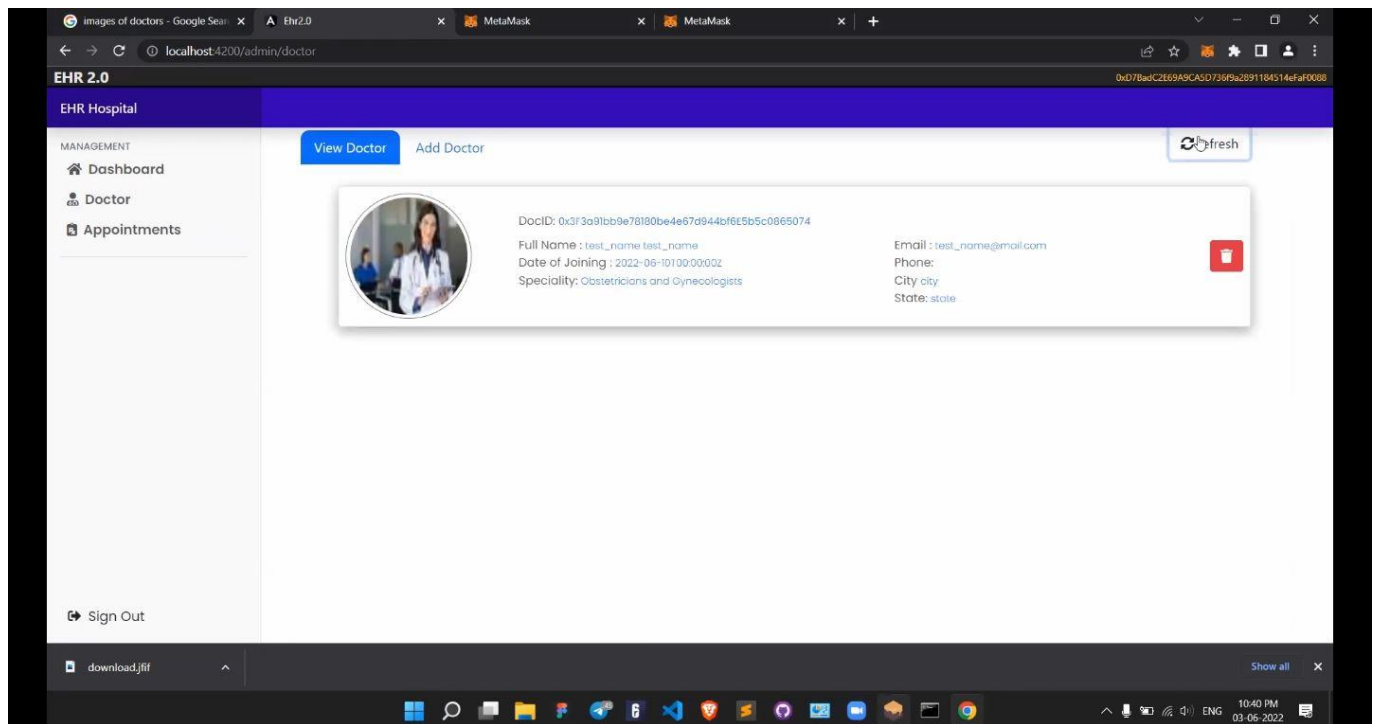
Running the Client and Server-side locally



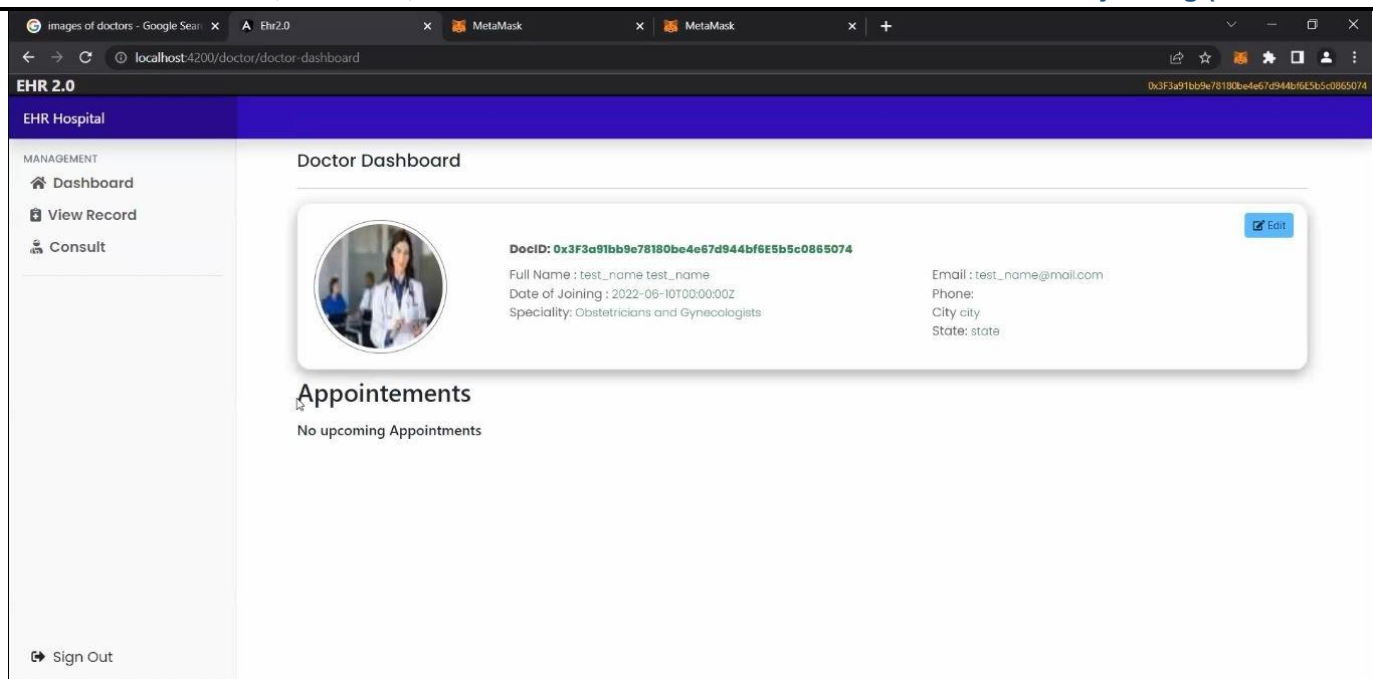
Connecting Metamask with the WebApp



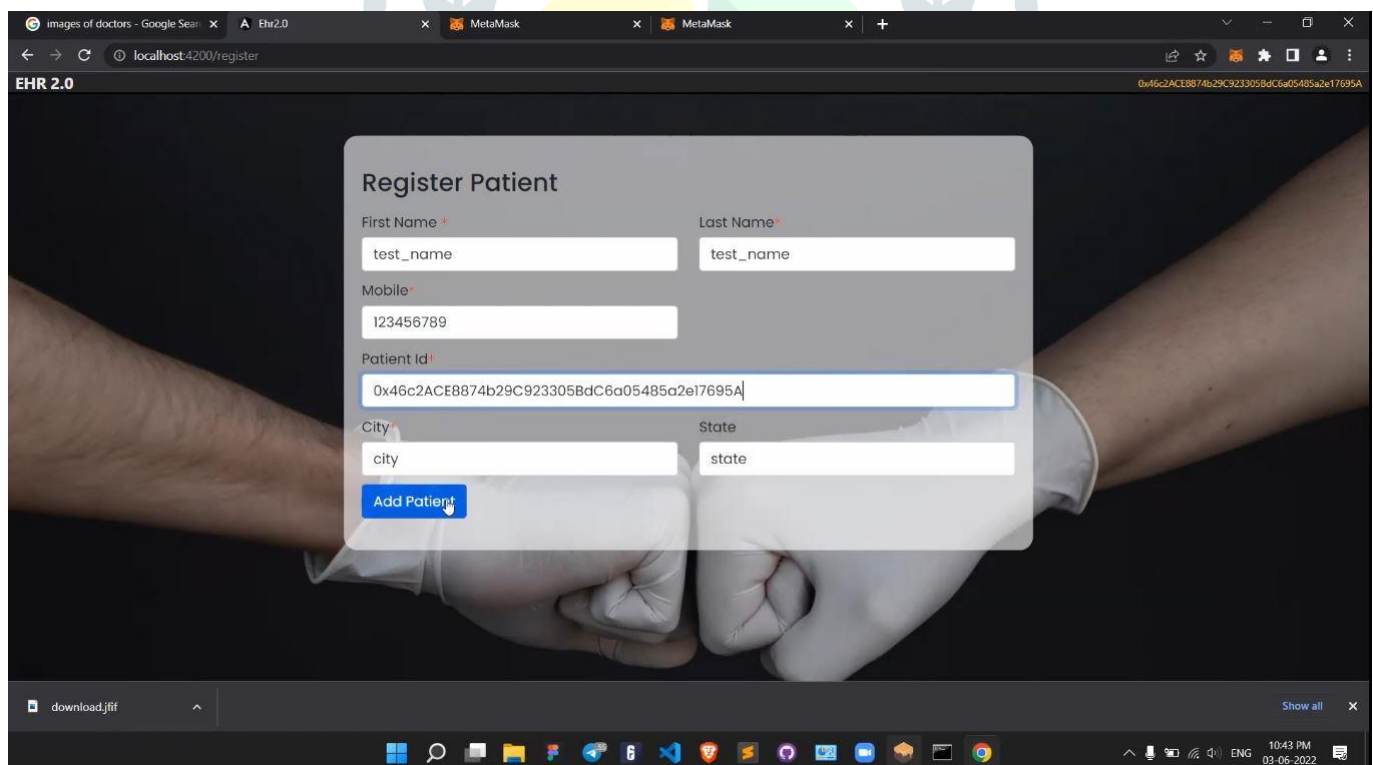
Admin adds a doctor to the IPFS network



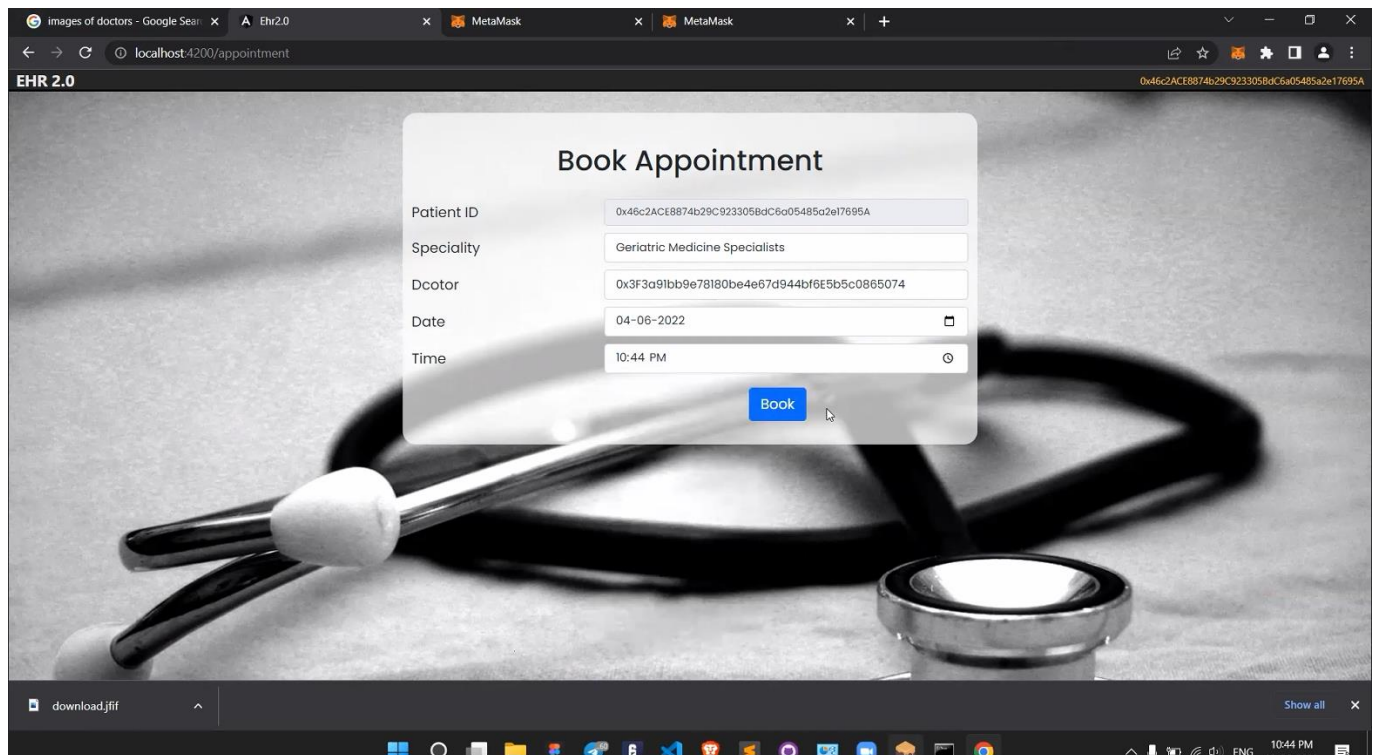
Admin Panel



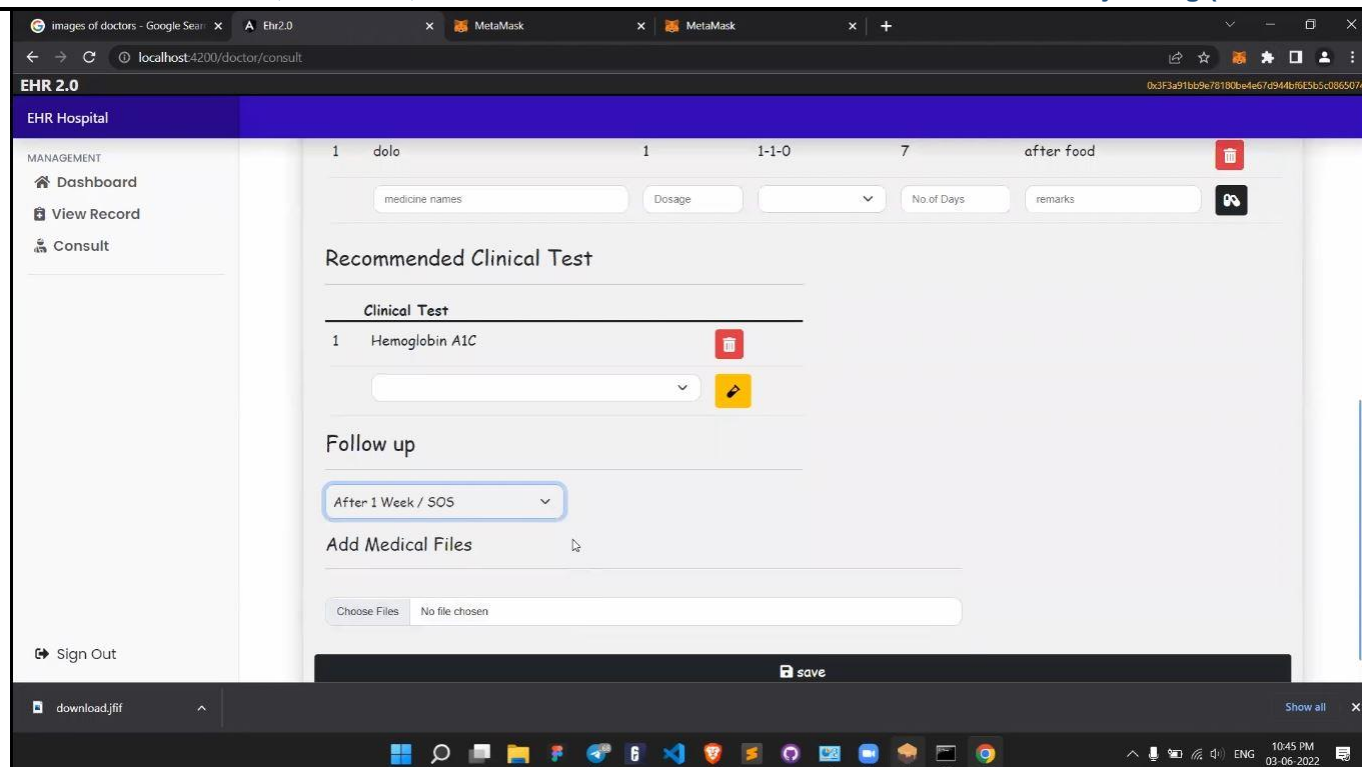
Doctor's Panel



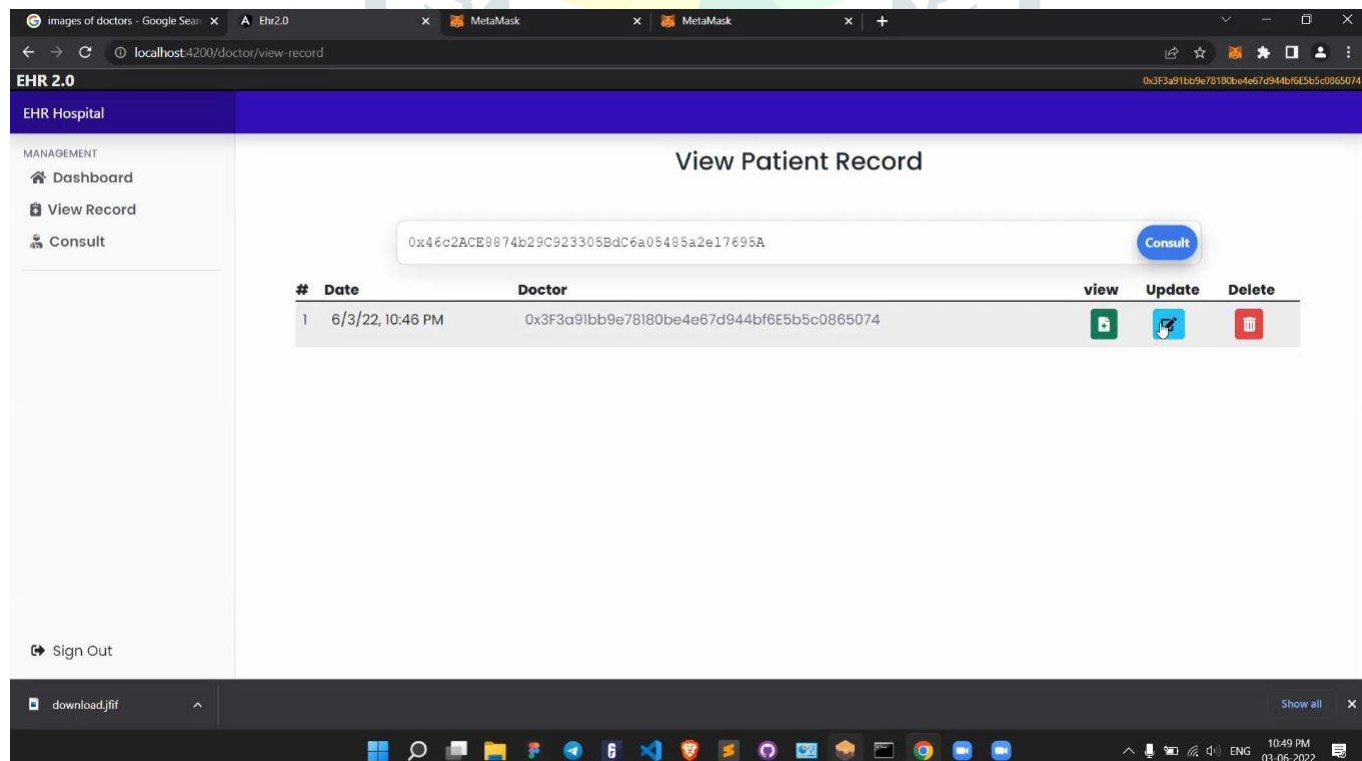
Patient Registration using his/her Public Key



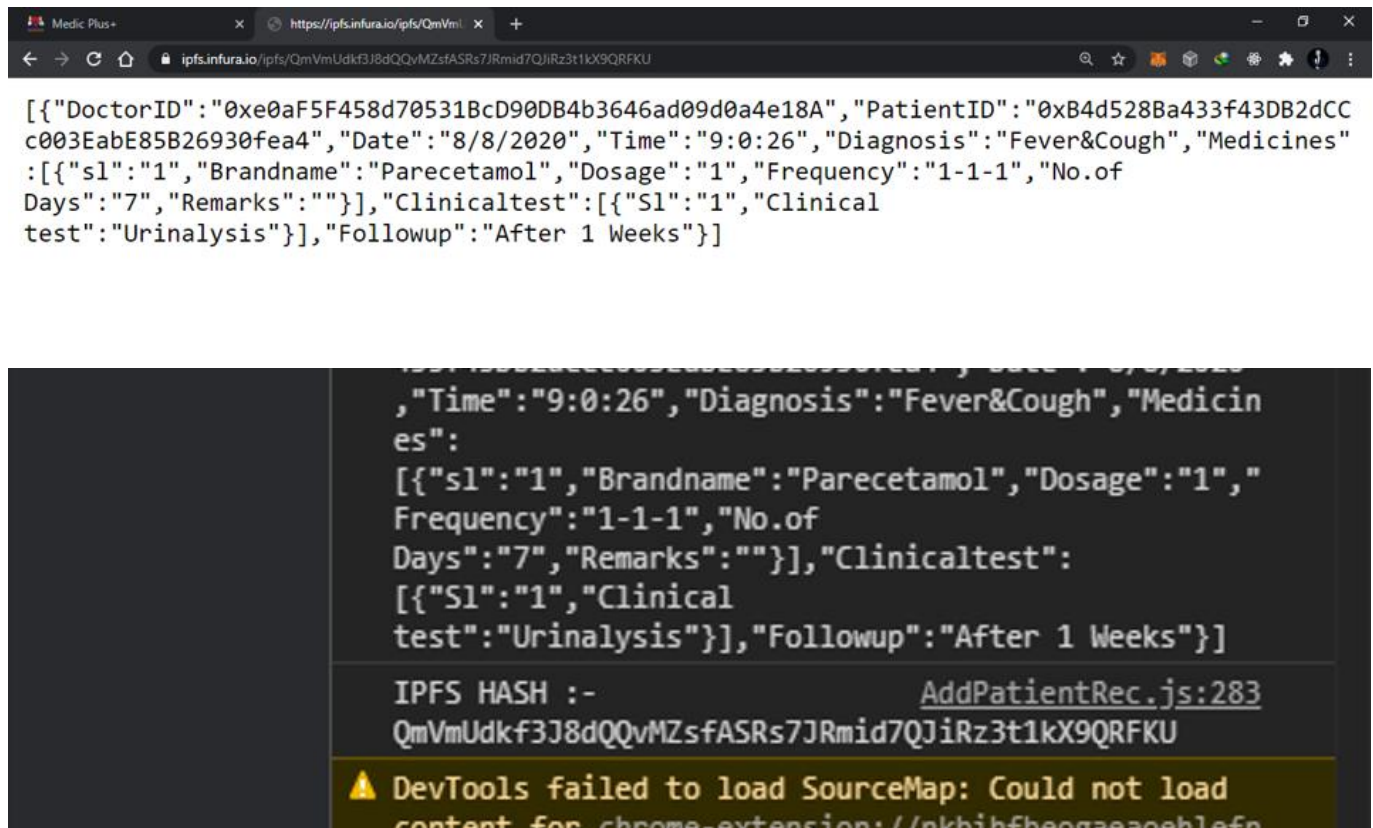
Appointment Booking



Doctor's Consultation



Patient's Record from Doctor's Panel



IPFS Storage as JSON Files

REFERENCES

- [1] Ibrahim Abunadi and Ramasamy Lakshmana Kumar (2021), “BSF-HER: Blockchain Security Framework for Electronic Health Records of Patients”, MDPI
- [2] Baocheng Wang and Zetao Li (2021), “Healthchain: A Privacy Protection System for Medical Data Based on Blockchain”, MDPI.
- [3] Zhao, Yanqi, Yong Yu, Yannan Li, Gang Han, and Xiaojiang Du (2019), “A Machine learning-based privacy-preserving fair data trading in big data market”, IEEE.
- [4] Shen, Wenting, Jing Qin, Jia Yu, Rong Hao, and Jiankun Hu. (2018), “Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage”, IEEE.
- [5] Mathis Steichen; Beltran Fiz; Robert Norvill; Wazen Shbair; Radu State (2019), “Blockchain-Based, Decentralised Access Control for IPFS”, IEEE.
- [6] Kaidong Wu, Yun Ma, Gang Huang, Xuanzhe Liu (2019), “A First Look at Blockchain-based Decentralized Applications”, IEEE

- [7] Auqib Hamid Lone, Roohie Naaz (2020), “Demystifying Cryptography behind Blockchains and a Vision for Post-Quantum Blockchains”, IEEE
- [8] Chen, Jinchuan, and Yunzhi Xue (2017), “Bootstrapping a blockchain based ecosystem for big data exchange”, IEEE
- [9] Divya Anand, Hani Moaiteq Aljahdali, Santos Gracia Villar (2022), “Blockchain Interoperability: Towards a Sustainable Payment System”, MDPI
- [10] Ms. Nilima V. Pardakhe, Dr. V. M. Deshmukh (2021), “Secure electronic health care (EHC) system using Blockchain technique”, IJCRT
- [11] SHANGPING WANG , DAN ZHANG , AND YALING ZHANG.” Blockchain-Based Personal Health Records Sharing Scheme With Data Integrity Verifiable.” IEEE Access, vol. 7, pp. 102888–102901, 2019
- [12] RUI GUO, HUIXIAN SHI, QINGLAN ZHAO, AND DONG ZHENG, “Secure AttributeBased Signature Scheme With Multiple Authorities for Blockchain in Electronic Health Records Systems”, 2169-3536, 2018 IEEE.
- [13] Jingwei Liu_, Xiaolu Li_, Lin Yey, Hongli Zhangy, Xiaojiang Duz, and Mohsen Guizanix, “BPDS: A Blockchain based Privacy-Preserving Data Sharing for Electronic Medical Records”, 978-1-5386-4727-1/18/\$31.00 ©2018 IEEE.
- [14] HaoWang, Yujiao Song, “Secure Cloud-Based EHR System Using Attribute-Based Cryptosystem and Blockchain”, <https://doi.org/10.1007/s10916-018-0994-6>, 12 June 2018.
- [15] Hina Abrar, Syed Jawad Hussain, Junaid Chaudhry, Kashif Saleem, Mehmet A. Orgun, Jalal Al-Muhtadi, Craig Valli, “Risk Analysis of Cloud Sourcing in Healthcare and Public Health Industry”, IEEE Access 2018.
- [16] Zhao Y, Cui M, Zheng L, Zhang R, Meng L, Gao D, et al. Research on electronic medical record access control based on blockchain. International Journal of Distributed Sensor Networks 2019;15(11):1550147719889330.
- [17] XIAODONG YANG, TING LI, XIZHEN PEI ,LONG WEN, AND CAIFEN WANG (2020), “Medical Data Sharing Scheme Based on Attribute Cryptosystem and Blockchain Technology.” IEEE Access

- [18] Tareq Ahram, Arman Sargolzaei, Saman Sargolzaei, Jeff Daniels, and Ben Amaba, “Blockchain Technology Innovations”, 978-1-5090-1114- 8/17/\$31.00 ©2017 IEEE.
- [19] J.-H. Lee, “BIDaaS: Blockchain Based ID As a Service,” IEEE Access, vol. 6, pp. 2274–2278, 2018.
- [20] Pinyaphat Tasatanattakool, Chian Techapanupreeda (2018), “Blockchain: Challenges and Applications”, IEEE

