

BLOCKCHAIN BASED ELECTRONIC HEALTH RECORD MANAGEMENT IMPLEMENTATION USING ETHEREUM TESTNET PLATFORM

A PROJECT REPORT

Submitted by

HARSHIT AGRAWAL [Reg No: RA1511008010272]

Under the Guidance of

Ms. C. Fancy

(Assistant Professor, Department of Information Technology)

*In partial fulfillment of the Requirements for the Degree
of*

BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY



**DEPARTMENT OF INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603203**

MAY 2019

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report titled “**BLOCKCHAIN BASED ELECTRONIC HEALTH RECORD MANAGEMENT IMPLEMENTATION USING ETHEREUM TESTNET PLATFORM**” is the bonafide work of “**HARSHIT AGRAWAL [Reg No: RA1511008010272]**”, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Ms. C. FANCY
GUIDE
Assistant Professor
Dept. of Information Technology

Dr. G. VADIVU
HEAD OF THE DEPARTMENT
Dept. of Information Technology

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

Businesses growing exponentially require revolutionary changes in all the domains of their businesses as the time passes and the technology progresses. Blockchain technology has the power to revolutionize every field including healthcare and how the data is stored. Blockchain lets us create decentralized applications that is not in power of any central authority.

In current systems, patients' data and information that are critical are scattered across different systems and institutions. Due to this, non trivial data is not handily available or accessible when needed. Such problems related to patient healthcare data can be handled effectively using blockchain solutions. This ever increasing data leads to problems in managing patient information within clinics and hospitals.

Our project aims to create a hash for individual patient health information blocks. It would also permit patients to keep their identity secret and reveal the necessary information to third parties at the same time. The access permissions and time limit for data sharing can also be controlled by the them. A platform called "Ethereum" will be used as the blockchain technology to create an application that will store patients' data securely in the blockchain. A contract based language called "Solidity" will be used to bind each patients' data with their unique addresses to the blockchain and also to provide functionalities to the application. A browser based user-interface will be developed to be used by the users that will connect with the blockchain internally.

Blockchain is finally making its way in the healthcare industry due to its increased popularity and is being accepted positively in the healthcare domain. The outcome of this project will be a patient-centric decentralized application that will help patients manage their medical records in a more efficient way.

ACKNOWLEDGEMENT

The success and the final outcome of this project required guidance and assistance from different sources and we feel extremely fortunate to have got this all along the completion of our project. Whatever we have done is largely due to such guidance and assistance and we would not forget to thank them. We express our sincere thanks to the Head of the Department, Department of Information Technology, Dr. G.Vadivu (PhD), for all the help and infrastructure provided to us to complete this project successfully and her valuable guidance. We owe our profound gratitude to our project guide Ms. C. Fancy (Assistant Professor), who took keen interest in our project and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system. We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the Teaching staff of the Department of Information Technology which helped us in successfully completing our major project work. Also, we would like to extend our sincere regards to all the non-teaching staff of the department of Information Technology for their timely support.

HARSHIT AGRAWAL

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	ABBREVIATIONS	xi
1.	INTRODUCTION	1
1.1	BACKGROUND	1
1.1.1	BLOCKS	1
1.1.2	NODES	2
1.2	TYPES OF BLOCKCHAIN	2
1.2.1	PUBLIC PERMISSIONLESS	2
1.2.2	CONSORTIUM	2
1.2.3	PRIVATE	3
1.3	DAPPS	3
1.4	PURPOSE	4
1.5	PROBLEM STATEMENT	4
1.6	OBJECTIVES	4
1.7	ORGANIZATION OF THE REPORT	5
2	LITERATURE REVIEW	6
3	SYSTEM REQUIREMENTS	10
3.1	FUNCTIONAL REQUIREMENTS	10
3.2	NON-FUNCTIONAL REQUIREMENTS	11
3.3	HARDWARE REQUIREMENTS	11

3.4	SOFTWARE REQUIREMENTS	11
4	PROPOSED METHODOLOGY	12
4.1	ISSUES IN EXISTING METHODOLOGY	12
4.2	NEED FOR NEW METHODOLOGY	12
4.3	METHOD OF IMPLEMENTATION	12
5	IMPLEMENTATION	14
5.1	CHOOSING THE BLOCKCHAIN	14
5.2	ETHEREUM TESTNET	14
5.3	METAMASK	14
5.4	WRITING SMART CONTRACTS	15
5.5	INTERACTION BETWEEN BROWSER AND BLOCKCHAIN	15
5.6	FRONT END	15
5.7	SYSTEM ARCHITECTURE	16
5.8	USE CASE DIAGRAM	17
5.9	MODULES AND ITS IMPLEMENTATION	17
5.9.1	ACCOUNT CREATION	17
5.9.2	ADD NEW MEDICAL DATA	18
5.9.3	DATA SHARING	19
5.9.4	ACCESS MEDICAL HISTORY	20
5.10	TOOLS USED	20
5.11	RESULTS	22
5.11.1	FRONT END OF THE APPLICATION	22
5.11.2	BACK END OF THE APPLICATION	25
6	TESTING	28
6.1	TESTING OBJECTIVES	28
6.2	FRONT END TESTING	28

	6.3 BACK END TESTING	29
7	CONCLUSION	31
8	FUTURE ENHANCEMENT	32
9	REFERENCES	33
	APPENDIX	35
	PAPER PUBLICATION STATUS	44
	PLAGIARISM REPORT	45

LIST OF TABLES

Table No.	Table Name	Page No.
2.1	Interoperability levels	7
2.2	Challenges in interoperability	8
6.1	Testing report	30

LIST OF FIGURES

Figure No.	Figure Name	Page No.
1.1	Chaining between blocks in a blockchain	1
1.2	Model of centralized vs Ethereum app	3
2.1	Papers published with the last few years	6
2.2	A Dapp in use	8
4.1	A Dapp user data flow overview	13
5.1	System architecture diagram	16
5.2	Use Case Diagram	17
5.3	Metamask account management	18
5.4	Sequence Diagram for adding new data	19
5.5	Sequence Diagram for allowing data access	20
5.6	Medical History Records	22
5.7	Individual Medical Records Page	23
5.8	New medical data waiting for approval	23
5.9	Permission Page to allow access	24
5.10	Permission Page to enter time details	24
5.11	Metamask popup for confirming new transaction	25
5.12	Block mining in Ganache	26
5.13	Contract creation call	26
5.14	Contract call to change contract state	26
5.15	Logs stored in the blockchain	27

6.1	Coverage report for jasmine testing	28
6.2	Spec files for jasmine testing	29
6.3	Testing report for Solidity code testing	29

ABBREVIATIONS

Eth	Ethereum
ETH	Ether currency
Dapp	Decentralized Application
JS	JavaScript
TTP	Trusted Third Party
Sol	Solidity
HER	Electronic Health Record
GUI	Graphical User Interface

CHAPTER 1

INTRODUCTION

1.1 THEORETICAL BACKGROUND

A blockchain can be referred to as a distributed ledger. It enables data sharing in a network of peers. It was introduced along with Bitcoin and. It solved an enduring problem of the double-spend problem. Cryptocurrency was the first domain where blockchain technology was applied. However, introducing cryptocurrency is optional in order to use blockchain. Decentralized applications can be built even without the use of cryptocurrency.

A blockchain can be regarded as a chain made of blocks which are linked together using cryptographic hashes and are time-stamped. These blocks are locked in an immutable and secure manner. The chain constantly grows with time as new nodes are added every few seconds.

1.1.1 BLOCKS

A block is a set of transactions. It contains the transactions chronographically. A blockchain is made of a series of blocks. Each block contains the hash of the previous block, and thus they make a series. The first block in a blockchain is also known as the genesis block.

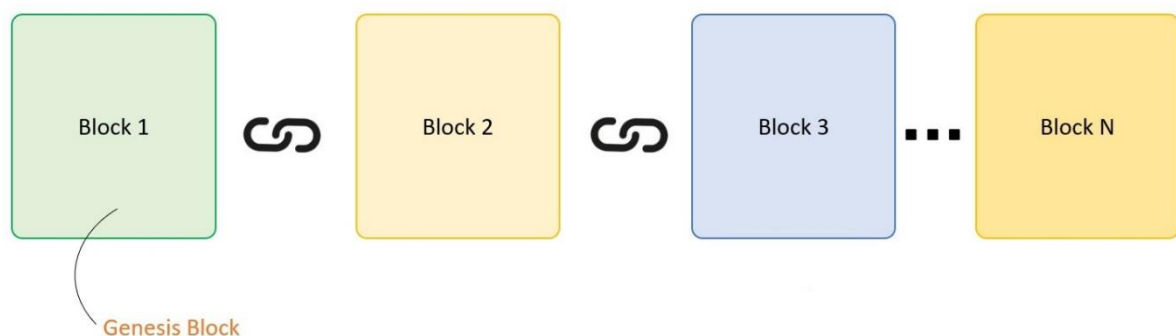


Fig 1.1 Chaining between blocks in a blockchain

1.1.2 NODES

A node is also stated as the shareholder of the blockchain. Every node in the network has two keys, a public key and also a private key. The private key is used for decrypting the messages and it allows a node to read that message. The public key is used for encryption of messages sent to a node. Some nodes also act as the miners of the blockchain ecosystem. The miners are referred to as special nodes that uses an algorithm to validate the transactions submitted by other nodes and make a block that will be a candidate for the next block to be added in the blockchain.

1.2 TYPES OF BLOCKCHAIN

On the availability of data and what actions can be performed by the user, the blockchain can be classified into three types: -

1. Public permissionless
2. Consortium (public permissioned)
3. Private

1.2.1 PUBLIC PERMISSIONLESS

The data in public blockchain is always accessible and visible to everyone. In a public permissionless blockchain, anyone has the right to join the blockchain and become a node without any approval. These types of blockchains are usually given some kind of economic incentive, like in cryptocurrency networks. Examples of such networks are Bitcoin, Ethereum, Litecoin etc.

1.2.2 CONSORTIUM

A consortium blockchain allows only a particular group of nodes that can take part in the consensus process. It falls between the categories of public and private blockchain. It can be used within one or across several industries. It is opened for limited public use and partially centralized.

1.2.3 PRIVATE

A private blockchain allows only particular nodes to be a part of the network either as a node or as a miner. Thus, it is a distributed yet centralized network. They control which nodes can perform transactions, execute smart contracts or act as miners. They are managed by one organization which is the trusted party. It is used for private purposes. Examples of such blockchains are Hyperledger Fabric, Ripple etc.

1.3 DAPPS

A Dapp is an autonomously operated open-source application that is not managed by any central authority. Instead it is decentralized over the web built on the top of blockchain technologies. The data of a Dapp is stored cryptographically in a public and also decentralized blockchain to avoid any single point of failure. It uses cryptographic tokens for monetizing the application.

The left figure shows the model of a centralized application whereas the right figure shows the model of an ethereum application that is not controlled by any kind of central authority.

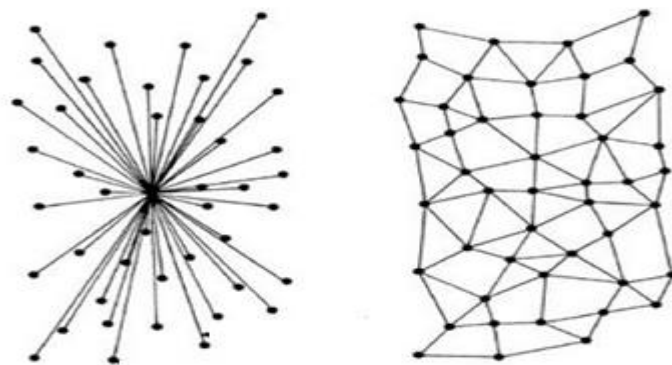


Fig 1.2 Model of centralized vs Ethereum app

1.4 PURPOSE

Blockchain technology can help decentralized the data and removing a central dependency that in turn will decentralize the power. Thus the main purpose of using blockchain platform for an EHR management system is to move from organization centric to patient centric applications. This technology will provide security and immutability to the data along with readily accessible data. The focal point of this application is to make the life of patients easier by providing them a platform where they can manage all their medical records easily and interact with the providers in an efficient way.

1.5 PROBLEM STATEMENT

There are a bunch of softwares and databases currently that are being used to store medical data. However, the problem of with current systems is that they are controlled by some kind of central authority. Moreover, the data is distributed among different organizations which cause accessibility issues when needed. Because of this, patients have to collect their medical data from various places when going for further treatment. In addition to that, the data is not secure and susceptible to mutation. It can also be misused by third parties. Thus the data is not in control of the patients. It is also not easy to manage since most of the data is paper based.

1.6 OBJECTIVES

The fundamental thought of this research is create a decentralized Electronic health records(EHR) application that will store patients' medical data. The focus of this research is to shift from provider-centric approach (as with the traditional systems) to patient-centric approach. This research expects to address this issue by moving from centralized databases to blockchain based data storage systems. This type of database that is distributed will help eliminate the giant organizations and bring the power back to the actual users. This approach will help security, immutability and authorized access to the patients of their medical data.

1.7 ORGANIZATION OF THE REPORT

The report has been divided into the following sections:

1. Section two describes the related work done using different approaches.
2. Section three focuses on the background and strategies of this research, it introduces Blockchain, Ethereum, Solidity, Angular etc. that is used in the project.
3. Section four focuses on the application functional and non-functional requirements along.
4. Section five of the report presents the implementation details of the project and the specific technologies used.
5. Section six presents experimental results.
6. The final section of this report summarizes the test results of the research and concludes some directions for future work.

CHAPTER 2

LITERATURE STUDY

Blockchain innovation empowers a decentralized and distributed network environment with no requirement for a central entity. Transactions are secure as well as trustworthy due to the utilization of cryptographic standards. Lately, blockchain innovation has turned out to be in popular, trendy and has infiltrated various areas, generally because of the prevalence of digital forms of money. One field where blockchain innovation has gigantic potential is healthcare because of the requirement for a more patient-centric way to deal with healthcare systems and to increase the interoperability of the systems and increase the precision of Electronic Healthcare Records (EHRs). Based on this, a survey was done [5], an analysis of the no of increasing interest of blockchain technologies in the healthcare domain. Firstly, web crawling was done to identify the papers related to blockchain which were further filtered based on the specific healthcare domain.

The research demonstrated that blockchain innovation in medicinal services is expanding and it is for the most part utilized for information sharing, overseeing health records and access control. Other scenarios are extremely uncommon. Given below is the graphical representation of the no of different types of papers growing over each year. It is expected that this no will increase in the coming years.

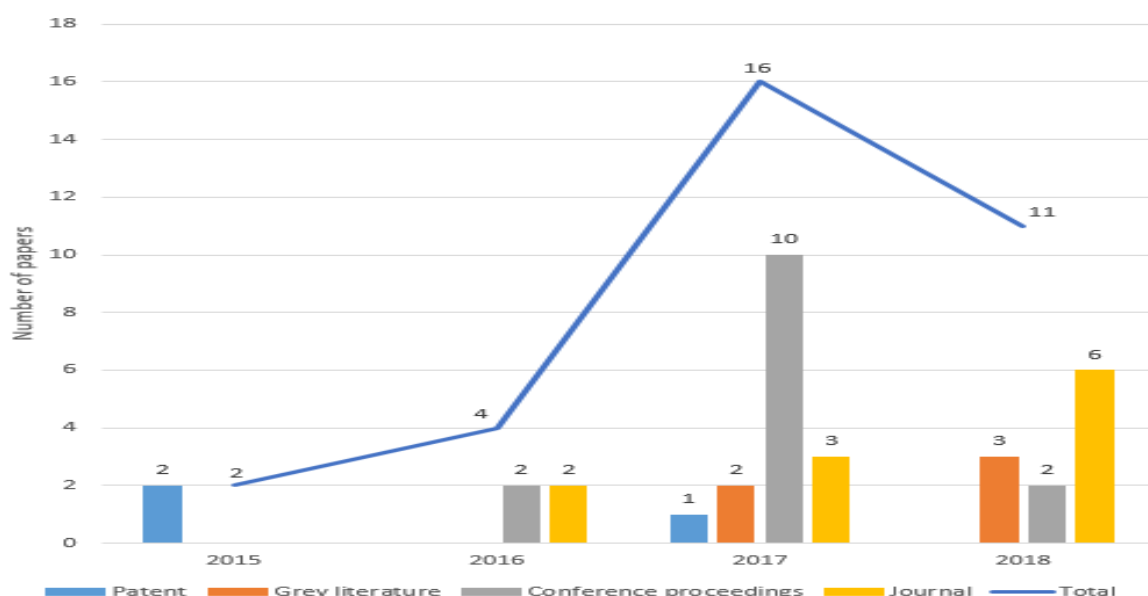


Fig 2.1 Papers published with the last few years

Sergey Avdoshin proposes a few use cases of the blockchain technology in the healthcare domain in his paper [6]. The broad use cases are: -

- **Blockchain for Electronic Medical Records** – Blockchain can be used to store digital medical records of patient data in a decentralized system. This system is believed to be of great benefit as it guarantees data integrity and protect patient privacy.
- **Blockchain for Tracking and Tracing Medical Fraud** – The market is flooded with fake medical drugs. Blockchain can be used to move the fake drugs out of the picture by tracking the drugs from their manufacture to their distribution. This will also help the country in an economical point of view.
- **Blockchain for Artificial Intelligence** – AI can be used for the analysis of complex medical data. This requires huge amount of data with diverse range to ensure accuracy and obtain effective results. Blockchain can provide a platform where patients, with the help of an advanced AI doctor, can discuss their medical data and get proper results. Moreover, this huge generated data can also be used as training sets for the AI systems.
- **Blockchain for Secure and Guaranteed Payments** – Financial sectors was the first domain in which Blockchain technology was used and it has proved to be very successful. Even for healthcare applications, Blockchain can facilitate payment of fees for treatments.
- **Blockchain for Medical Research** – Because of the large audience and huge amount of generated data, it will not become easy to find people who fit a certain medical history or and has the potential for clinical trials. They can even be incentivised with this platform. Moreover, abundant data will be available that can boost the medical research.

The focus of this report is the application of blockchain in the EHR systems. However, it faces many challenges like security and design of the system. Interoperability is one of the issues described in the paper written by Peng Zhang [4], who classifies it into three types.

Interoperability Level	Summary
Foundational	Enables; data interpretation is no required
Structural	Formats are defined for the exchanged data
Semantic	Exchanged data interpretation is required

Table 2.1 Interoperability levels

Foundational interoperability ensures that the data exchange is enabled between healthcare systems. It eliminates the extra step of interpretation of the data received by the providers. Structural interoperability defines the formats for the clinical data to be exchanged and ensures that the received data is preserved and can be interpretable at the data field level using predefined formats. Semantic interoperability ensures that the interpretability of exchanged data syntactically (structure) and semantically (meaning).

Other interoperability challenges faced by blockchain technologies are: -

Challenge	Summary
Evolvability	Blockchain is immutable but needs to support health system evolution
Storage	Inefficient and costly on-chain storage should be minimized
Privacy	Data transparency of the blockchain balanced with privacy concerns
Scalability	Relevant data should be filtered out form the blockchain

Table 2.2 Challenges in interoperability

Andreas Bogner, in his paper [8] explains the use case of Ethereum blockchain in creating a decentralized application running a smart contract. It focuses on the elimination of TTP (Trusted Third Party) [8]. The main components of the application are the smart contracts hosted on the blockchain, the local Ethereum client, and a web app. The web app provides a GUI for the local Ethereum client, which in turn interacts with the smart contract on the Ethereum blockchain.

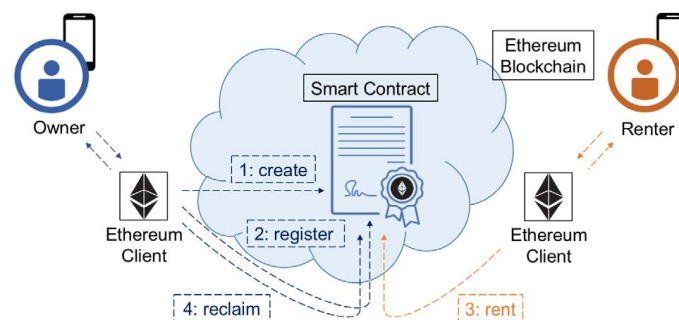


Fig 2.2 A Dapp in use

One of the applications of this technology is being worked on by the Truefield [7] team. The practice of medical recordkeeping reduces the healthcare cost and also assure quality health care. The Truefield platform aims at creating a patient centric approach to storing of medical information. They plan to achieve this by creating a unique digital medical IDs for each user. Their objective is to create an EHR system that is secure, private, confidential, easily accessible, cheap, facilitates information sharing etc. They plan on creating a decentralized application that will be accessible to users even without internet access through Trufield's unique protocol USSD. It will be a combination of centralized and decentralized platform. They plan to have their own tokens that can be owned, used and received as rewards.

CHAPTER 3

SYSTEM REQUIREMENTS

It is critical to characterize functional and non-functional necessities when making a dapp for ensuring that the system will enable the users to achieve the business targets. Useful necessities characterize what the framework does. Non-functional prerequisites direct and oblige the design. The functional requirements are like describing the characteristics of the system as it relates to the system's functionality. The non-functional requirement emphasises on the performance characteristics of the system.

3.1 FUNCTIONAL REQUIREMENTS

Account Creation: Since the application is deployed in the blockchain, it is of utmost importance that the accounts of each user is created successfully, satisfying the condition that it is unique. Metamask, a browser extension automatically takes care of that. It provides the user with a seed phrase with which it can generate random accounts. The user just has to remember the seed phrase and the user can get as many accounts as possible. The user can even recover his old accounts with that seed phrase.

Data creation: Providers should be able to create medical data for patients and upload them easily. Also, the data should not be reflected in the patients' medical history unless the patient has approved it. After the patient has approved the data, only then it should be appended to the blockchain. Data should not be created by any other means or third party users.

Data sharing: Providers and patients should be able to share data. The provider can ask a patient for his/her medical history and should be able to see it only after approval. The patient should not be notified if any provider wants to access his/her data. The patient can provide time period till which the provider can access the data. Also, the data cannot be modified by the provider.

3.2 NON-FUNCTIONAL REQUIREMENTS

Scalability: Analytics platform must be applicable to a machine or facility of any size. The application should be able to handle increasing no of users with no side effects on performance. The application is distributed across the network, and thus it won't have any issues scaling up or down and can do with no time loss.

Performance: The target of a mechanical examination stage is to give a creation office exact and opportune information especially the timely data. The performance of this application depends on the blockchain on which it is deployed, in our case, Ethereum. As the application is distributed, there are various nodes listening for the application requests, and thus the performance is very high.

Documentation: Coding standards are maintained throughout the project and the md files are written for the same.

Maintainability: This project can be maintained very easily. It can be modified and integrated with other computational and operational technologies since the whole project is made with angular and node which supports plugins and additional libraries to be easily added and deleted.

3.3 HARDWARE REQUIREMENTS

System: Core i5 Processor

Hard Disk :1 TB

Output Devices: Monitor

Input Devices: Keyboard, Mouse

RAM: 8GB.

3.4 SOFTWARE REQUIREMENTS

Operating system: Windows 8.1 or 10 /UBUNTU

Coding Languages: Solidity, HTML, CSS, Typescript

Softwares: Visual Studio, Node.js, Ganache

Web browser -Google chrome with Metamask extension

CHAPTER 4

PROPOSED METHODOLOGY

4.1 ISSUES IN EXISTING METHODOLOGY

The current systems use centralized databases to store patients' medical data which are in control of the big institutions scattered across various departments. The major issues thus created are –

- Inconsistency in medical data.
- The data is mutable.
- Lack of security.
- Unavailability of data when in need.
- The data can be used without the patient's consent.
- Much of the data is in paper form.

4.2 NEED FOR NEW METHODOLOGY

A new approach to manage medical data is needed. We propose a solution that creates a decentralized application with the help of blockchain technology. This new system will have the following characteristics –

- The control of medical data will transfer to the patient from the organisation.
- The data will be secure.
- The data will be immutable.
- The data will be readily available at one place whenever required.

4.3 METHOD OF IMPLEMENTATION

The proposed system will have backend part that will be done in blockchain technology. The front end part of the application will be used by the users to interact with the application directly. It will be made using angular.

- The new system proposed in the report makes use of the third generation of web i.e., decentralized web.
- The application is hosted on the Ethereum blockchain.
- The backend is developed using a contract based language “Solidity” specific to the Ethereum blockchain.
- The frontend of the application is browser based using web languages like HTML, CSS, JS. We have used the Angular framework to better manage the application and for testing purposes.
- The frontend will communicate with the blockchain using web3.js library.

Given below is a minimal overview of the application depicting the way the system will work.

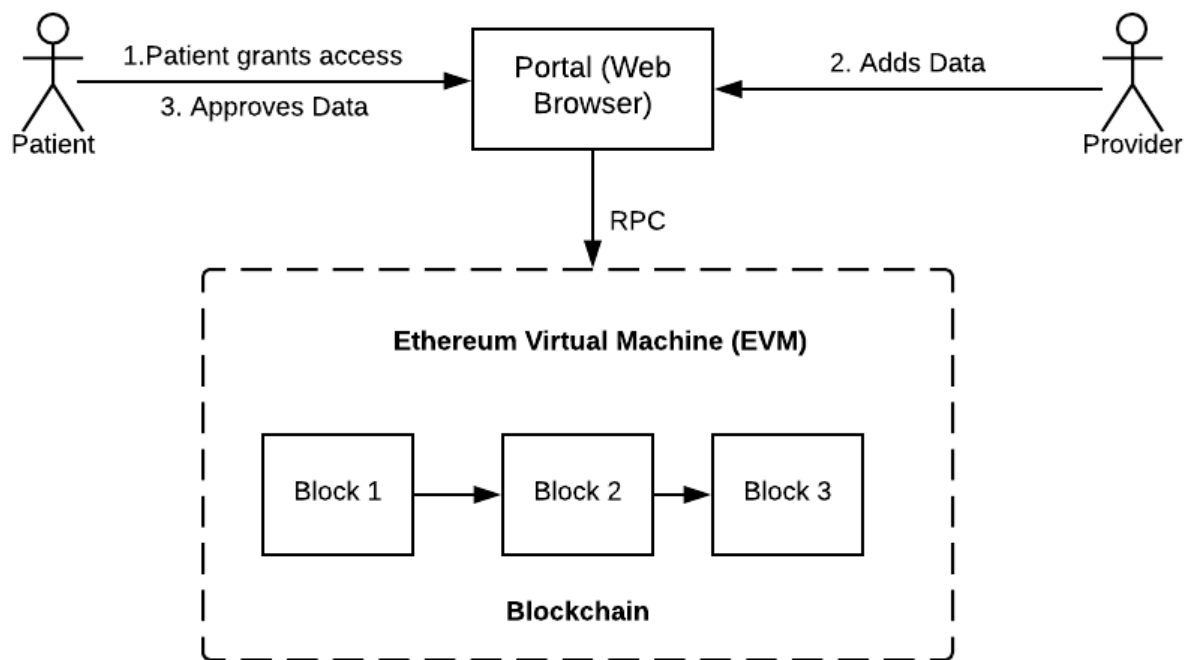


Fig 4.1 A Dapp user data flow overview

CHAPTER 5

IMPLEMENTATION

5.1 CHOOSING THE BLOCKCHAIN

The first implementation of the blockchain was Bitcoin whose use case is purely financial in nature. It is used to exchange money in digital currency form called 'Bitcoin'. However, in the past decade, many other blockchains have come up. Some of them are of purely financial nature, on the other hand, others can be used to create decentralized applications. Ethereum and Hyperledger are two of the most widely known blockchain platforms that can be used to create decentralized applications. Hyperledger is generally used for creating private blockchain applications. Our requirement is that of a public application. That is why, we have chosen Ethereum platform for creating our dapp.

5.2 ETHEREUM TESTNET

The project will be developed using Ethereum blockchain technology. For this, we need a blockchain to deploy and test our application throughout the development phase. Using the real Ethereum blockchain will cost a lot of money, so we have used Ganache. Ganache is a personal blockchain that runs on the local machine and can be used to deploy contracts, run applications and test them. It is available both as desktop application and command line tool. By default, it provides 10 accounts with 100 ethers each that we can use for transactions. We can see block history, events, logs etc.

5.3 METAMASK

Metamask is a browser extension that is specifically used for enabling the browsers to interact with the Ethereum blockchain. It uses a 12 words seed phrase which is used to generate random accounts. We can also recover our accounts with the seed phrase and password at any time. Metamask allows us to interact with the real Ethereum network, or test networks, or even instances running in the local machine like Ganache. We will be using this extension to run our application in the local machine in the port 8545.

5.4 WRITING SMART CONTRACTS

Smart contract is an agreement between two or third parties written in the form of code. Decentralized applications are defined by smart contracts. These smart contracts specify the rules/logic of the application so that it can work in a trust less environment. Solidity is a language for Ethereum blockchain that can be used to code smart contracts. We have used Solidity to write the logic for our application. Its syntax is similar to JavaScript and C++.

5.5 INTERACTION BETWEEN BROWSER AND BLOCKCHAIN

Web3.js is a JavaScript library that is used by the browsers to interact with the Ethereum blockchain. It can also interact with smart contracts and thus acts as a bridge between browsers and Ethereum blockchain. We will be using this library for all the interactions with the local blockchain with our front end code.

5.6 FRONT END

The front end code will be written using Angular framework. Angular is a framework that is used to build web applications for mobile and desktop. It provides a platform that separates the view and data logic and combines declarative templates, dependency injection, end to end tooling, and integrated best practices. We are using angular because it supports easy pluggable libraries through node.js features. We can easily use web3.js and other libraries and keep our bundle size to the minimum.

One more feature of Angular is that it uses Typescript, which is a superscript of JavaScript. It leads to much cleaner and efficient code.

Angular has a built in server that opens at localhost:4200 which can be used to deploy the application. It also provides Karma server that can be used to unit test the application using Jasmine testing.

5.7 SYSTEM ARCHITECTURE

The system consists of two main components: - front-end and back-end. The front-end is developed using Angular. It uses web3.js library to interact with the blockchain. The browser uses Metamask extension for account management. The back-end component consists of Ethereum blockchain. The decentralized application is built on top of this which is a public blockchain. In addition to this, Solidity programming language is used to write smart contracts which will be used to write the business logic of the application.

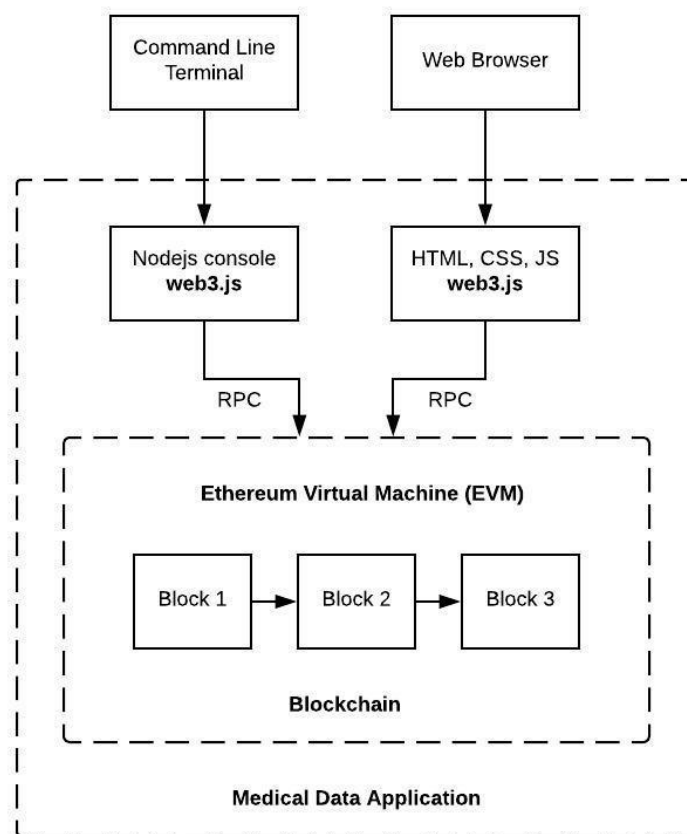


Fig 5.1 System architecture diagram

5.8 USE CASE DIAGRAM

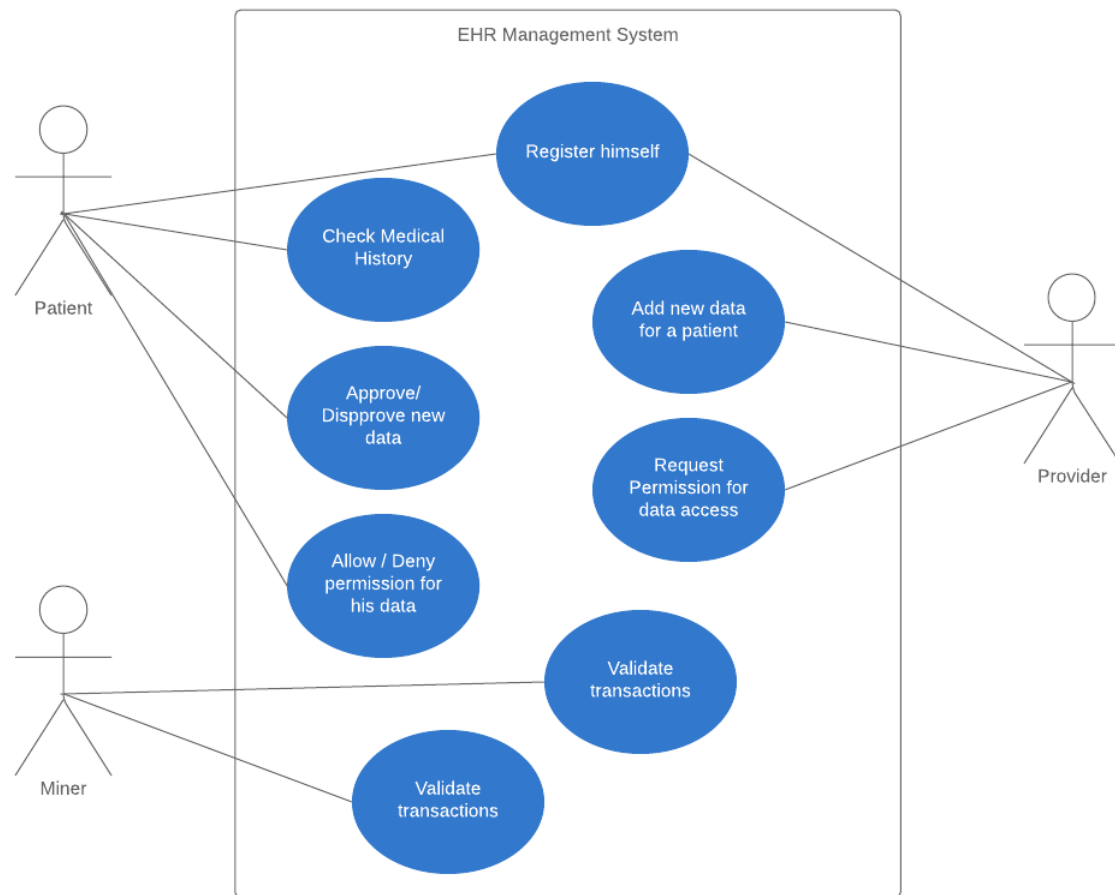


Fig 5.2 Use Case Diagram

5.9 MODULES AND ITS IMPLEMENTATION

5.9.1 Account Creation

To use this application, the users need a set of public and private key. The public key will be used for transactions, and the private key can be used to validate and sign the transactions. The private key is also used as a login mechanism. This process is too complex. That is why, we are using Metamask, a chrome extension that easily manages the accounts. It will create a seed phrase with which we can login. We can then add as many accounts as we want. Each the time the application is opened, the application identifies that is currently active in the Metamask, and uses that account for logging in the user automatically in the web application.

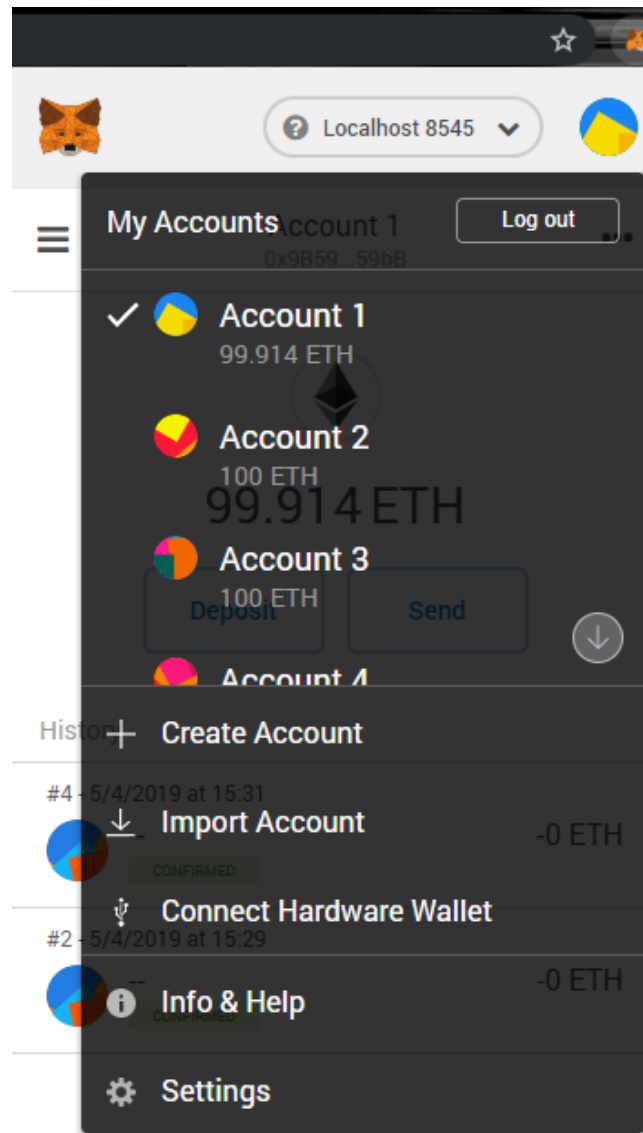


Fig 5.3 Metamask account management

5.9.2 Add New Medical Data

This is one of the main feature of the application. Instead of giving paper based records, the provider can upload the details of the medical data through a portal. The patient can then accept the record and then it will be appended to the blockchain as the patient record. The patient also has the option to deny the data if the provider has uploaded the wrong data.

The sequence diagram for the whole process is given below: -

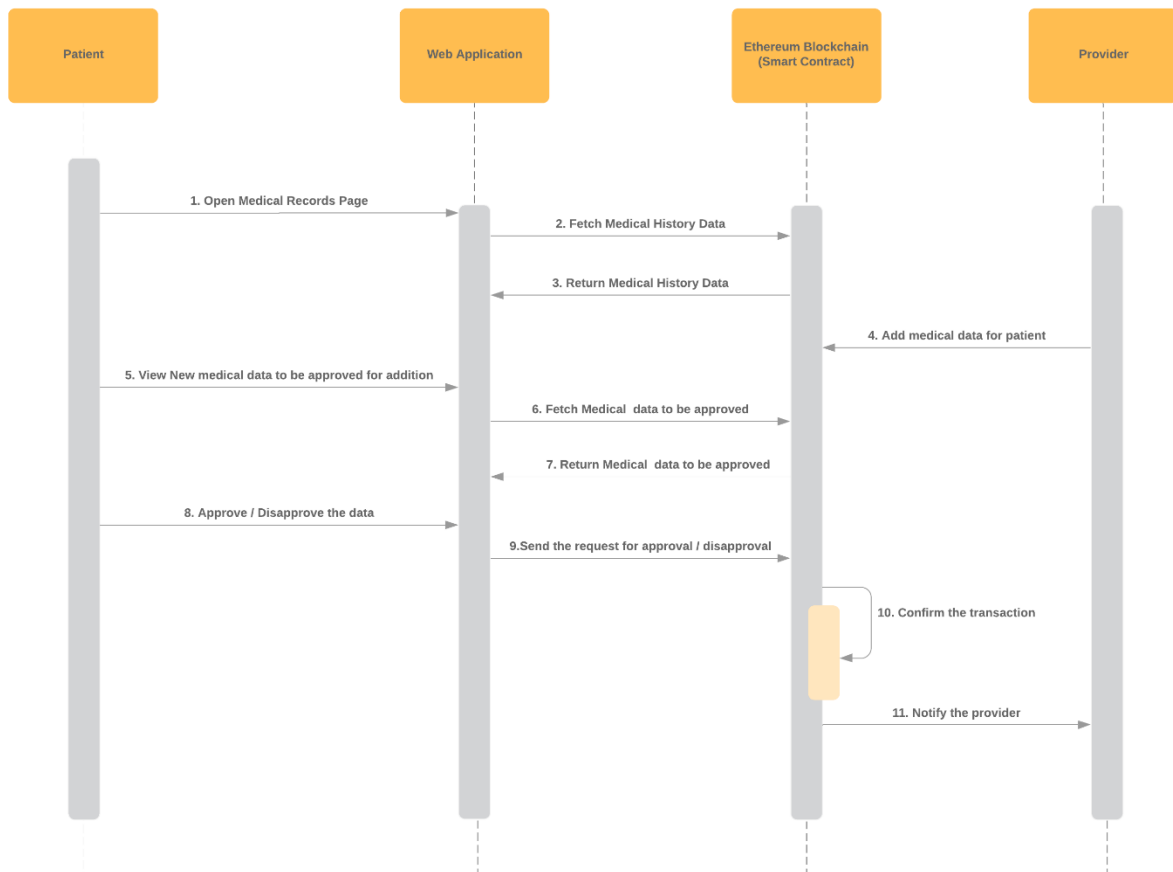


Fig 5.4 Sequence Diagram for adding new data

5.9.3 Data Sharing

Sometimes there is a need to share the patient data with the new provider to see the medical background history. The provider can ask for the patient data through the portal. The user has the option to either allow the access or deny it. If the user wants to share his data with the provider, he can also specify the time limit in hours and days. The limit specifies the time from that moment when the provider can access his data. After that, the data won't be accessible by the provider. Thus, this provides flexibility to share data without misuse.

The sequence diagram for the explained process is given below: -

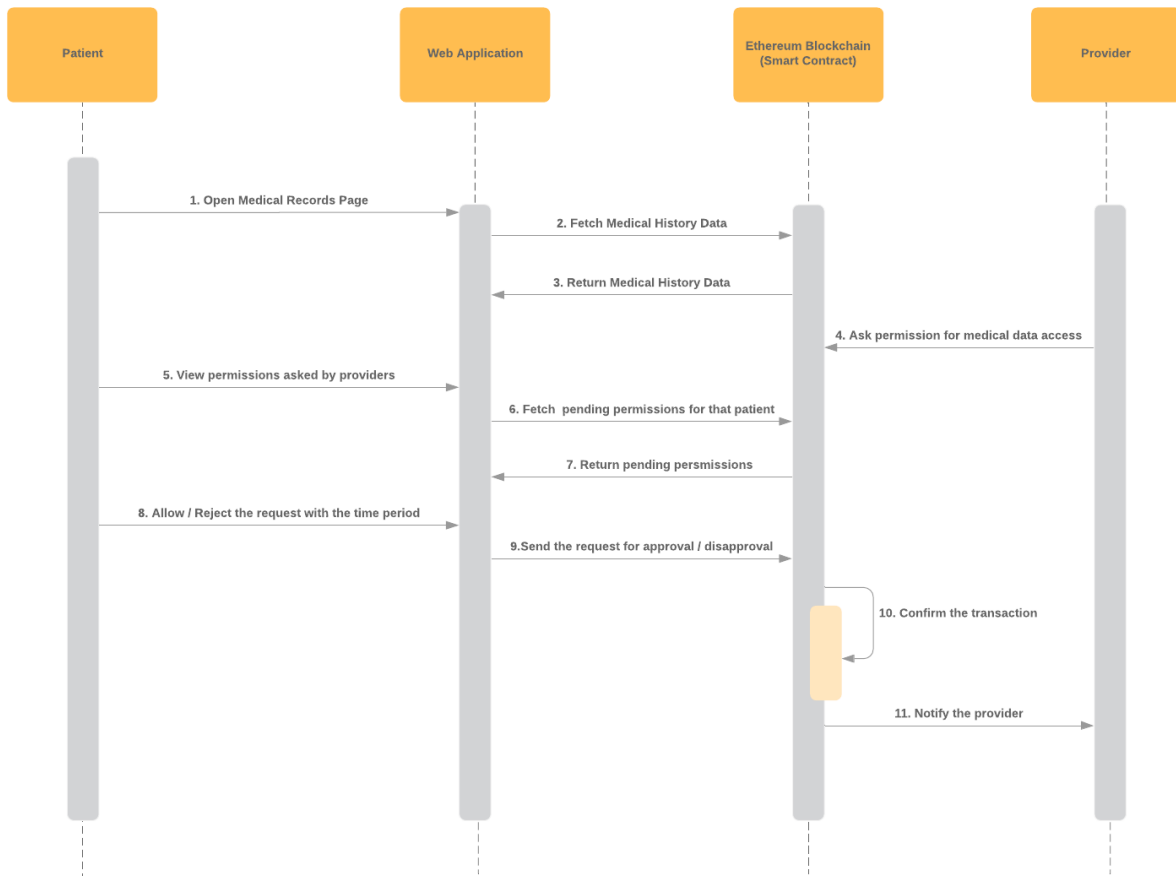


Fig 5.5 Sequence Diagram for allowing data access

5.9.4 Access Medical History

The patient can access his medical history. This is the landing page of the application if the user is logged in. The front end makes a call to the backend to get the medical history from the smart contract of that particular user. The user is identified by its public key. On sending the request, the account address is also sent which is then used to fetch the corresponding details. The medical data is then received as an object which is then converted to subject using rxjs library to reduce the delay.

5.10 TOOLS USED

We have used various tools and modules of JavaScript to create the required application. A short description of each of the tools has been given below.

Ganache: - It is a personal blockchain that runs on the local machine and can be used to deploy contracts, run applications and test them. It is available both as desktop application and command line tool. By default, it provides 10 accounts with 100 ethers each that we can use for transactions. We can see block history, events, logs etc.

Metamask: - Metamask is a browser extension that is specifically used for enabling the browsers to interact with the Ethereum blockchain. It uses a 12 words seed phrase which is used to generate random accounts. We can also recover our accounts with the seed phrase and password at any time. It allows us to interact with the real Ethereum network, or test networks, or even instances running in the local machine like Ganache.

Truffle: - Truffle is a development framework used for contract compilation, linking, deployment, linking etc. It also provides automate contract testing. This framework allows us to manage our smart contracts efficiently, compile them and migrate them to the blockchain network. We can also test our smart contracts with its supported JavaScript libraries or using solidity contracts itself.

Solidity: - Solidity is a contract oriented programming language that is used to write smart contracts for Ethereum blockchain. Its syntax is similar to that of JavaScript and that of C++. It is used to form agreements between two or more parties without using a third party. It is also used to write the business logic of a decentralized application.

Web3.js: - It is a JavaScript library used for browser interaction with the blockchain. It contains functions and global variables that enables browsers to interact the Ethereum blockchain. It also allows us to interact with the smart contracts as well.

Angular: - Angular is a framework that is used to build web applications for mobile and desktop. It provides a platform that separates the view and data logic and combines dependency injection, declarative templates, end to end tooling and integrated best practices. We are using angular because it supports easy pluggable libraries through node.js features. One more feature of Angular is that it uses Typescript, which is a superscript of JavaScript. It leads to much cleaner and efficient code.

Visual Studio Code: - It is an advanced text editor that is best suited for web applications. Its power comes from the fact that it can work best with node.js and other web frameworks. It has a rich ecosystem and provides plugins for many other languages like C++, Python etc. It uses colour coding, auto completion of code, auto alignment etc.

5.11 RESULTS

Below we have given the screenshots of the different modules and functionalities of the project executed from the chrome browser and ganache as the local Ethereum network.

5.11.1 FRONT END OF THE APPLICATION

The user will land on the page that shows him his medical records that has been added by various providers and was accepted by him.

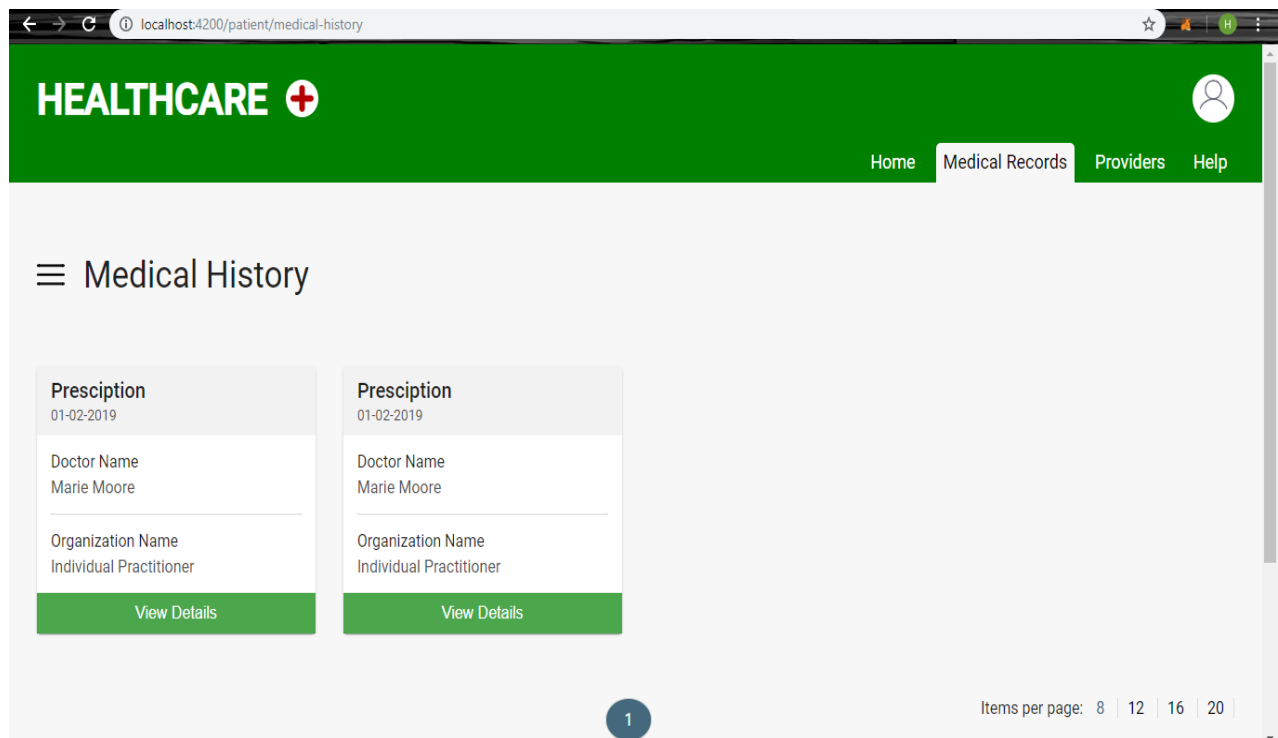


Fig 5.6 Medical History Records

The user can click on 'View Details' to view more details on that record.

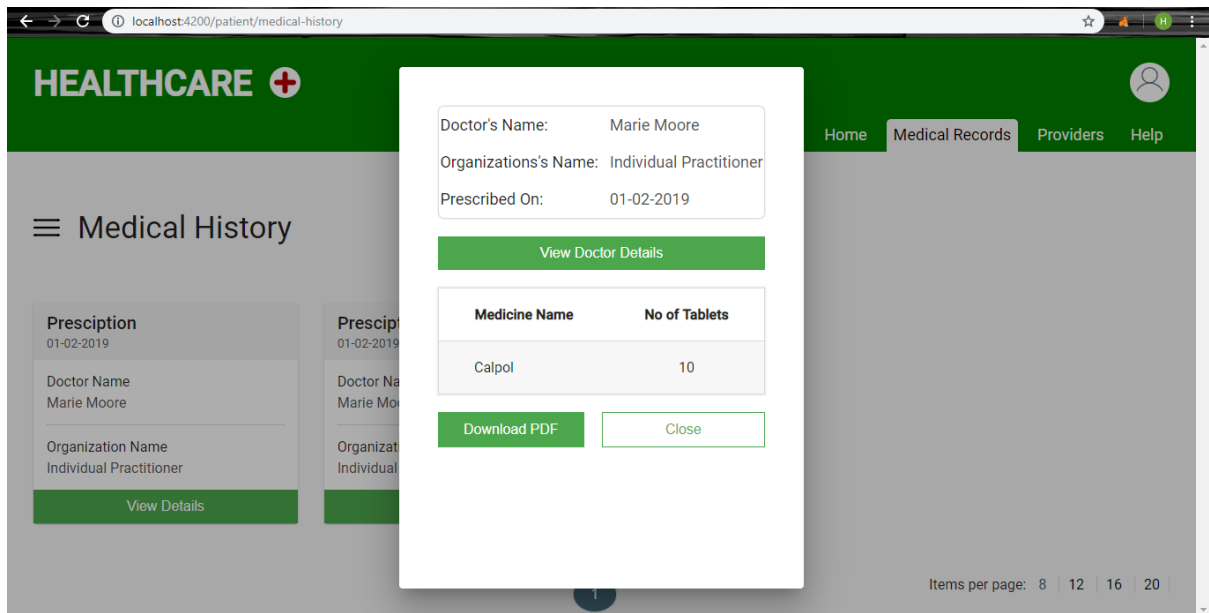


Fig 5.7 Individual Medical Records Page

The user can also navigate to the pending medical approvals page where he can approve or deny a new record to be added.

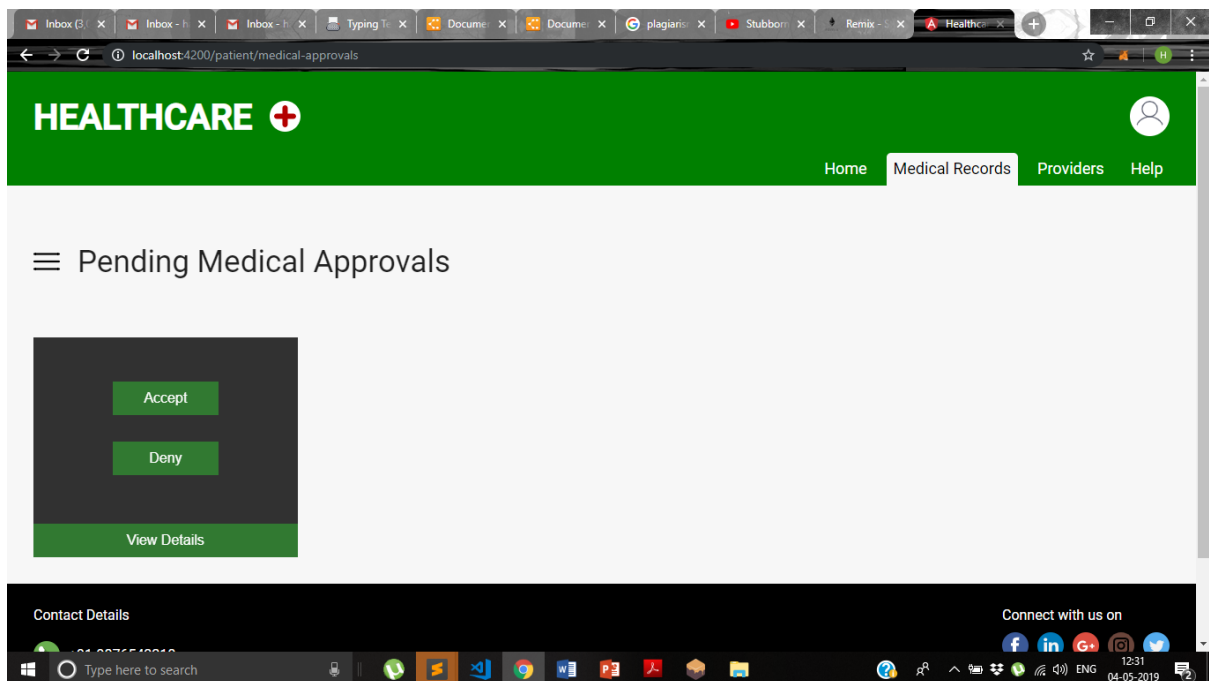


Fig 5.8 New medical data waiting for approval

The user can navigate on the pending permissions page where he/she can share the data with the provider who has requested. He/she can also specify the time period till when the provider can view his/her records.

The screenshot shows a web browser at the URL `localhost:4200/patient/medical-permissions`. The page has a green header with the 'HEALTHCARE' logo and a user profile icon. Navigation links include 'Home', 'Medical Records', 'Providers', and 'Help'. The main section is titled 'Manage Medical Permissions'. A permission request card is displayed with the following details:

Doctor Name Marie Moore Organization Name Individual Practitioner Date 13-04-2019	I need access to all your medical data so that I can better diagnose your problem. Please share it as soon as you can.
--	--

At the bottom of the card are two buttons: 'Deny' and 'Allow'. The footer contains contact details: '+91-9876543210' and 'healthcare-support@gmail.com', along with social media icons for Facebook, LinkedIn, Google+, Instagram, and Twitter.

Fig 5.9 Permission Page to allow access

This screenshot shows the same 'Manage Medical Permissions' page, but the permission request card now includes time selection fields. The details are as follows:

Doctor Name Marie Moore Organization Name Individual Practitioner Date 13-04-2019	I need access to all your medical data so that I can better diagnose your problem. Please share it as soon as you can.
--	--

Below the text in the card, there are input fields for 'Hours' and 'Days', followed by 'Allow' and 'Cancel' buttons. The rest of the page layout, including the header, navigation, and footer, remains identical to the previous figure.

Fig 5.10 Permission Page to enter time details

Whenever the user sends a request to initiate a transaction in the blockchain, the Metamask extension in his browser will pop up asking for the confirmation. It will also show the details like the gas cost, ether etc.

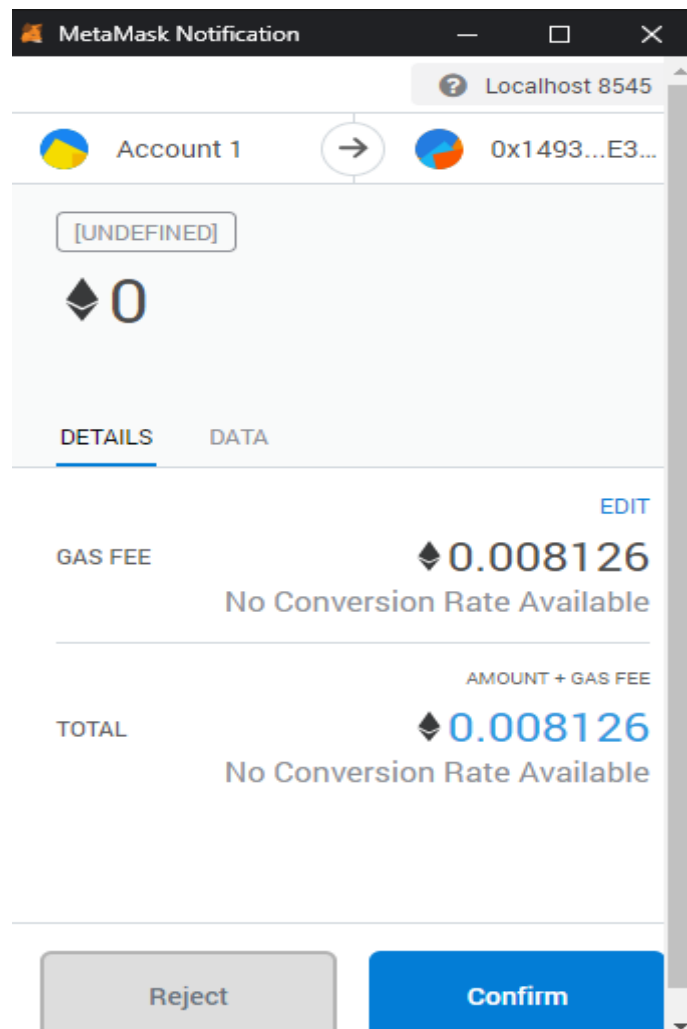
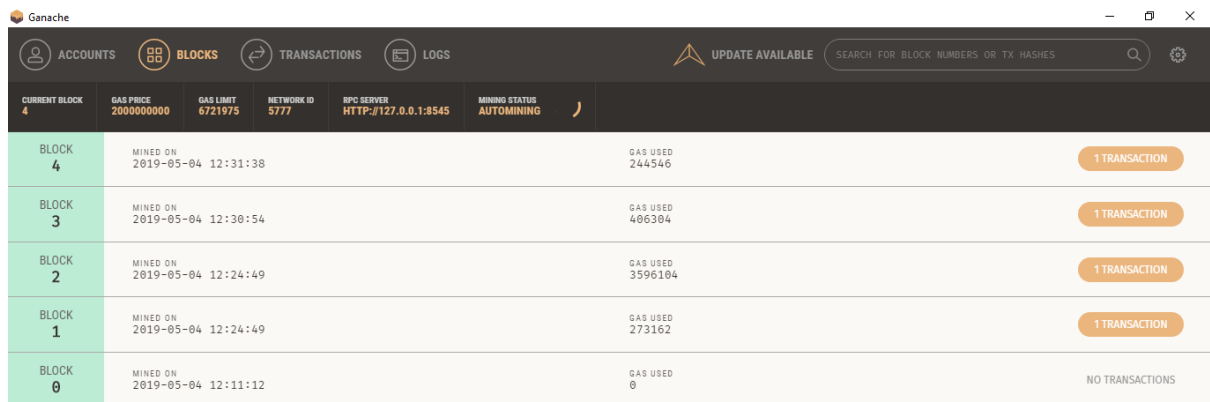


Fig 5.11 Metamask popup for confirming new transaction

5.11.2 BACK END OF THE APPLICATION

For the back end, we are using 'Ganache' as the local Ethereum test network where we can see the transactions, blocks and more details. By default, Ganache provides us with 10 accounts with 100 eth each.

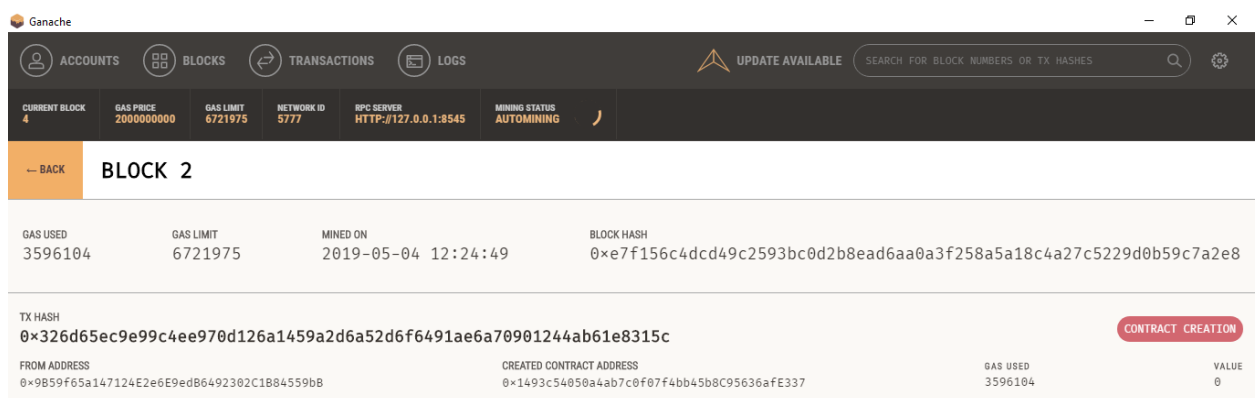
As the transactions are approved, the blocks get created. We can also see which transactions were contact creating and which were contract calls.



The screenshot shows the Ganache application window with the 'BLOCKS' tab selected. The top bar displays network information: Current Block 4, Gas Price 2000000000, Gas Limit 6721975, Network ID 5777, RPC Server HTTP://127.0.0.1:8545, and Mining Status AUTOMINING. Below this is a table of mined blocks.

BLOCK	MINED ON	GAS USED	TRANSACTIONS
BLOCK 4	2019-05-04 12:31:38	244546	1 TRANSACTION
BLOCK 3	2019-05-04 12:30:54	486304	1 TRANSACTION
BLOCK 2	2019-05-04 12:24:49	3596104	1 TRANSACTION
BLOCK 1	2019-05-04 12:24:49	273162	1 TRANSACTION
BLOCK 0	2019-05-04 12:11:12	0	NO TRANSACTIONS

Fig 5.12 Block mining in Ganache



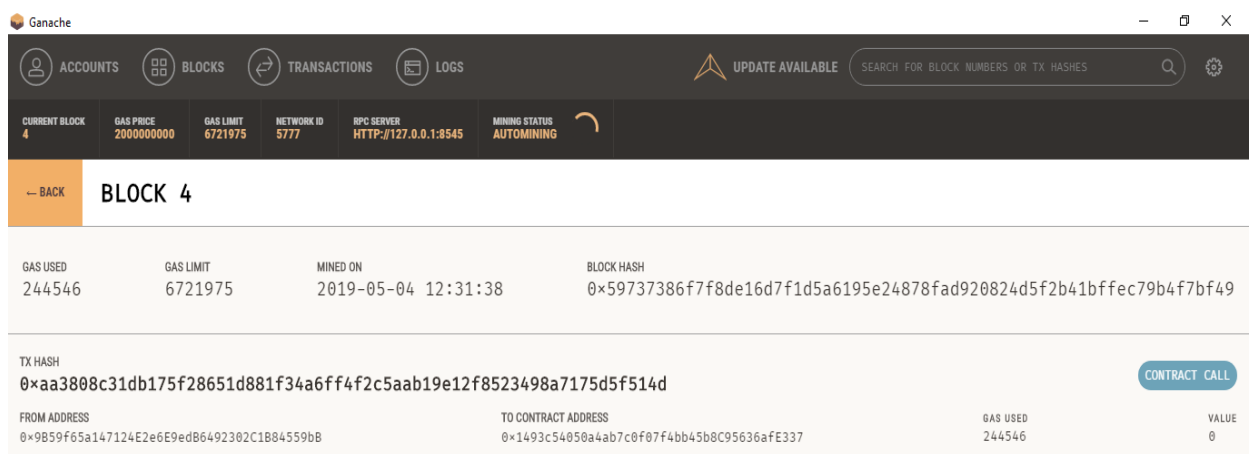
The screenshot shows the Ganache application window with the 'BLOCKS' tab selected and 'BLOCK 2' highlighted. The top bar displays network information: Current Block 4, Gas Price 2000000000, Gas Limit 6721975, Network ID 5777, RPC Server HTTP://127.0.0.1:8545, and Mining Status AUTOMINING. Below this is a table of mined blocks.

BLOCK	MINED ON	GAS USED	TRANSACTIONS
BLOCK 2	2019-05-04 12:24:49	3596104	1 TRANSACTION

Transaction details for Block 2:

TX HASH	FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0x326d65ec9e99c4ee970d126a1459a2d6a52d6f6491ae6a70901244ab61e8315c	0x9B59f65a147124E2e6E9edB6492302C1B84559bB	0x1493c54050a4ab7c0f07f4bb45b8C95636afE337	3596104	0

Fig 5.13 Contract creation call



The screenshot shows the Ganache application window with the 'BLOCKS' tab selected and 'BLOCK 4' highlighted. The top bar displays network information: Current Block 4, Gas Price 2000000000, Gas Limit 6721975, Network ID 5777, RPC Server HTTP://127.0.0.1:8545, and Mining Status AUTOMINING. Below this is a table of mined blocks.

BLOCK	MINED ON	GAS USED	TRANSACTIONS
BLOCK 4	2019-05-04 12:31:38	244546	1 TRANSACTION

Transaction details for Block 4:

TX HASH	FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xaa3808c31db175f28651d881f34a6ff4f2c5aab19e12f8523498a7175d5f514d	0x9B59f65a147124E2e6E9edB6492302C1B84559bB	0x1493c54050a4ab7c0f07f4bb45b8C95636afE337	244546	0

Fig 5.14 Contract call to change contract state

Ganache

ACCOUNTS BLOCKS TRANSACTIONS LOGS UPDATE AVAILABLE SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 2	GAS PRICE 2000000000	GAS LIMIT 6721975	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:8545	MINING STATUS AUTOMINING	CLEAR LOGS
--------------------	-------------------------	----------------------	--------------------	-------------------------------------	-----------------------------	------------

```

[1:37:38 PM] <      "code": -32000,
[1:37:38 PM] <      "data": {
[1:37:38 PM] <        "stack": "Error: LevelUpArrayAdapter named 'blocks' index out of range: index 4; length: 3\n    at C:\\Program
Files\\WindowsApps\\Ganache_1.2.2.0_x64_zh355ej5cj694\\app\\resources\\app.asar\\node_modules\\ganache-
core\\lib\\database\\leveluparrayadapter.js:51:23\n    at C:\\Program
Files\\WindowsApps\\Ganache_1.2.2.0_x64_zh355ej5cj694\\app\\resources\\app.asar\\node_modules\\ganache-
core\\lib\\database\\leveluparrayadapter.js:25:5\n    at C:\\Program
Files\\WindowsApps\\Ganache_1.2.2.0_x64_zh355ej5cj694\\app\\resources\\app.asar\\node_modules\\ganache-core\\node_modules\\level-
sublevel\\shell.js:102:12\n    at C:\\Program
Files\\WindowsApps\\Ganache_1.2.2.0_x64_zh355ej5cj694\\app\\resources\\app.asar\\node_modules\\ganache-core\\node_modules\\level-
sublevel\\nut.js:122:19\n    at C:\\Program
Files\\WindowsApps\\Ganache_1.2.2.0_x64_zh355ej5cj694\\app\\resources\\app.asar\\node_modules\\ganache-
core\\node_modules\\cachedown\\index.js:53:7\n    at _combinedTickCallback (internal/process/next_tick.js:131:7)\n    at process._tickCallback
(internal/process/next_tick.js:180:9)",
[1:37:38 PM] <      "name": "Error"
[1:37:38 PM] <    }
[1:37:38 PM] <  }
[1:37:38 PM] < }
[1:37:39 PM] eth_blockNumber
[1:37:39 PM] eth_blockNumber
[1:37:39 PM] > {
[1:37:39 PM] >   "id": 7096153141444992,
[1:37:39 PM] >   "jsonrpc": "2.0",
[1:37:39 PM] >   "params": [],
[1:37:39 PM] >   "method": "eth_blockNumber"
[1:37:39 PM] > }
[1:37:40 PM] < {
[1:37:40 PM] <   "id": 7096153141444992,
[1:37:40 PM] <   "jsonrpc": "2.0",
[1:37:40 PM] <   "result": "0x02"
[1:37:40 PM] < }

```

Fig 5.15 Logs stored in the blockchain

CHAPTER 6

TESTING

6.1 TESTING OBJECTIVES

The core purpose of testing is to find out unwanted errors and bugs. Testing is the methodology of trying to figure or discover every conceivable faults, defects or weaknesses in a working project. It provides different methods to check the expected functionality of the components, sub-components, dependencies, integration mechanisms and security systems etc. The basic form of testing that can be performed in a web application is unit testing and integration testing.

6.2 FRONT END UNIT TESTING

The front end is made using JavaScript and Angular. The unit testing for the logic of front end is done using Jasmine. Jasmine is JavaScript framework that follows BDD (behaviour driven development) procedure to make sure that all JavaScript statements including functions are properly unit tested. It provides spies and stubs for the service and other dependencies so that the components are properly unit tested. In this project, all the components and services were unit tested with Jasmine. The testing report and the code coverage report is given below: -

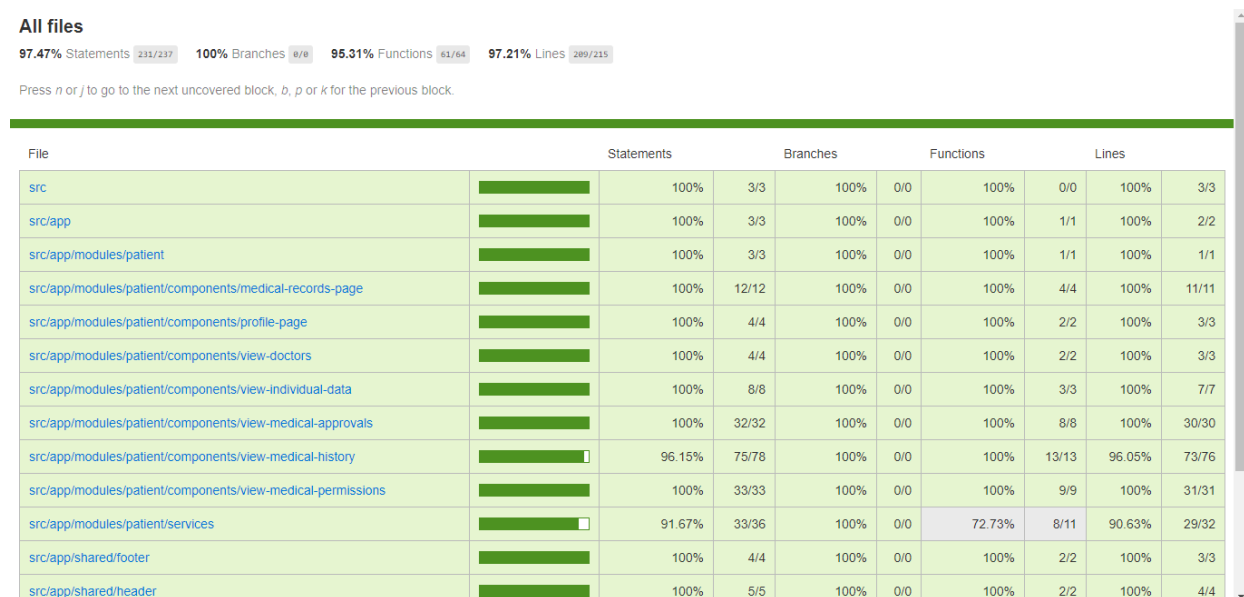


Fig 6.1 Coverage report for jasmine testing

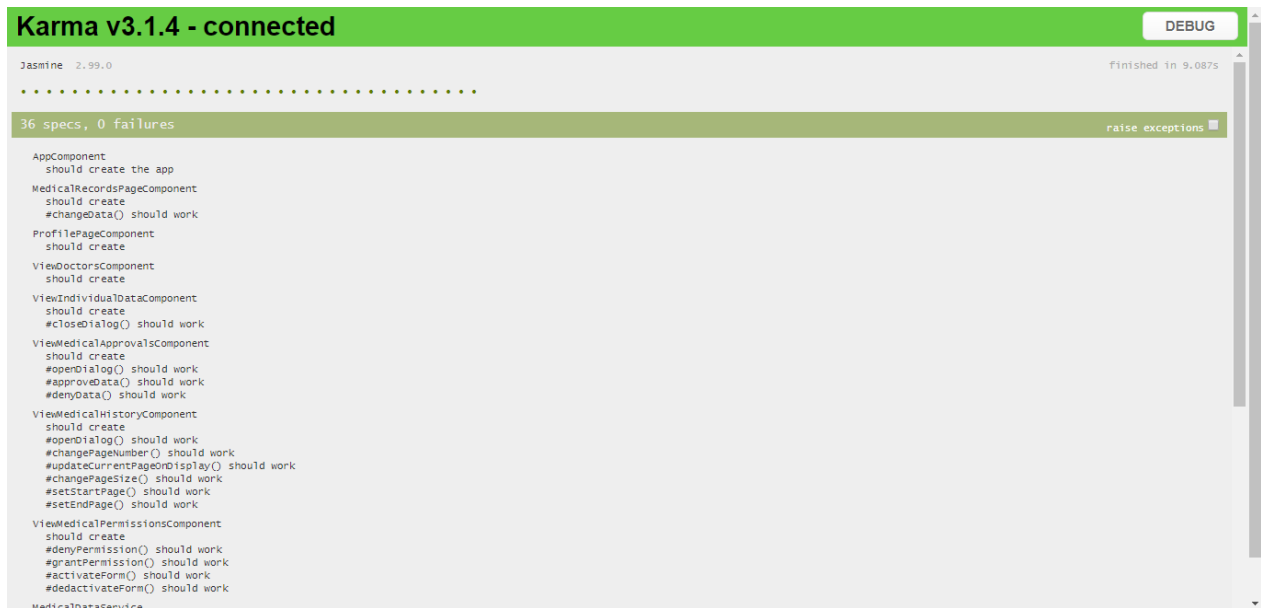


Fig 6.2 Spec files testing for jasmine testing

6.3 BACK END TESTING

The back end is written using Solidity programming language. The Solidity codes can be managed with the help of Truffle framework. Truffle provides testing the Solidity code with Solidity contracts itself which is built on top of the Mocha framework. It uses assert statements from the Mocha framework. This kind of testing is done to ensure that the functions of the smart contract work as expected and the values are modified according to the use case. The testing report has been given below: -

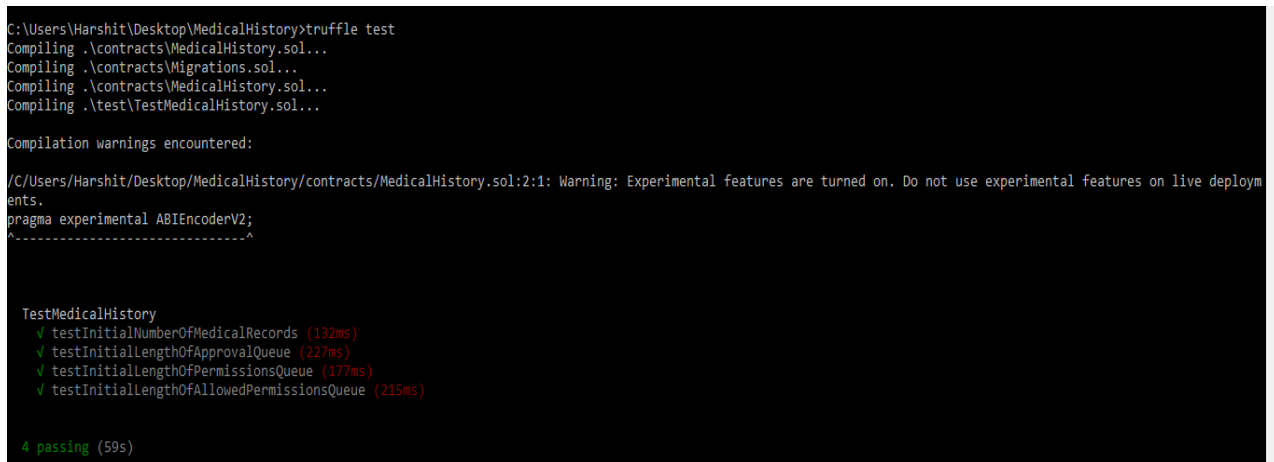


Fig 6.3 Testing report for solidity code testing

The test results have been summarized below: -

Test Case Name	Test Case Description	Expected Result	Actual Result	Test Status (P/F)
New Account Initial Data	There should be no initial data for a new user	There should not be any medical records for a new user	There was no medical records found for a new user	P
Landing Page	The landing page of the application when opened	User lands on the home page when the application opens	User landed on the hope page when application opened	P
Medical History Page	Medical history data should be fetched	Medical data of patient should be fetched	Medical data of the patient was fetched	P
Allow pending approvals	Pending approval can be approved	User can approve new data to be added	User approved new data to be added	P
Deny pending approvals	Pending approval can be denied	User can deny new data to be added	User denied new data to be added	P
Allow data access	Permission to access data by provider	User can give permission to access data	User gave permission to access data	P
Deny data access	Permission to access data by provider	User can deny permission to access data	User denied permission to access data	P
Time limit	Specify time period for data sharing	User can specify time limit to share data	Data was not accessible after the time limit	P

Table 6.1 Testing report

CHAPTER 7

CONCLUSION

Blockchain innovation is slowly ending up mainstream. The blockchain has huge advantages, from decentralization, to protection and affordability, to security and immutability. Both experts and associations will certainly work quicker and more effectively, with respect to how safe and reliable the data accessibility is. It can prove to be a revolutionary change in the industry of healthcare with various use cases like drug tracking, EHR management systems etc. The public would be enormously benefitted with the EHR systems hosted on the blockchain as it will provide transparency, security and control over their data. Now, they won't have to depend on the big organisations for their own medical data. This revolutionary technology will exponentially increase the data accessibility for the experts as well as public.

The application focuses on the patient-centric approach rather than provider-centric approach. It transfers the control of the data back to the patients in comparison to the current systems in which all the data resides with the institutions. Now, the data is readily and immediately accessible, all at one place without compromising the security. This application has made it quite easier for patients to manage their medical records. Now, they don't have to be worried about paper based records while visiting different providers.

It also facilitates data sharing with chosen providers. In contrast to the older systems, the medical data cannot be misused by third parties as the data is not accessible by them. Patients can choose who to share their data with and can also specify the time till when the provider can assess his/her data. Thus the new system is built completely focusing on the patients.

CHAPTER 8

FUTURE ENHANCEMENT

This application currently has very few use cases which satisfies just the most necessary functionalities of an EHR management system. The application can further be extended to store all types of medical documents other than prescriptions like X-rays, blood reports etc. Furthermore, the application can be extended to provide templates to fill in the details directly and then generate and upload reports directly. This can save the overhead created from the file uploads. One another approach can be to store the hash of the files instead of storing the whole file in the blockchain. The actual files will be stored in the users' local file system and just the hash will be stored in the decentralized web. This way the load on the blockchain will be less and it will improve the performance of the system drastically.

One of the features that can be added to such a kind of medical application is the payments. While approving the medical data, i.e., expecting some kind of medical report from the provider, that patient can also send the respective amount of ether as a payment, which if not sent, the data won't be added to the patient's profile. To implement these kinds of features, the functions in the smart contract can be made payable by using the 'payable' keyword that can accept payments. For this purpose, we can either create our own cryptocurrency for the application or use ether directly. Both the approaches will have different effect. Creating a cryptocurrency for the application alone will bring market value of the application into the picture.

The application can also be extended to be used by the researchers and governments. Since this application will generate so much of medical data of individual patients, it can be a good source of data for researchers and governments. The diversity of the data will be huge in all aspects like people, regions, type of medical data etc. This data source can be used to do researches for finding medical patterns by the researchers. It can also be used by governments to find out ways to improve the medical practices in the country and find out regions of poor medical health conditions etc.

CHAPTER 9

REFERENCES

- [1] Matthias Mettler, "Blockchain Technology in Healthcare" *Boydak Strategy Consulting AG Freienbach, Switzerland*.
- [2] Zainab Alhadhrami, Salma Alghfeli, Mariam Alghfeli, Juhar Ahmed Abedlla and Khaled Shuaib, "Introducing Blockchain for Healthcare," *College of Information Technology, United Arab Emirates University*.
- [3] Claude Pirtle, Jesse Ehrenfeld, "Blockchain for Healthcare: The Next Generation of Medical Records ?," *Springer Science + Business Media, LLC, part of Springer Nature 2018*.
- [4] Peng Zhang, Douglas C. Schmidt, and Jules White, "Blockchain Technology Use Cases in Healthcare," *Vanderbilt Universtiy, Nahsville, TN*.
- [5] Marko Holbl, Marko Kompara, Aida Kamisalic, Lili Nemec Z;atolas, "A Systematic Review of the Use of Blockchain in Healthcare," *University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia*.
- [6] Sergey Avdoshin, Elena Pesotskaya, "Blockchain Revolution in the Healthcare Industry," *National Research University Higher School of Economics, Moscow, Russian Federation*.
- [7] Kwabena Owusu, "Trufield," *trufield.com*.
- [8] Andreas Bogner, Arne Meeuw, Mathieu Chanson, "A Denetralised Sharing App running a Smart Contract on the Ethereum Blockchain," *Research Gate, Conference Paper November 2016*.
- [9] Zhang, P. White, J. Schmidt, D.C., and Lenaz, "Applying Software Patters to address interoperability in Blockchain based healthcare apps," *arXiv preprint arXiv: 1706.03700, 2017*.
- [10] Middleton B., Bloomrosen M., Dente M.A., Hashmat B., Koppel R., Overhage J. M., Payne T. H., Rosenbloom S. T., Weaver C., Zhang J., "Enhancing patient safety and quality of care by improving the usability of electronic health record systems: recommendations from AMIA," *Journal of the American Medical Informatics Association, 2013*.

- [11] Zyskind, Guy, and Oz Nathan, "Decentralized privacy: Using blockchain to protect personal data," *Proceedings of the 2015 IEEE Security and Privacy Workshop, May 21-21, 2015, San Jose, California, USA*.
- [12] A. Azaria, A. Ekblaw, T. Vieira and A. Lippman, "MedRec: Using Blockchain for Medical Data Access and Permission Management," *2016 2nd International Conference on Open and Big Data (OBD), Vienna, 2016*.
- [13] Hogan, S., Fraser, H., Korsten, P., V., Gopinath R., "Healthcare rallies for blockchain: keeping Patients at the center," *IBM Corporation Homepage. Accessed 20 Jan 2018*.
- [14] Steffens, B., Billiot, J., Marques, A., Gawas, D., Harmalkar, O., "Facilitate health care on block chain," *MediBond Homepage*.
- [15] Bresnick, Jennifer, "Illinois to Uses Blockchain for Healthcare Credential Management," *healthianalytics.com/news/ August 2017*.

APPENDIX A: - SUPPLEMENTARY INFORMATION

1 CRYPTOCURRENCY

A cryptocurrency is form of digital currency or virtual currency that uses cryptography for providing security, thereby making it difficult to counterfeit. It is used in decentralized context and not issued by any central authority. It is immune to government interference or manipulation. A cryptocurrency is built on blockchain that only exists online.

2 DECENTRALIZED APP (Dapp)

A Dapp is an autonomously operated open-source application that is not controlled by any central authority. Instead it is decentralized over the web built on top of blockchain technologies. The data of a Dapp is stored cryptographically in a public, decentralized blockchain to avoid central points of failure. It uses cryptographic tokens for monetizing the application.

3 DIGITAL ASSET

A digital asset is anything that can be of financial value and can be monetized using digital currencies. It could be any asset that lacks physical substance and can be owned or produced value from, such as house, movie, images, software etc.

4 ELECTRONIC HEALTH RECORDS (EHRs)

EHRs are digital records of a patient's paper based medical records. It can be any type of record including prescriptions, X-ray reports, etc. EHRs are stored in digital formats. The benefit of EHRs is that they make information readily available and are only authorized to healthcare professionals and the respective patients. They contain medical histories of patients and can also store information other than standard data collected in provider's office, such as lab results, medications, treatment plans, allergies etc.

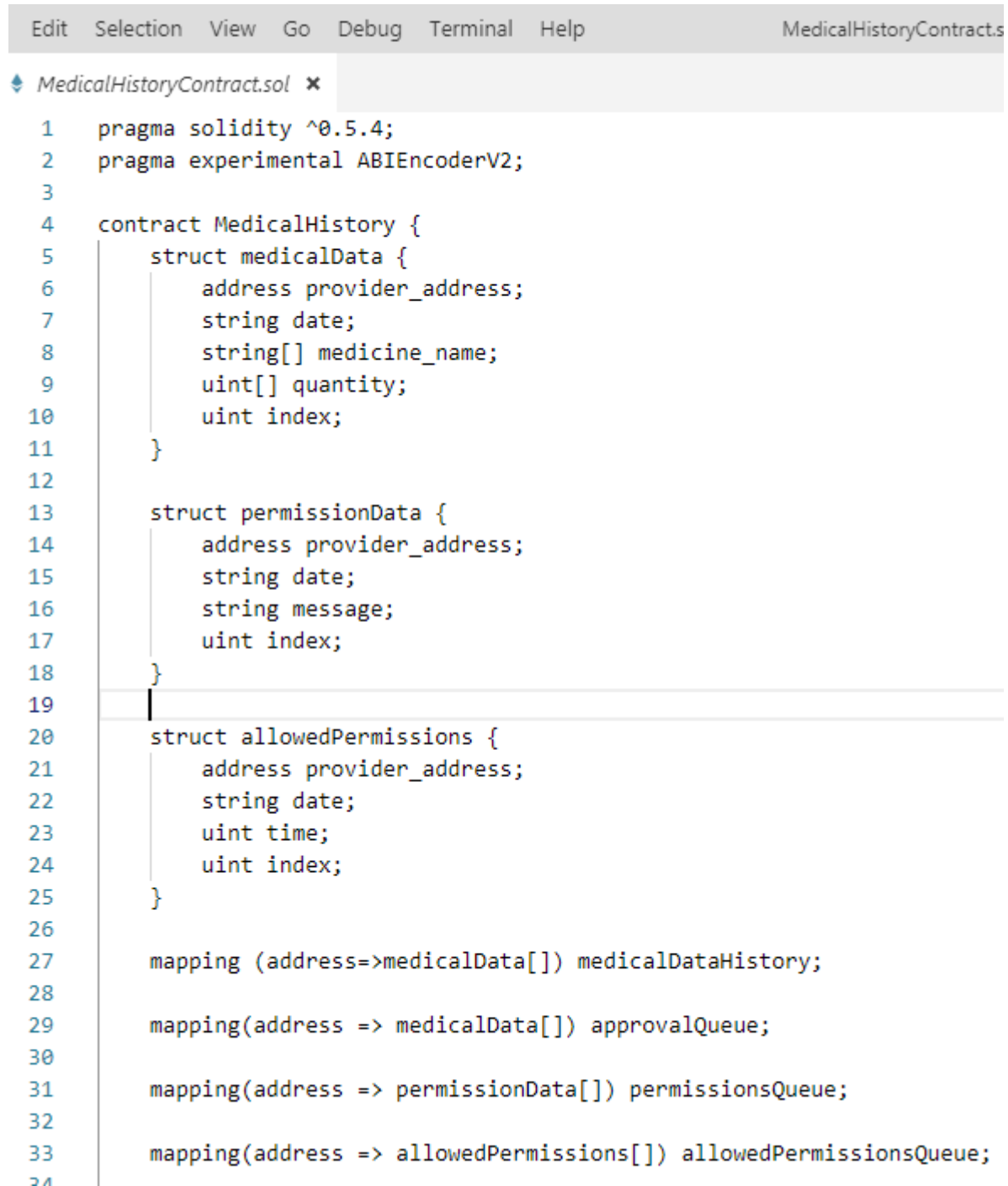
5 SMART CONTRACT

They are built on top of blockchain technologies like Ethereum. They are code that provides the logic for a decentralized app. They also control the exchange and redistributions of any digital asset between two or more parties. They provide rules and agreements that establishes trust between the involved parties without involving any third party. They also enable DApps to interact with the blockchain and support on-chain storage.

APPENDIX B: SOURCE CODE

A: SOLIDITY CODE (BACKEND)

MedicalHistoryContract.sol



```
1  pragma solidity ^0.5.4;
2  pragma experimental ABIEncoderV2;
3
4  contract MedicalHistory {
5      struct medicalData {
6          address provider_address;
7          string date;
8          string[] medicine_name;
9          uint[] quantity;
10         uint index;
11     }
12
13     struct permissionData {
14         address provider_address;
15         string date;
16         string message;
17         uint index;
18     }
19
20     struct allowedPermissions {
21         address provider_address;
22         string date;
23         uint time;
24         uint index;
25     }
26
27     mapping (address=>medicalData[]) medicalDataHistory;
28
29     mapping(address => medicalData[]) approvalQueue;
30
31     mapping(address => permissionData[]) permissionsQueue;
32
33     mapping(address => allowedPermissions[]) allowedPermissionsQueue;
```


MedicalHistoryContract.sol ✕

```

35     function getMedicalDataHistoryLength() public view returns(uint)
36     {
37         uint length = 0;
38         uint dataLength = medicalDataHistory[msg.sender].length;
39         for(uint i=0; i<dataLength; i++) {
40             if(medicalDataHistory[msg.sender][i].index == 0) {
41                 continue;
42             }
43             length++;
44         }
45         return length;
46     }
47
48     function getApprovalQueueLength() public view returns(uint) {
49         uint length = 0;
50         uint dataLength = approvalQueue[msg.sender].length;
51         for(uint i=0; i<dataLength; i++) {
52             if(approvalQueue[msg.sender][i].index == 0) {
53                 continue;
54             }
55             length++;
56         }
57         return length;
58     }
59
60     function getPermissionsQueueLength() public view returns(uint) {
61         uint length = 0;
62         uint dataLength = permissionsQueue[msg.sender].length;
63         for(uint i=0; i<dataLength; i++) {
64             if(permissionsQueue[msg.sender][i].index == 0) {
65                 continue;
66             }
67             length++;
68         }
69     }

```

MedicalHistoryContract.sol ✕

```
68     return length;
69 }
70
71 function getAllowedPermissionsQueueLength() public view returns(uint) {
72     uint length = 0;
73     uint dataLength = allowedPermissionsQueue[msg.sender].length;
74     for(uint i=0; i<dataLength; i++) {
75         if(allowedPermissionsQueue[msg.sender][i].index == 0) {
76             continue;
77         }
78         length++;
79     }
80     return length;
81 }
82
83 function getMyData() public view returns(medicalData[] memory) {
84     return medicalDataHistory[msg.sender];
85 }
86
87 function getOtherPatientData(address patient_address) public view returns(medicalData[] memory) {
88     uint flag = 0;
89     uint length = allowedPermissionsQueue[patient_address].length;
90     for(uint i=0; i<length; i++) {
91         if(allowedPermissionsQueue[patient_address][i].provider_address == msg.sender) {
92             if(allowedPermissionsQueue[patient_address][i].time > now) {
93                 flag = 1;
94             }
95             break;
96         }
97     }
98     require(flag==1);
99     return medicalDataHistory[patient_address];
100 }
101
```

B: SERVICE FOR CALLING BACKEND FUNCTIONS

medical-data.service.ts



```
1  import { Injectable } from '@angular/core';
2  import Web3 from 'web3';
3  import { Subject, Observable } from 'rxjs';
4  import { throwMatDialogContentAlreadyAttachedError } from '@angular/material';
5
6  declare global {
7    interface Window { web3: any; }
8  }
9  const abi = [
10   {
11     "constant": false,
12     "inputs": [
13       {
14         "name": "index",
15         "type": "uint256"
16       },
17       {
18         "name": "time",
19         "type": "uint256"
20       },
21       {
22         "name": "date",
23         "type": "string"
24       }
25     ],
26     "name": "grantPermission",
27     "outputs": [],
28     "payable": false,
29     "stateMutability": "nonpayable",
30     "type": "function"
31   },
32   {
33     "constant": true,
34     "inputs": []
```

```

Edit Selection View Go Debug Terminal Help • medical-data.service.ts - healthcare - Visual
TS medical-data.service.ts •
356 const address = '0x1493c54050a4ab7c0f07f4bb45b8c95636afE337';
357
358 @Injectable({
359   providedIn: 'root'
360 })
361 export class MedicalDataService {
362   web3: any;
363   MedicalContract: any;
364   patient_address: string;
365   options: {};
366   medicalHistoryDataSubject = new Subject<any>();
367   approvalQueueDataSubject = new Subject<any>();
368   permissionsQueueDataSubject = new Subject<any>();
369   constructor() {
370     this.web3 = new Web3('ws://localhost:8545');
371     this.MedicalContract = new this.web3.eth.Contract(abi, address);
372     this.patient_address = "0x9B59f65a147124E2e6E9edB6492302C1B84559bB";
373     this.options = { from: this.patient_address, gas: 3000000 };
374   }
375
376   getMyData(): Observable<any> {
377     this.MedicalContract.methods.getMyData().call().then((result) => {
378       this.medicalHistoryDataSubject.next(result);
379     });
380     return this.medicalHistoryDataSubject.asObservable();
381   }
382
383   getApprovalQueueData(): Observable<any> {
384     this.MedicalContract.methods.getApprovalQueueData().call().then((result) => {
385       this.approvalQueueDataSubject.next(result);
386     });
387     return this.approvalQueueDataSubject.asObservable();
388   }

```

C: FRONT END CODE

Medical-records-page.component.ts

```

Edit Selection View Go Debug Terminal Help medical-records-page.component.t
TS medical-records-page.component.ts ✕
1  import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
2  import { MatSidenav } from '@angular/material';
3
4  @Component({
5    selector: 'app-medical-records-page',
6    templateUrl: './medical-records-page.component.html',
7    styleUrls: ['./medical-records-page.component.scss']
8  })
9  export class MedicalRecordsPageComponent implements OnInit {
10
11    @ViewChild('sideNav') sideNav: MatSidenav;
12    sideNavOptions: {}[];
13    title: string;
14    notifications: number[];
15
16    constructor() { }
17
18    ngOnInit() {
19      this.initializeData();
20    }
21
22    initializeData() {
23      this.sideNavOptions = [
24        {value: 'Medical History', route: 'medical-history'},
25        {value: 'Pending Medical Approvals', route: 'medical-approvals'},
26        {value: 'Manage Medical Permissions', route: 'medical-permissions'},
27        {value: 'My Doctors', route: 'my-doctors'}
28      ];
29      this.title = this.sideNavOptions[0]['value'];
30    }
31
32    changeData(title) {
33      this.sideNav.close();
34      this.title = title;

```

Medical-records-page.component.html

```
Edit Selection View Go Debug Terminal Help • medical-records-page.component.html - healthcare - Visual Studio Code

<> medical-records-page.component.html ●

1  <mat-sidenav-container>
2    <mat-sidenav #sideNav>
3      <div *ngFor="let option of sideNavOptions" class="sideNav" routerLink="{{option.route}}"
4        routerLinkActive="sideNavActive" (click)="changeData(option.value)">
5        | {{option.value}}
6      </div>
7    </mat-sidenav>
8    <mat-sidenav-content>
9      <div>
10         <div class="heading" fxLayout="row" fxLayoutGap="20px" fxLayoutAlign="start center">
11           
12           <p class="heading-text">{{title}}</p>
13         </div>
14         <router-outlet></router-outlet>
15       </div>
16     </mat-sidenav-content>
17 </mat-sidenav-container>
```

PAPER PUBLICATION

Paper publication has not yet started. Writing part is in progress.