CRYPTOGRAPHY TERM PROJECT

BLOCKCHAIN TO PRESERVE INTEGRITY OF MEDICAL RECORDS

TEAM MEMBERS:

2019A7PS0048H ROHAN RAO
2019A3PS0205H PAVAN SHYAMENDRA
2019A7PS0036H SAI SANDEEP

PROBLEM STATEMENT:

When users submit their medical health records to some hospitals or some private organisations,there is a chance that some data might be tampered.Since the medical health related data is very sensitive,the integrity and privacy of the data must be preserved.Also to verify the user the verifier might need some other sensitive information like ssn id or pan card number ,etc of the user,so we need to ensure that we authenticate the user properly without knowing those ssnid or any sensitive data of the user using zero knowledge proofs.

Blockchain introduction

A blockchain is a digital and distributed ledger of transactions, recorded and replicated in real-time across a network of computers or nodes. Every transaction must be cryptographically validated via a consensus mechanism executed by the nodes before being permanently added as a new “block” at the end of the “chain.” There is no need for a central authority to approve the transaction, which is why blockchain is sometimes referred to as a peer-to-peer trustless mechanism.

Blockchain can be thought of as a linked list with each node containing multiple transactions. Each transaction has a hash that depends on the previous transactions hash as well. So we can see that the order of transactions is important. If we were to change one transaction somewhere, it would have a ripple effect and change the hash of all subsequent transactions. This is one of the reasons why blockchain is a powerful medium for storing transactions.

(cntd)

The placing of a transaction in a block is called a successful conclusion to a proof of work challenge, and is carried out by special nodes called miners. Proof of Work is a system that requires some work from the service requester, usually meaning processing time by a computer. Producing a proof of work is a random process with low probability, so normally a lot of trial and error is required for a valid proof of work to be generated. When it comes to Bitcoins, hash is what serves as a proof of work. Miners on a Blockchain are nodes that produce blocks by solving proof of work problems. If a miner produces a block that is approved by an electronic consensus of nodes then the miner is rewarded with coins. This essentially is the crux of blockchain. Proof of Work is what is keeping all transactions on the blockchain secure and protecting it from malicious attempts to alter these transactions.

How blockchain solves this problem?

We can store the medical health records of the user at multiple nodes (ie owned by a set of hospitals or organisations).

So we store the data in a distributed environment and data addition is governed by the consensus mechanism implemented by the designer.

We store the medical health record of any user as a transaction.

Then the user sends this transaction to transaction pool from where the validators will verify the transaction and add this to the block which that the validator proposes.

(CNTD)

Then every node tries to propose a block but since this could lead to race condition the consensus mechanism comes for the protection .

Only the first node to compute the proof of work will propose the block .

Every other node will validate this block and update the local blockchains stored in them.

And whenever there is forking of blockchain then a rule known as longest chain rule is applied to know the correct chain of blockchain

How and where do we use zkp?

So when the user submits their data as a transaction,since there could also be a possibility of malicious user submitting random and unwanted data which is not desired,we would require the user to authenticate before inserting any record.

But to verify we need some ssn id or pan card number or debit card number for payment processing etc,which the user wouldn’t want to share with the verifier.

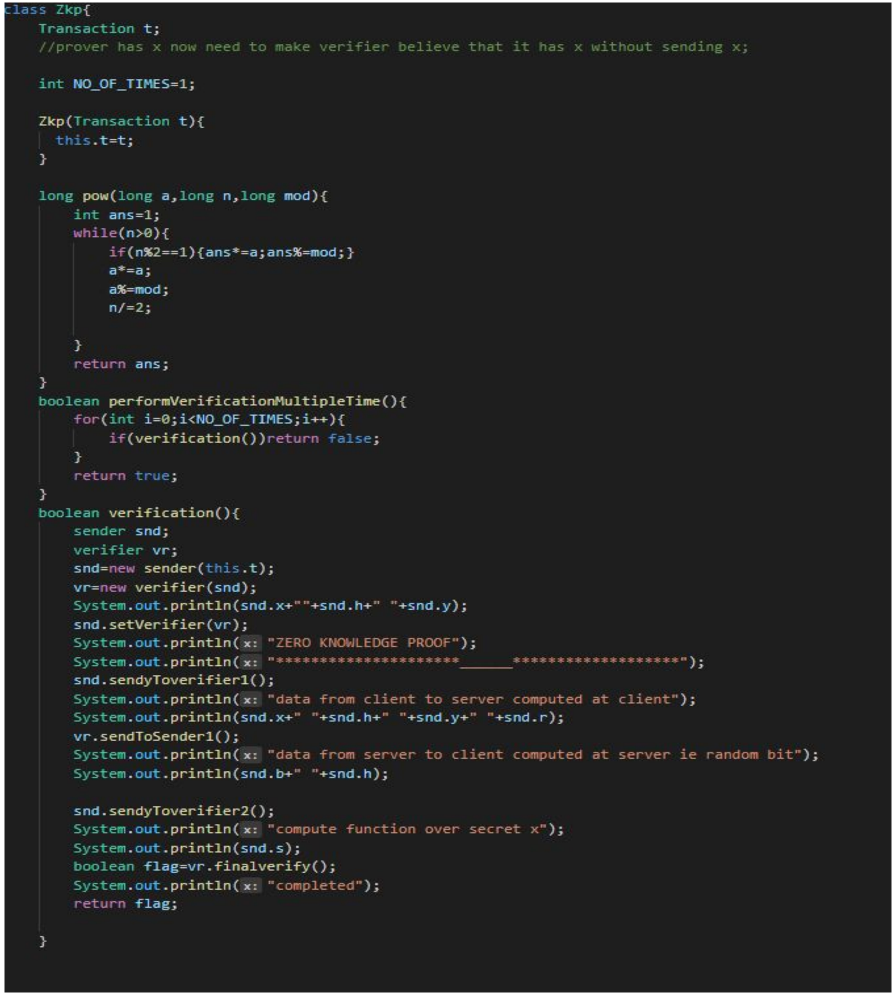
So here we apply zero knowledge proof to verify the user and to prove the authenticity of the user.

Zkp in our code

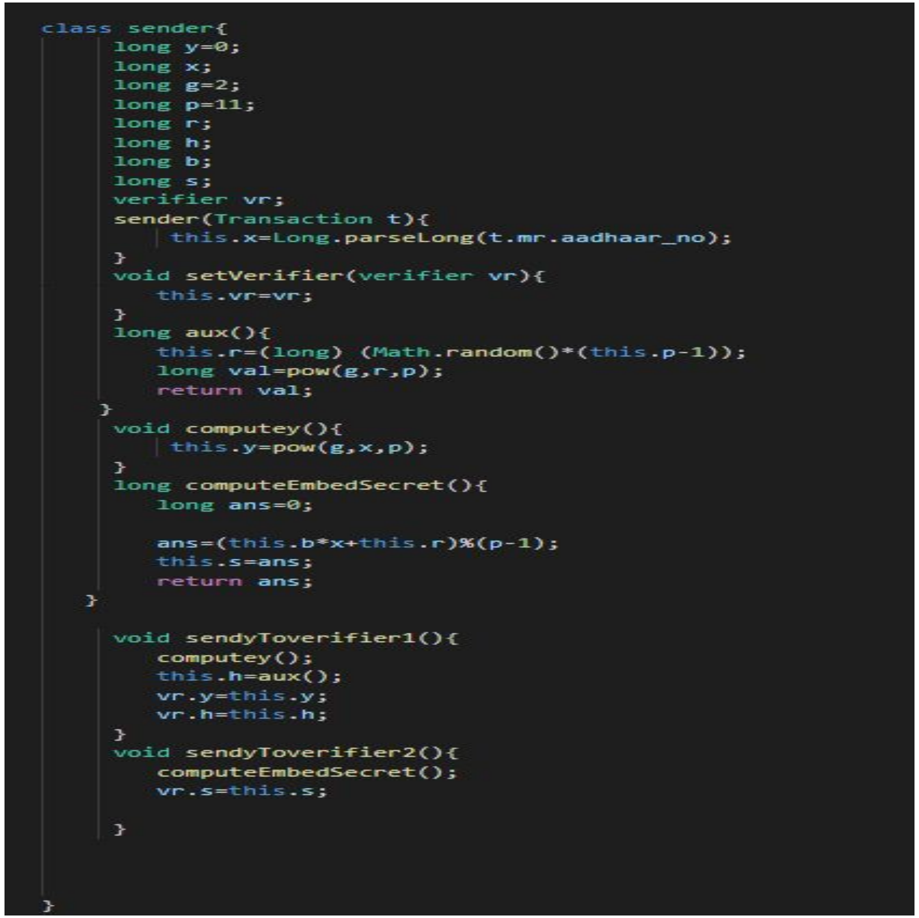
In our project, zero-knowledge protocol is used for verifying the individual transactions (medical records).

1. Client chooses a random number $0 \leq r < p-1$ and sends it to server as $h = g^\wedge r \bmod (p)$
2. server receives h and sends back a random bit b (could be 0/1).
3. client sends $s = (r + bx) \bmod (p-1)$ to Bob.
4. server computes $g^\wedge s \bmod (p)$ which should equal $hy^\wedge b \bmod (p)$.

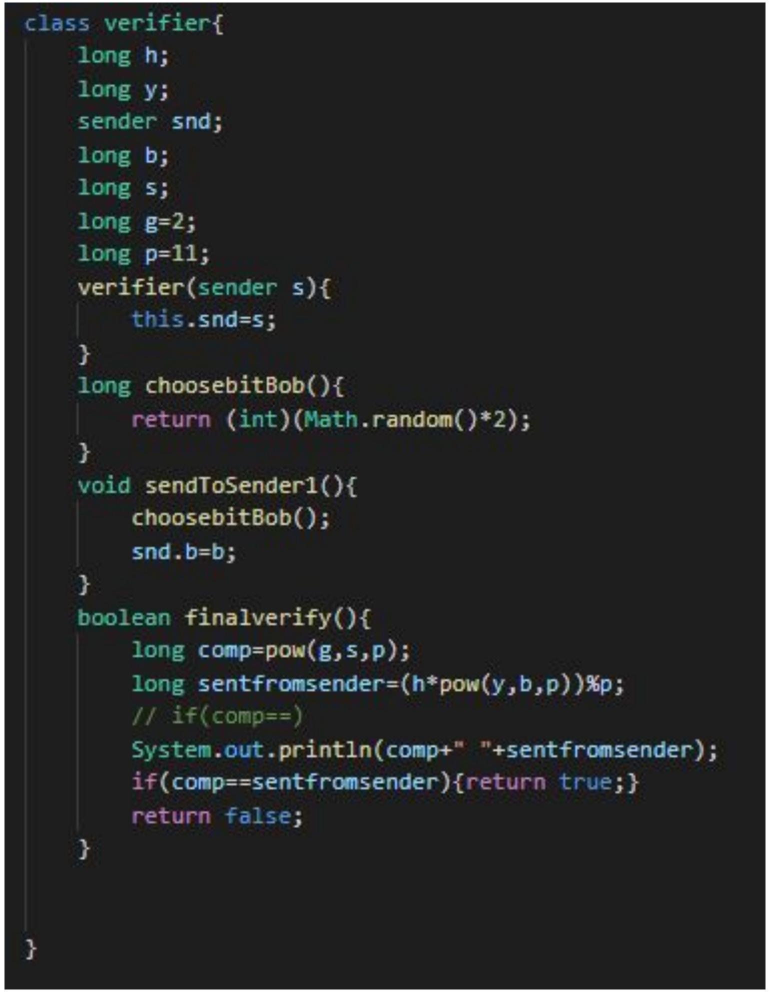
Zkp implementation:



```java
class Zkp{
    Transaction t;
    //prover has x now need to make verifier believe that it has x without sending x;

    int NO_OF_TIMES=1;

    Zkp(Transaction t){
      this.t=t;
    }

    long pow(long a,long n,long mod){
        int ans=1;
        while(n>0){
            if(n%2==1){ans*=a;ans%=mod;}
            a*=a;
            a%=mod;
            n/=2;

        }
        return ans;
    }
    boolean performVerificationMultipleTime(){
        for(int i=0;i<NO_OF_TIMES;i++){
            if(verification())return false;
        }
        return true;
    }
    boolean verification(){
        sender snd;
        verifier vr;
        snd=new sender(this.t);
        vr=new verifier(snd);
        System.out.println(snd.x+""+snd.h+" "+snd.y);
        snd.setVerifier(vr);
        System.out.println(x: "ZERO KNOWLEDGE PROOF");
        System.out.println(x: "**********************______*******************");
        snd.sendyToverifier1();
        System.out.println(x: "data from client to server computed at client");
        System.out.println(snd.x+" "+snd.h+" "+snd.y+" "+snd.r);
        vr.sendToSender1();
        System.out.println(x: "data from server to client computed at server ie random bit");
        System.out.println(snd.b+" "+snd.h);

        snd.sendyToverifier2();
        System.out.println(x: "compute function over secret x");
        System.out.println(snd.s);
        boolean flag=vr.finalverify();
        System.out.println(x: "completed");
        return flag;

    }
```



```
class sender{
        long y=0;
        long x;
        long g=2;
        long p=11;
        long r;
        long h;
        long b;
        long s;
        verifier vr;
        sender(Transaction t){
            this.x=Long.parseLong(t.mr.aadhaar_no);
        }
        void setVerifier(verifier vr){
            this.vr=vr;
        }
        long aux(){
            this.r=(long) (Math.random()*(this.p-1));
            long val=pow(g,r,p);
            return val;
       }
        void computey(){
            this.y=pow(g,x,p);
        }
        long computeEmbedSecret(){
            long ans=0;

            ans=(this.b*x+this.r)%(p-1);
            this.s=ans;
            return ans;
      }

        void sendyToverifier1(){
            computey();
            this.h=aux();
            vr.y=this.y;
            vr.h=this.h;
        }
        void sendyToverifier2(){
            computeEmbedSecret();
            vr.s=this.s;

        }


}
```

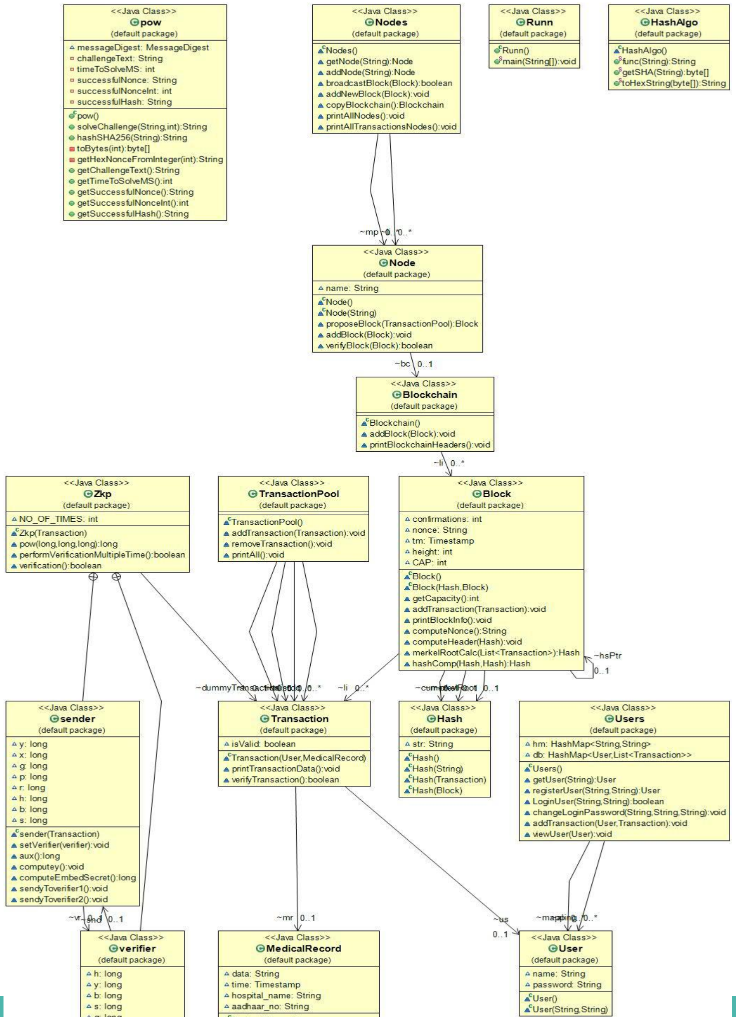


```java
class verifier{
    long h;
    long y;
    sender snd;
    long b;
    long s;
    long g=2;
    long p=11;
    verifier(sender s){
        this.snd=s;
    }
    long choosebitBob(){
        return (int)(Math.random()*2);
    }
    void sendToSender1(){
        choosebitBob();
        snd.b=b;
    }
    boolean finalverify(){
        long comp=pow(g,s,p);
        long sentfromsender=(h*pow(y,b,p))%p;
        // if(comp==)
        System.out.println(comp+" "+sentfromsender);
        if(comp==sentfromsender){return true;}
        return false;
    }


}
```

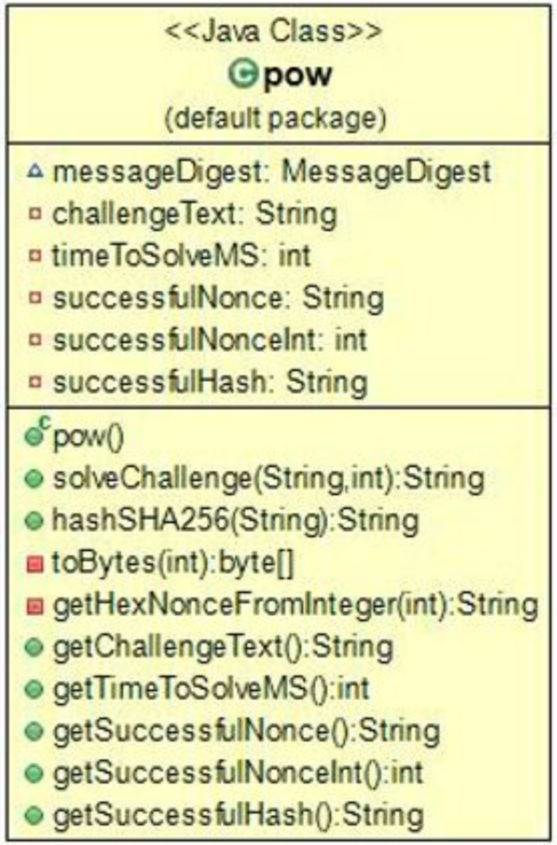
UML Diagrams

Link for details

https://ibb.co/rkNbvSv

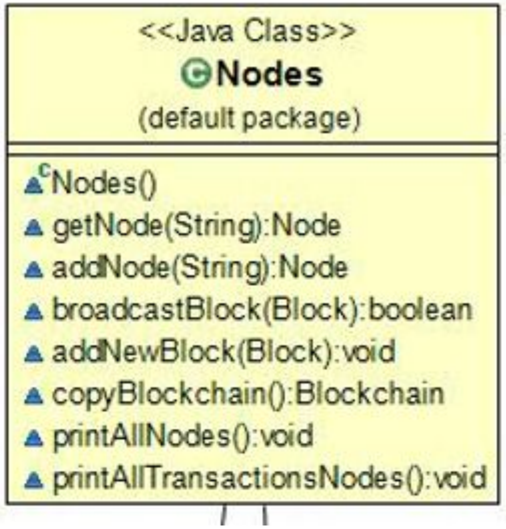


<<Java Class>> pow (default package)
messageDigest: MessageDigest
challengeText: String
timeToSolveMS: int
successfulNonce: String
successfulNonceInt: int
successfulHash: String
pow()
solveChallenge(String,int):String
hashSHA256(String):String
toBytes(int):byte[]
getHexNonceFromInteger(int):String
getChallengeText():String
getTimeToSolveMS():int
getSuccessfulNonce():String
getSuccessfulNonceInt():int
getSuccessfulHash():String
<<Java Class>> Nodes (default package)
Nodes()
getNode(String):Node
addNode(String):Node
broadcastBlock(Block):boolean
addNewBlock(Block):void
copyBlockchain():Blockchain
printAllNodes():void
printAllTransactionsNodes():void
<<Java Class>> Runn (default package)
Runn()
main(String[]):void
<<Java Class>> HashAlgo (default package)
HashAlgo()
func(String):String
getSHA(String):byte[]
toHexString(byte[]):String
~mp 0..* 0..*
<<Java Class>> Node (default package)
name: String
Node()
Node(String)
proposeBlock(TransactionPool):Block
addBlock(Block):void
verifyBlock(Block):boolean
~bc 0..1
<<Java Class>> Blockchain (default package)
Blockchain()
addBlock(Block):void
printBlockchainHeaders():void
~li 0..*
<<Java Class>> Zkp (default package)
NO_OF_TIMES: int
Zkp(Transaction)
pow(long,long,long):long
performVerificationMultipleTime():boolean
verification():boolean
<<Java Class>> TransactionPool (default package)
TransactionPool()
addTransaction(Transaction):void
removeTransaction():void
printAll():void
<<Java Class>> Block (default package)
confirmations: int
nonce: String
tm: Timestamp
height: int
CAP: int
Block()
Block(Hash,Block)
getCapacity():int
addTransaction(Transaction):void
printBlockInfo():void
computeNonce():String
computeHeader(Hash):void
merkelRootCalc(List<Transaction>):Hash
hashComp(Hash,Hash):Hash
~hsPtr 0..1
~dummyTransaction 0..1 0..*
~li 0..*
~currentRoot 0..1
<<Java Class>> sender (default package)
y: long
x: long
g: long
p: long
r: long
h: long
b: long
s: long
sender(Transaction)
setVerifier(verifier):void
aux():long
computey():void
computeEmbedSecret():long
sendyToverifier1():void
sendyToverifier2():void
<<Java Class>> Transaction (default package)
isValid: boolean
Transaction(User,MedicalRecord)
printTransactionData():void
verifyTransaction():boolean
<<Java Class>> Hash (default package)
str: String
Hash()
Hash(String)
Hash(Transaction)
Hash(Block)
<<Java Class>> Users (default package)
hm: HashMap<String,String>
db: HashMap<User,List<Transaction>>
Users()
getUser(String):User
registerUser(String,String):User
LoginUser(String,String):boolean
changeLoginPassword(String,String,String):void
addTransaction(User,Transaction):void
viewUser(User):void
~vr ~snd 0..1 0..1
~mr 0..1
~us 0..1
~mapping 0..*
<<Java Class>> verifier (default package)
h: long
y: long
b: long
s: long
<<Java Class>> MedicalRecord (default package)
data: String
time: Timestamp
hospital_name: String
aadhaar_no: String
<<Java Class>> User (default package)
name: String
password: String
User()
User(String,String)

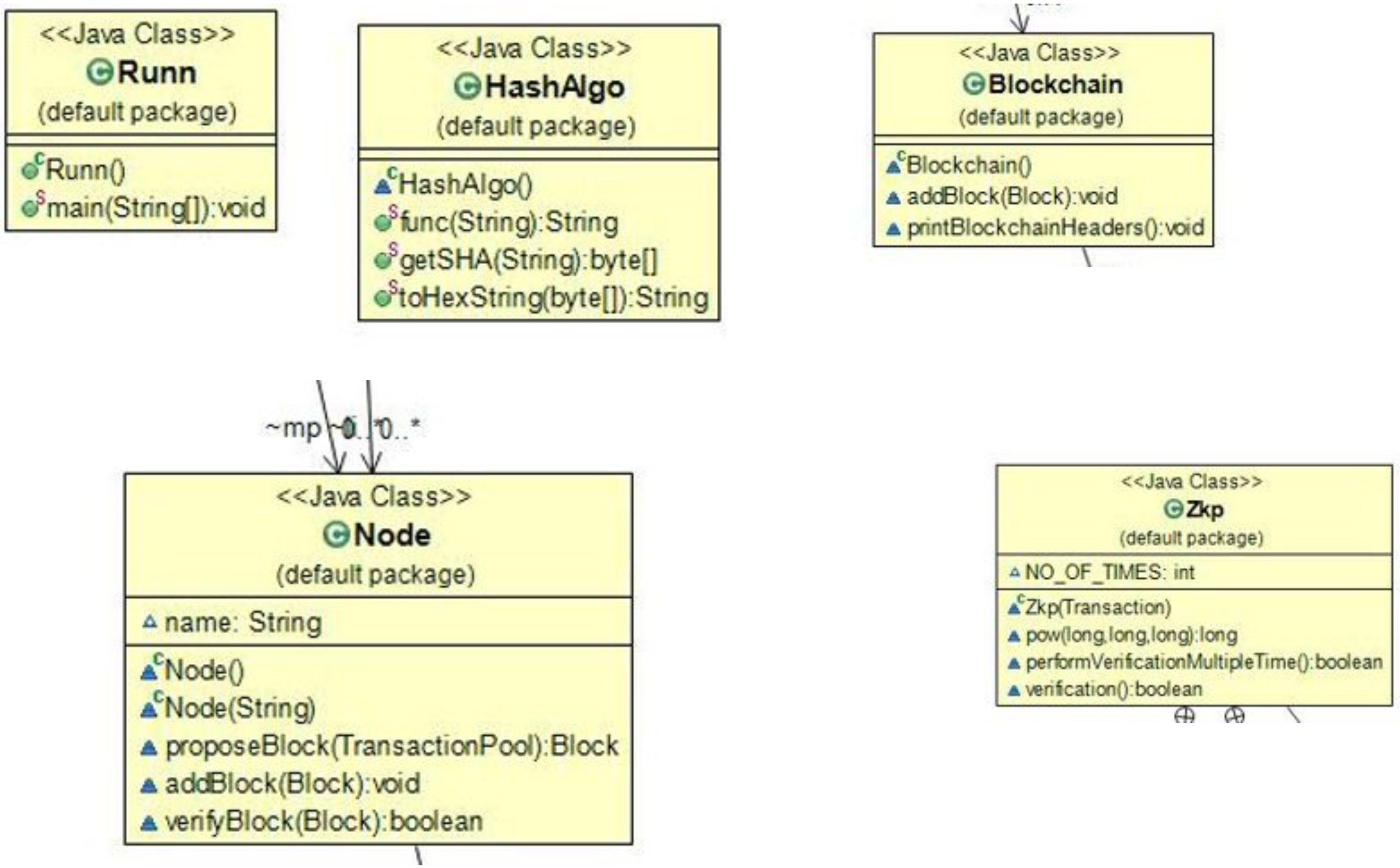

<<Java Class>>
pow
(default package)

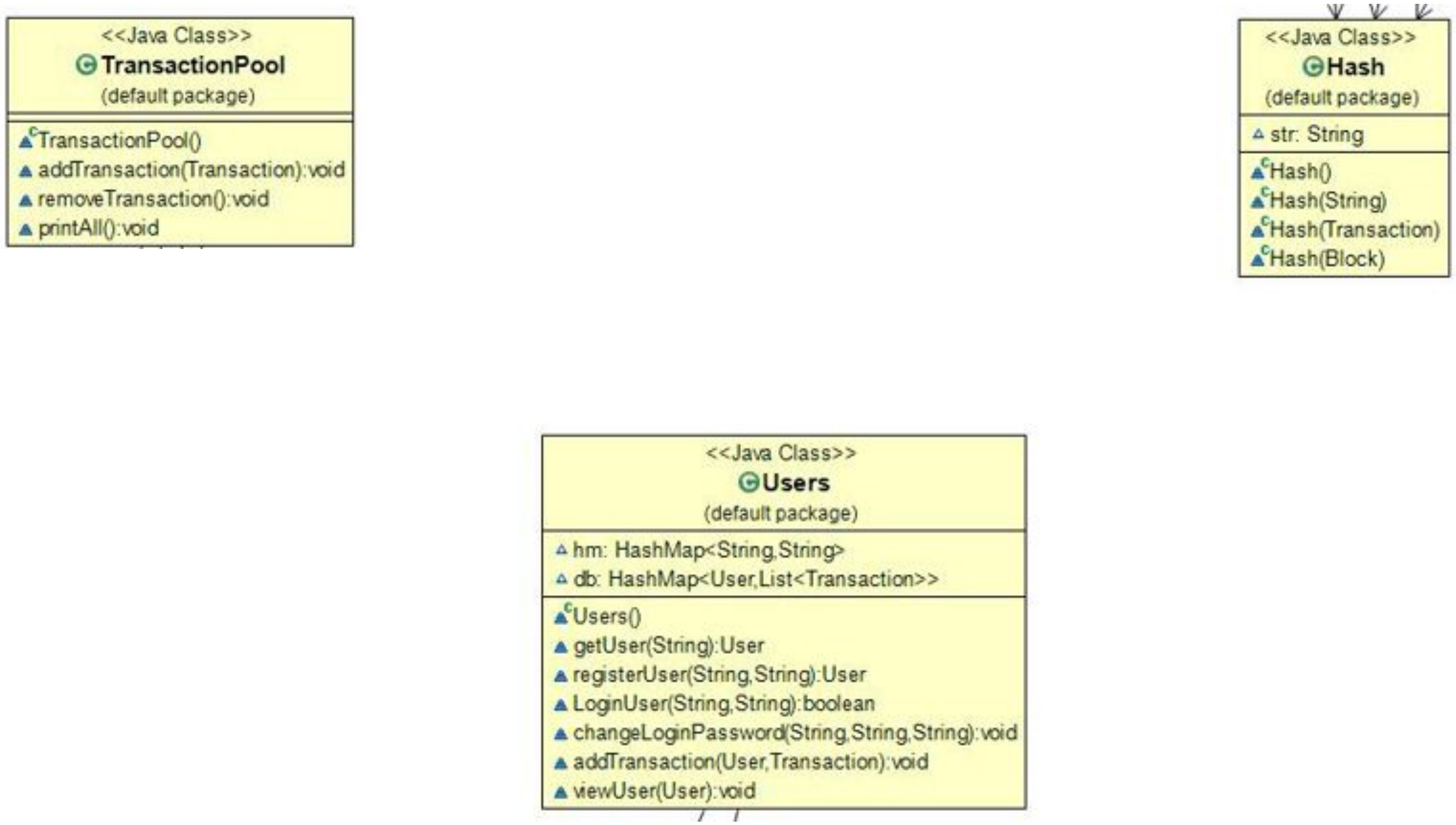
- messageDigest: MessageDigest
- challengeText: String
- timeToSolveMS: int
- successfulNonce: String
- successfulNonceInt: int
- successfulHash: String

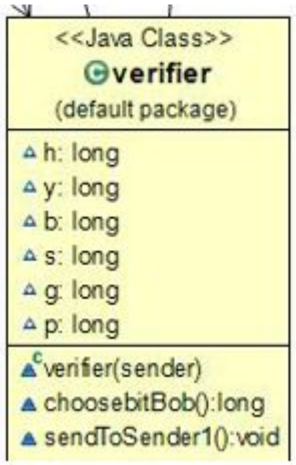
- pow()
- solveChallenge(String,int):String
- hashSHA256(String):String
- toBytes(int):byte[]
- getHexNonceFromInteger(int):String
- getChallengeText():String
- getTimeToSolveMS():int
- getSuccessfulNonce():String
- getSuccessfulNonceInt():int
- getSuccessfulHash():String

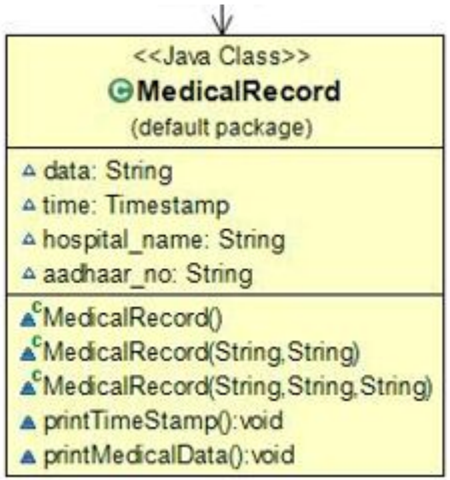

<<Java Class>>
Nodes
(default package)

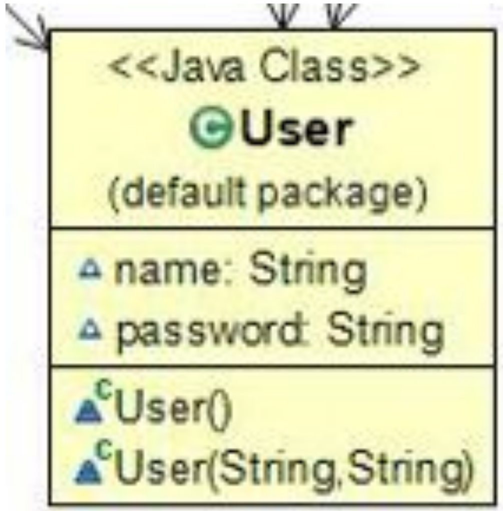
- Nodes()
- getNode(String):Node
- addNode(String):Node
- broadcastBlock(Block):boolean
- addNewBlock(Block):void
- copyBlockchain():Blockchain
- printAllNodes():void
- printAllTransactionsNodes():void


<<Java Class>>
Runn
(default package)
Runn()
main(String[]):void
<<Java Class>>
HashAlgo
(default package)
HashAlgo()
func(String):String
getSHA(String):byte[]
toHexString(byte[]):String
<<Java Class>>
Blockchain
(default package)
Blockchain()
addBlock(Block):void
printBlockchainHeaders():void
~mp ~0..*0..*
<<Java Class>>
Node
(default package)
name: String
Node()
Node(String)
proposeBlock(TransactionPool):Block
addBlock(Block):void
verifyBlock(Block):boolean
<<Java Class>>
Zkp
(default package)
NO_OF_TIMES: int
Zkp(Transaction)
pow(long,long,long):long
performVerificationMultipleTime():boolean
verification():boolean

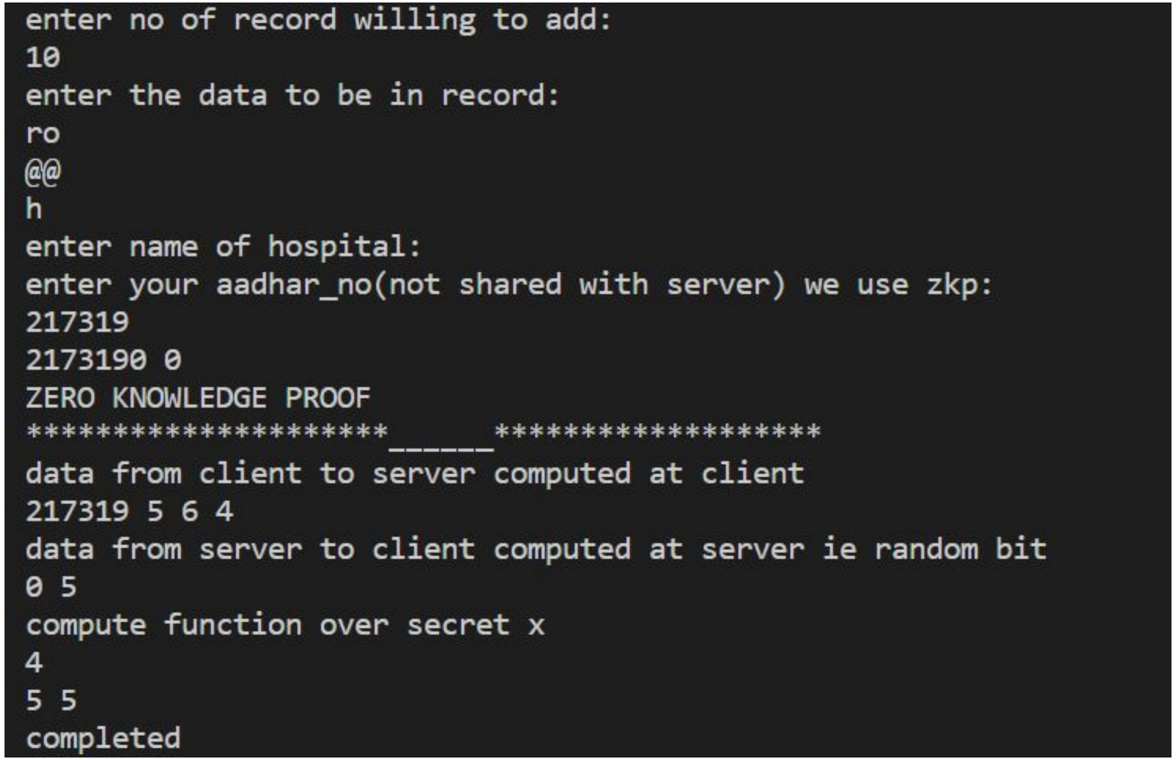

<<Java Class>>
TransactionPool
(default package)
TransactionPool()
addTransaction(Transaction):void
removeTransaction():void
printAll():void
<<Java Class>>
Hash
(default package)
str: String
Hash()
Hash(String)
Hash(Transaction)
Hash(Block)
<<Java Class>>
Users
(default package)
hm: HashMap<String,String>
db: HashMap<User,List<Transaction>>
Users()
getUser(String):User
registerUser(String,String):User
LoginUser(String,String):boolean
changeLoginPassword(String,String,String):void
addTransaction(User,Transaction):void
viewUser(User):void


<<Java Class>>
verifier
(default package)
h: long
y: long
b: long
s: long
g: long
p: long
verifer(sender)
choosebitBob():long
sendToSender1():void

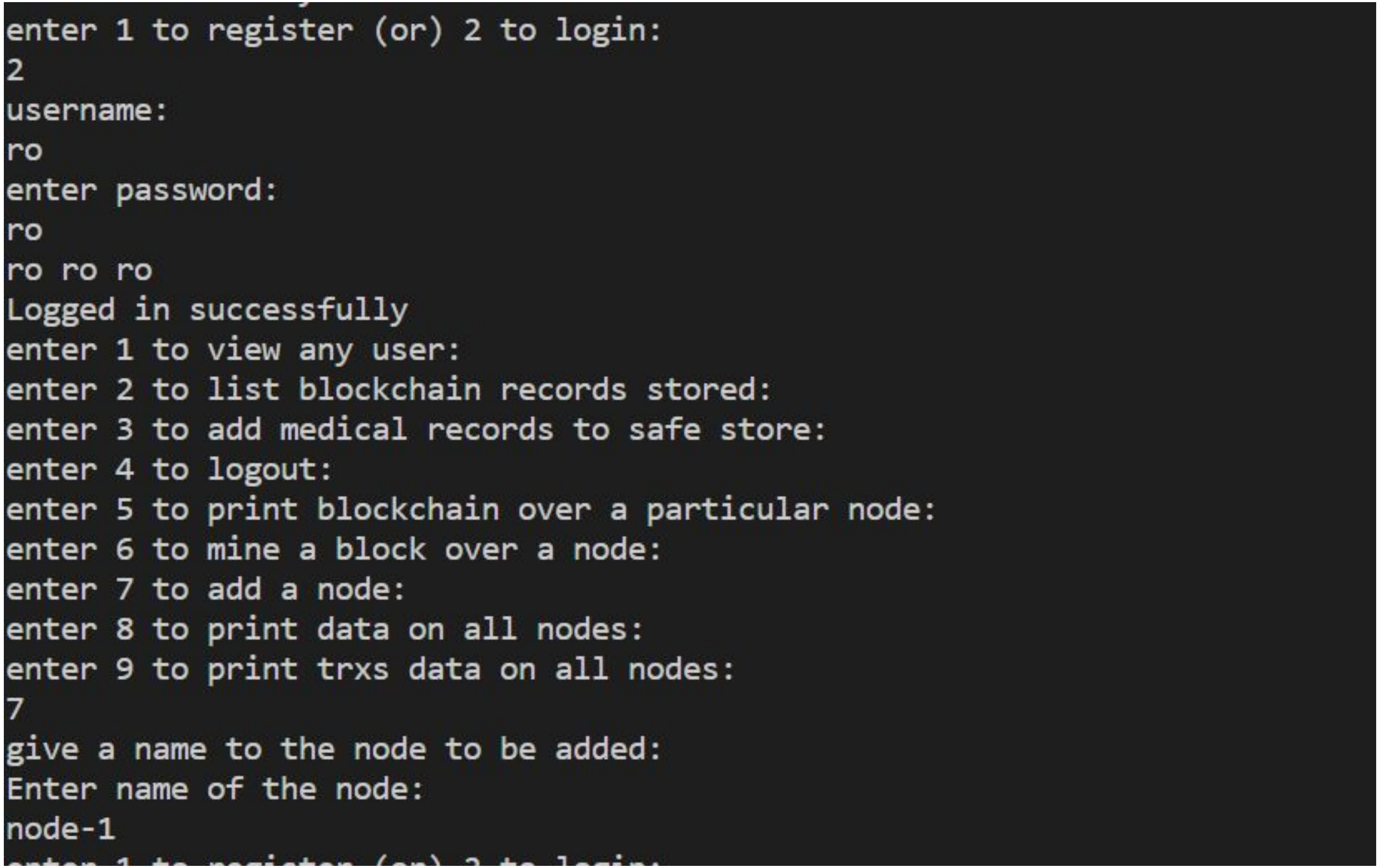

<<Java Class>>
MedicalRecord
(default package)
data: String
time: Timestamp
hospital_name: String
aadhaar_no: String
MedicalRecord()
MedicalRecord(String,String)
MedicalRecord(String,String,String)
printTimeStamp():void
printMedicalData():void


<<Java Class>>
User
(default package)
name: String
password: String
User()
User(String,String)

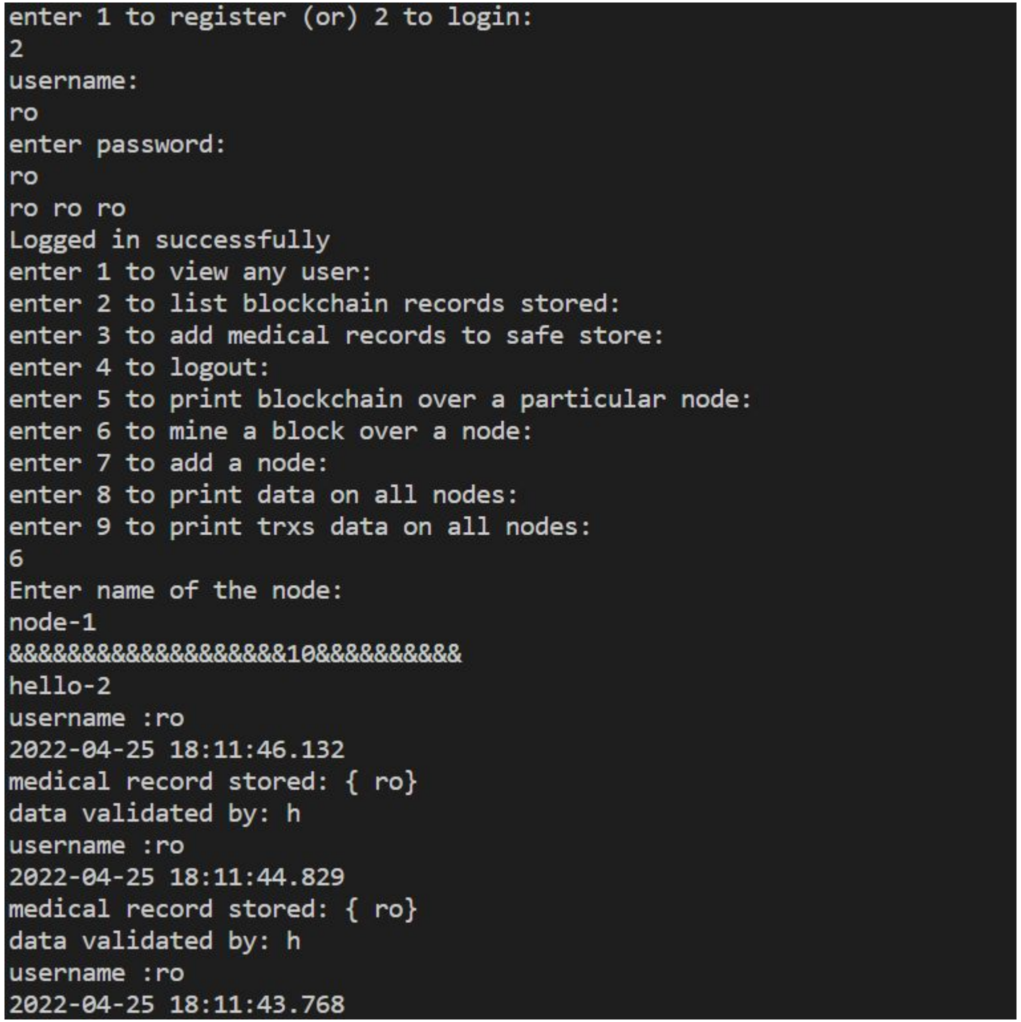
ScreenShots

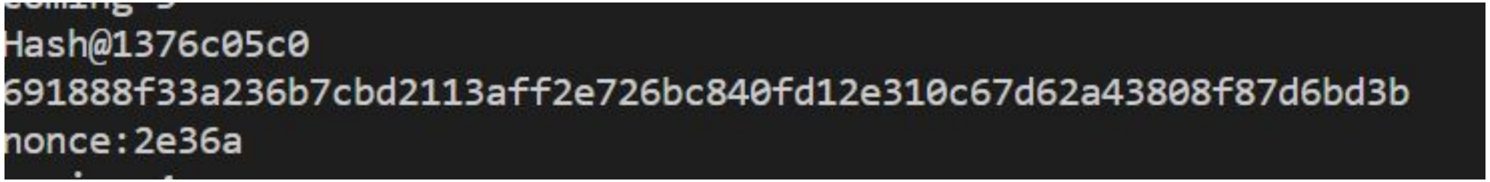


```
enter no of record willing to add:
10
enter the data to be in record:
ro
@@
h
enter name of hospital:
enter your aadhar_no(not shared with server) we use zkp:
217319
2173190 0
ZERO KNOWLEDGE PROOF
*********************______******************
data from client to server computed at client
217319 5 6 4
data from server to client computed at server ie random bit
0 5
compute function over secret x
4
5 5
completed
```

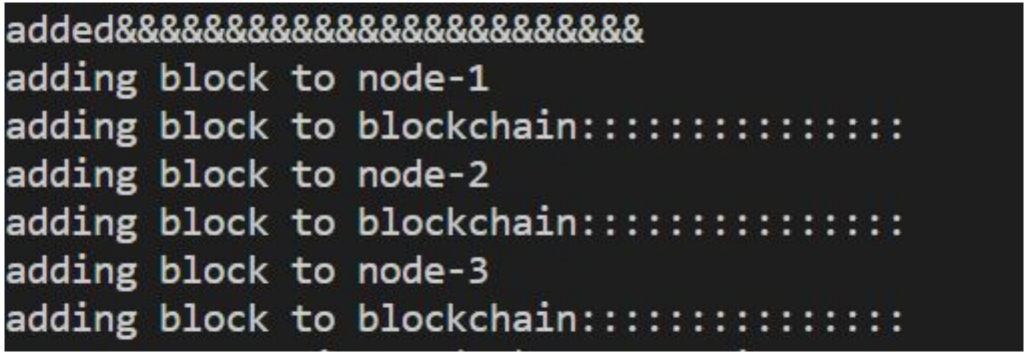


```
enter 1 to register (or) 2 to login:
2
username:
ro
enter password:
ro
ro ro ro
Logged in successfully
enter 1 to view any user:
enter 2 to list blockchain records stored:
enter 3 to add medical records to safe store:
enter 4 to logout:
enter 5 to print blockchain over a particular node:
enter 6 to mine a block over a node:
enter 7 to add a node:
enter 8 to print data on all nodes:
enter 9 to print trxs data on all nodes:
7
give a name to the node to be added:
Enter name of the node:
node-1
```

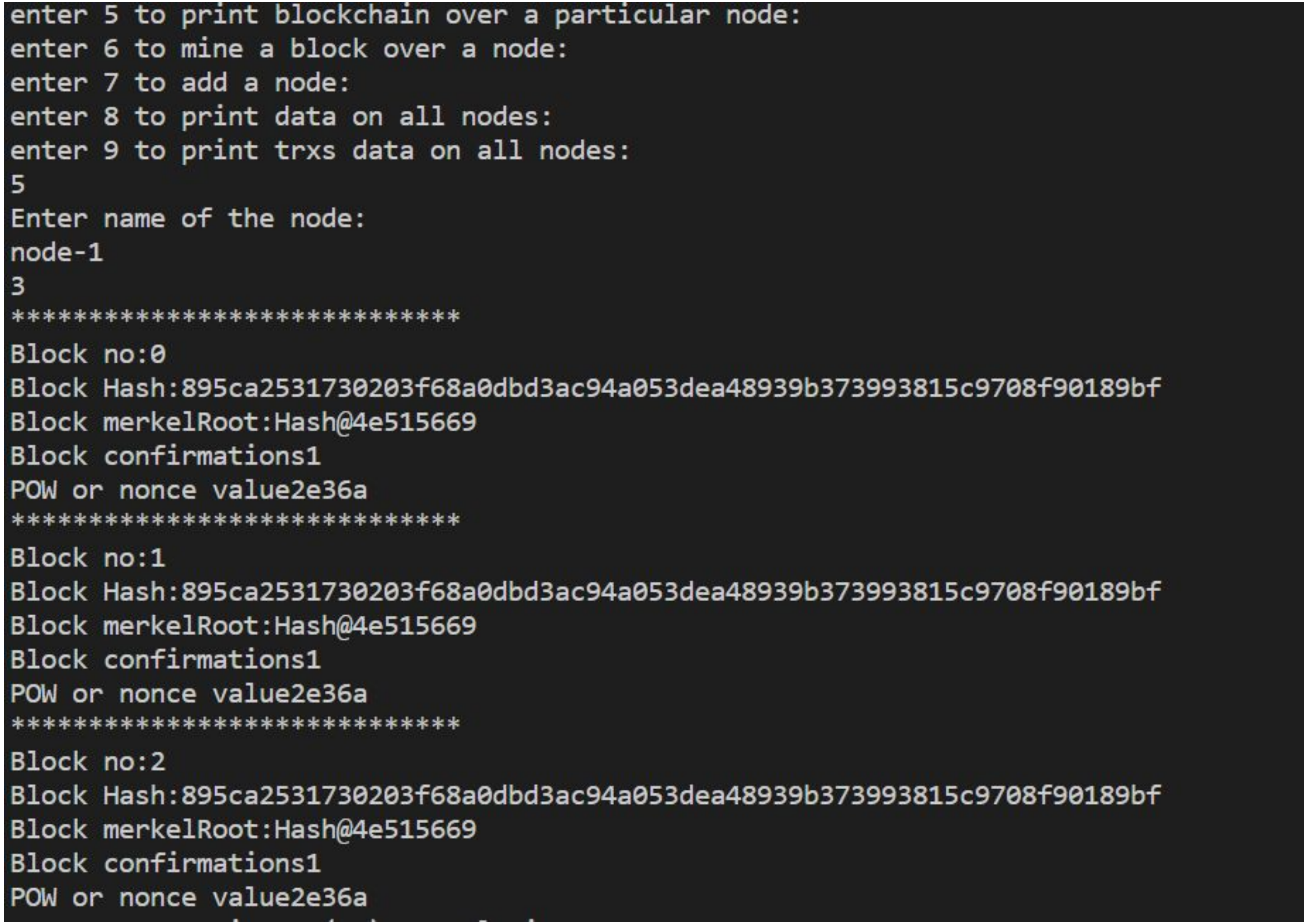

```
enter 1 to register (or) 2 to login:
2
username:
ro
enter password:
ro
ro ro ro
Logged in successfully
enter 1 to view any user:
enter 2 to list blockchain records stored:
enter 3 to add medical records to safe store:
enter 4 to logout:
enter 5 to print blockchain over a particular node:
enter 6 to mine a block over a node:
enter 7 to add a node:
enter 8 to print data on all nodes:
enter 9 to print trxs data on all nodes:
6
Enter name of the node:
node-1
&&&&&&&&&&&&&&&&&&&10&&&&&&&&&&&
hello-2
username :ro
2022-04-25 18:11:46.132
medical record stored: { ro}
data validated by: h
username :ro
2022-04-25 18:11:44.829
medical record stored: { ro}
data validated by: h
username :ro
2022-04-25 18:11:43.768
```



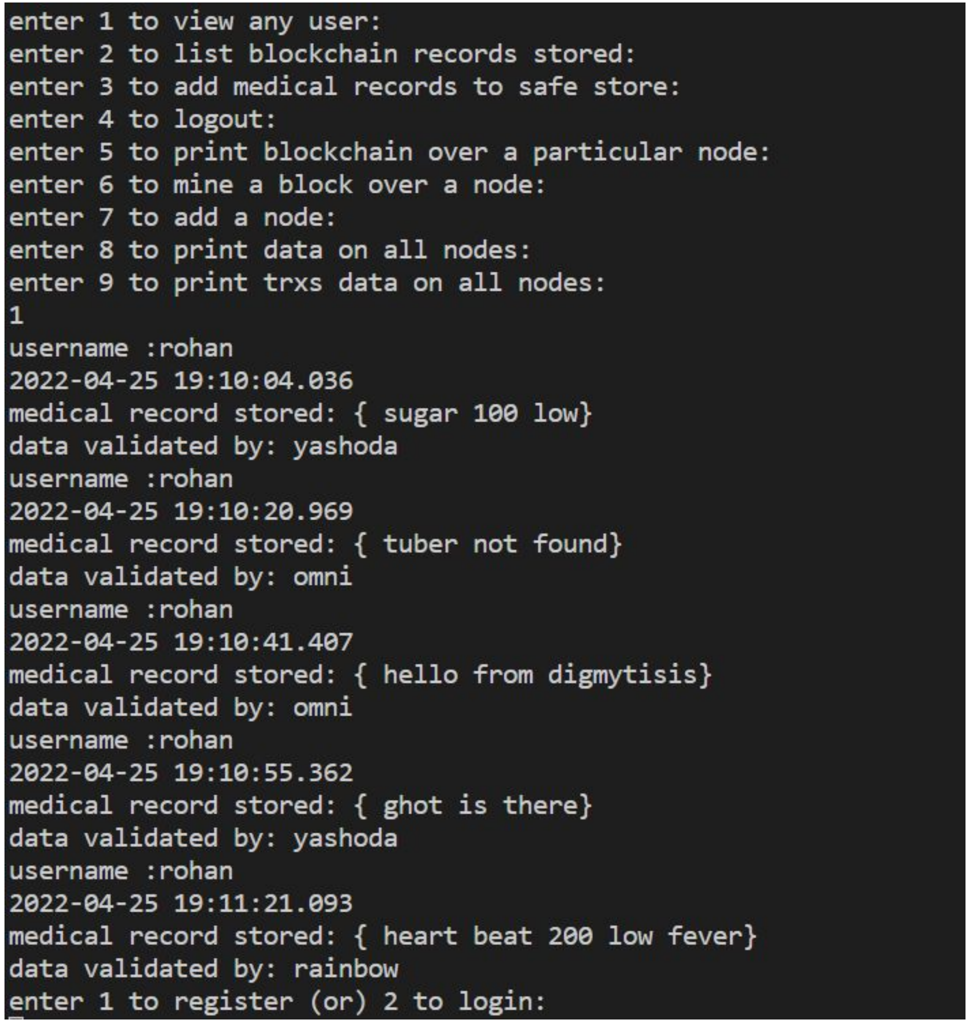
```
Hash@1376c05c0
691888f33a236b7cbd2113aff2e726bc840fd12e310c67d62a43808f87d6bd3b
nonce:2e36a
```



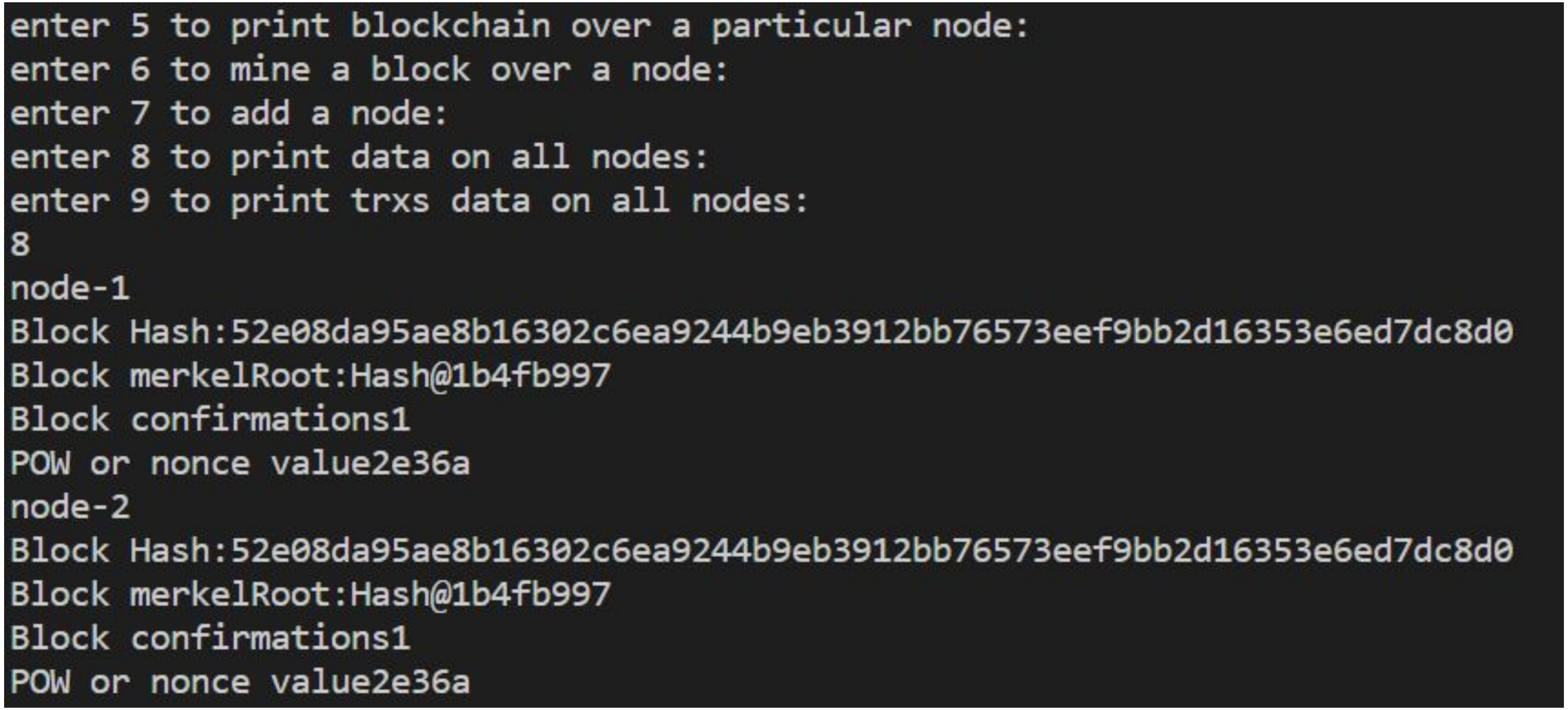
```
added&&&&&&&&&&&&&&&&&&&&&&&
adding block to node-1
adding block to blockchain:::::::::::::::
adding block to node-2
adding block to blockchain:::::::::::::::
adding block to node-3
adding block to blockchain:::::::::::::::
```



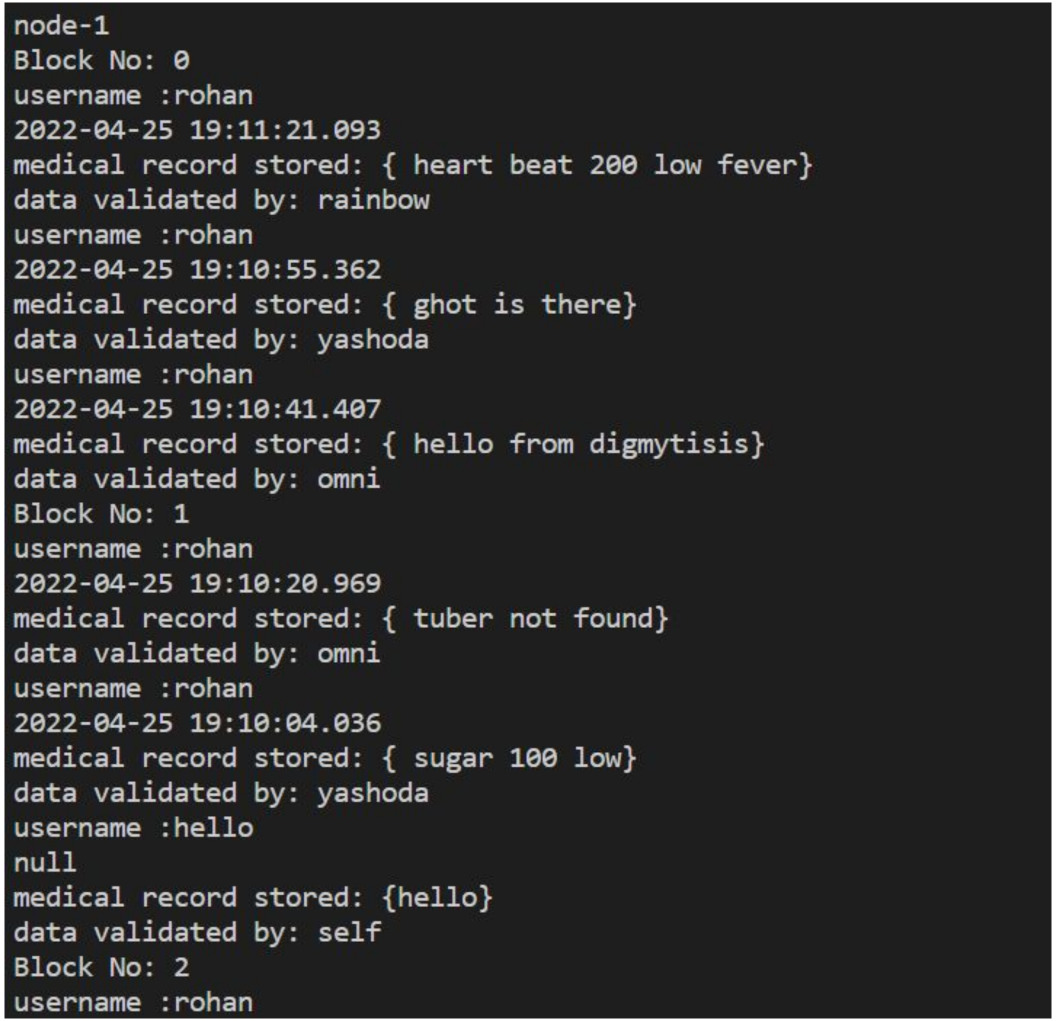
```
enter 5 to print blockchain over a particular node:
enter 6 to mine a block over a node:
enter 7 to add a node:
enter 8 to print data on all nodes:
enter 9 to print trxs data on all nodes:
5
Enter name of the node:
node-1
3
*****************************
Block no:0
Block Hash:895ca2531730203f68a0dbd3ac94a053dea48939b373993815c9708f90189bf
Block merkelRoot:Hash@4e515669
Block confirmations1
POW or nonce value2e36a
*****************************
Block no:1
Block Hash:895ca2531730203f68a0dbd3ac94a053dea48939b373993815c9708f90189bf
Block merkelRoot:Hash@4e515669
Block confirmations1
POW or nonce value2e36a
*****************************
Block no:2
Block Hash:895ca2531730203f68a0dbd3ac94a053dea48939b373993815c9708f90189bf
Block merkelRoot:Hash@4e515669
Block confirmations1
POW or nonce value2e36a
```



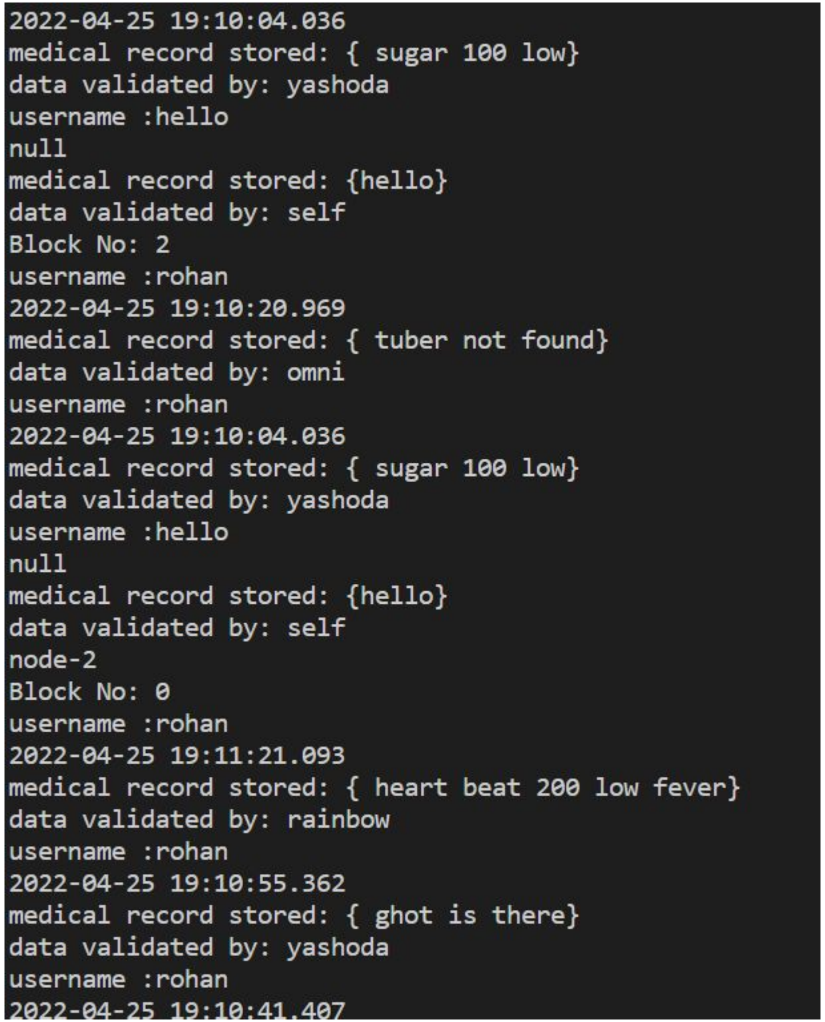
```
enter 1 to view any user:
enter 2 to list blockchain records stored:
enter 3 to add medical records to safe store:
enter 4 to logout:
enter 5 to print blockchain over a particular node:
enter 6 to mine a block over a node:
enter 7 to add a node:
enter 8 to print data on all nodes:
enter 9 to print trxs data on all nodes:
1
username :rohan
2022-04-25 19:10:04.036
medical record stored: { sugar 100 low}
data validated by: yashoda
username :rohan
2022-04-25 19:10:20.969
medical record stored: { tuber not found}
data validated by: omni
username :rohan
2022-04-25 19:10:41.407
medical record stored: { hello from digmytisis}
data validated by: omni
username :rohan
2022-04-25 19:10:55.362
medical record stored: { ghot is there}
data validated by: yashoda
username :rohan
2022-04-25 19:11:21.093
medical record stored: { heart beat 200 low fever}
data validated by: rainbow
enter 1 to register (or) 2 to login:
```



```
enter 5 to print blockchain over a particular node:
enter 6 to mine a block over a node:
enter 7 to add a node:
enter 8 to print data on all nodes:
enter 9 to print trxs data on all nodes:
8
node-1
Block Hash:52e08da95ae8b16302c6ea9244b9eb3912bb76573eef9bb2d16353e6ed7dc8d0
Block merkelRoot:Hash@1b4fb997
Block confirmations1
POW or nonce value2e36a
node-2
Block Hash:52e08da95ae8b16302c6ea9244b9eb3912bb76573eef9bb2d16353e6ed7dc8d0
Block merkelRoot:Hash@1b4fb997
Block confirmations1
POW or nonce value2e36a
```



```
node-1
Block No: 0
username :rohan
2022-04-25 19:11:21.093
medical record stored: { heart beat 200 low fever}
data validated by: rainbow
username :rohan
2022-04-25 19:10:55.362
medical record stored: { ghot is there}
data validated by: yashoda
username :rohan
2022-04-25 19:10:41.407
medical record stored: { hello from digmytisis}
data validated by: omni
Block No: 1
username :rohan
2022-04-25 19:10:20.969
medical record stored: { tuber not found}
data validated by: omni
username :rohan
2022-04-25 19:10:04.036
medical record stored: { sugar 100 low}
data validated by: yashoda
username :hello
null
medical record stored: {hello}
data validated by: self
Block No: 2
username :rohan
```



```
2022-04-25 19:10:04.036
medical record stored: { sugar 100 low}
data validated by: yashoda
username :hello
null
medical record stored: {hello}
data validated by: self
Block No: 2
username :rohan
2022-04-25 19:10:20.969
medical record stored: { tuber not found}
data validated by: omni
username :rohan
2022-04-25 19:10:04.036
medical record stored: { sugar 100 low}
data validated by: yashoda
username :hello
null
medical record stored: {hello}
data validated by: self
node-2
Block No: 0
username :rohan
2022-04-25 19:11:21.093
medical record stored: { heart beat 200 low fever}
data validated by: rainbow
username :rohan
2022-04-25 19:10:55.362
medical record stored: { ghot is there}
data validated by: yashoda
username :rohan
2022-04-25 19:10:41.407
```