PROJECT DESIGN PHASE &
SOLUTION ACHITECTURE

Date	04 November 2023
Team ID	NM2023 TMID00389
Project Name	Blockchain 1“eclinology For Electronic Health Records
klaxnnuiR Starks	4 Marks

Solution Architecture:

Before the introdiiction of smart contacts on the bloekcliain, the ion in discussions on Electronic Health Record (EHR) Maiiagenient focused on wlietliei to use cloud infrastriictuics or local centralized systems for storing and sliaruig EHRs. These centi’a1ized systems implied that each hospital and healthcare conpany would have to keep data on preiiiises in locally managed structures nd databases.

A bloclicliain is a data structui’e where the records are stored iii a lanced sequence of blocks. This sequence forms a distributed ledger, which means it is replicated in iiniltiple machines, called nodes, that communicate with one another. The nodes form a peer-to-peer network where every update to the ledger must be accepted by the network using a consensus protocol. The consensus protocol assui’es tliat everybody has the same view on the status of the system

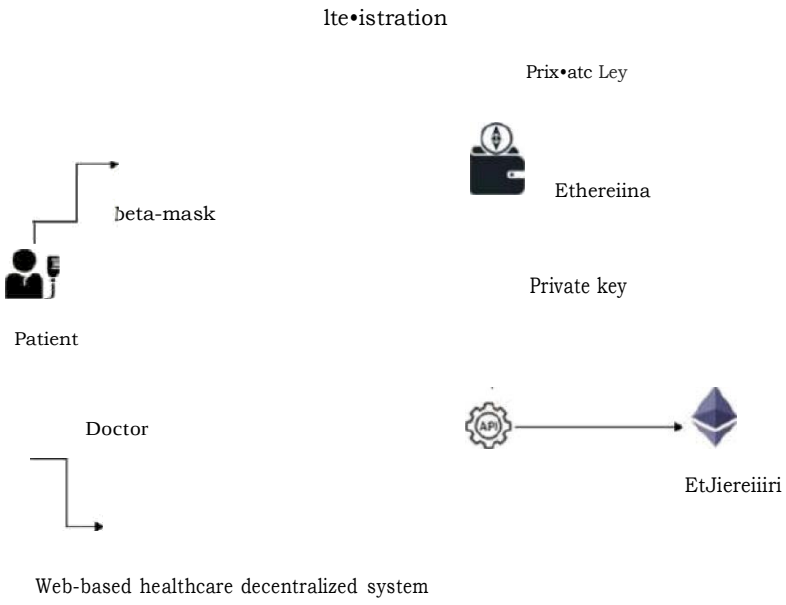
After the design and implementation of a basic EHRs inanagelTlent systeiti end the execution of a set of test cases, it will be possible to discuss the benefits and trade-offs that the system entails. The discussion will focus on the peifoiiiiance of a perinissioned blockcliain for EHRs management. Normal and disaster scenarios will be compared using the following indicators to get important insights on how a crisis affects the operations of a blockcliain neHvork:

• Success iute: the number of successes and fia ihires of a batch of requests. It is irriportant to linilt the number of failures caused by a surge of i’equests during an emergency;

• Transaction commit and read latency: this refers to the time it takes for the blockchain-based system to process an access request to an EHR in n disastei' situation. It is important as tiinelñiess in getting health Sta, especially in emergencies, is critical.

• Transaction commit and state read throughput (TPS): this refers to the number of requests tliat can be managed by the system at the saiiie time. Being able to access and modify a growing iiiiiiiber or i'eqiiest is essential to enable everybody Io interact with the system;

• Resourcc consumption {CPU, Memory and network IO): it is necessary to take these parameters into account as they affect all the other iudicatoi's.

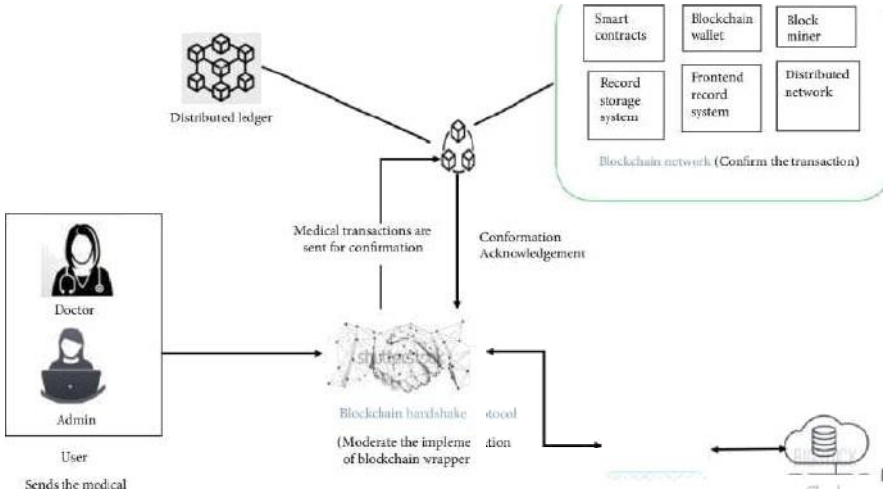


6.2 Sprint Planning & Estimation:

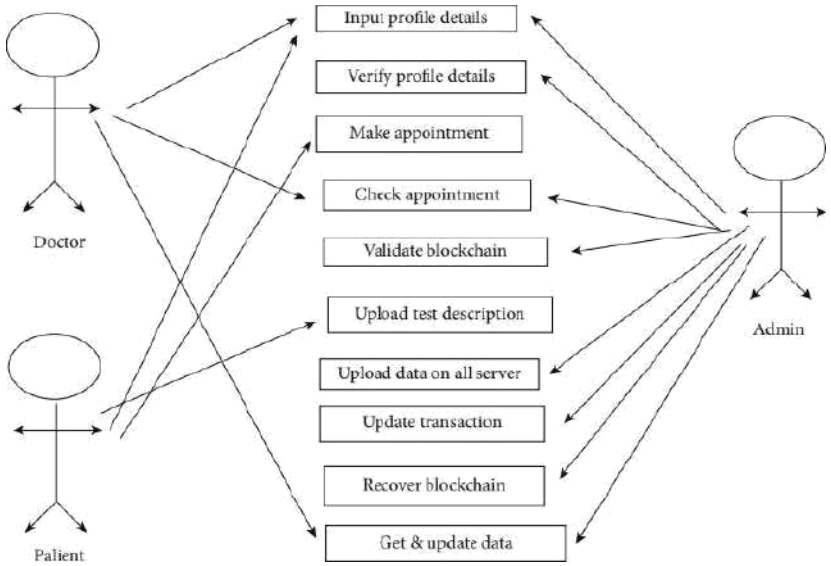
Block Diagi out illustrates the block diagram Our proposed design has four major coiTiponents: a user application, a blockcliain linndsliake protocol, a cloud, and a public blockcliain neuvork. Ilie system is a virtual representation thai serves two purposes. For steers, it provides users with access to application uiterfaces. Doctors and system administrators are two p'pes of usei's in our system Each riser has a distinct hinction. As a result, the user application delivers different user intei'fnces depending on the user role. Second, based on the data entered by the user, the user application creates an initial ti'ansaction. For the piupose of confirmation, the transaction is submitted to the blockcliaui handshake protocol. Finally, a user interface establishes the relationship between users and the blockchain handshake protocol. The proposed architecture 's fundamental component is the blockchain handshake (BH) protocol. TliiS conyonent connects the database servei, the blockchain network, and the cloud-based health recoi'd system which acts as a

wrapper'. This proposed architecture makes use of the Ethereum blockchain network. A distributed ledger that connects blockchain nodes is known as the public blockchain neDvork. Blockchain nodes are miners who are in charge of updating the blockchain based on the decision method. Alternatively, blockchain nodes accept Oansactions and use the network's smart contracts to authenticate them.

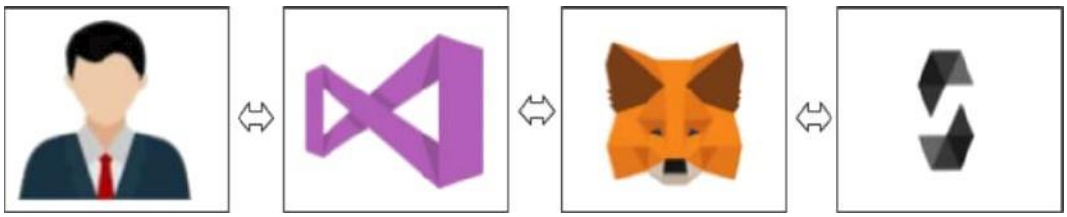
Block diagram of 4he blockchain-based EHR sys4em



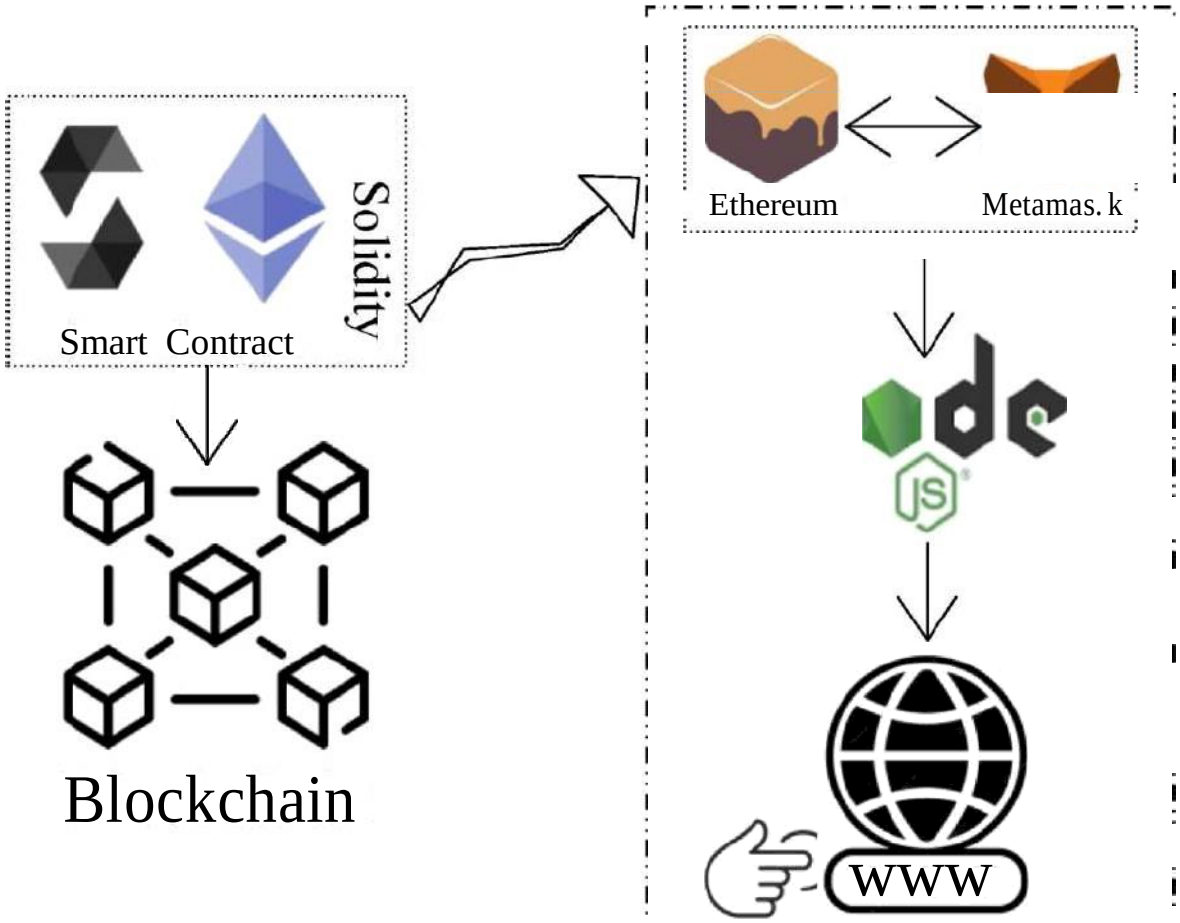
Use case diagram of the EHR systerrt



Solution Architecture Diagram



End User | Frontend (VS code) | Metamask | Solidity code fblod‹chain code)



Output Screenshots

Local Host & address : http:/.'localliost:3000/

Screenshot of output

1'ideo Demo Link : bttps://voutu.be/cNsvMfDFRks?feature=shared

Source code 1 :

/ SPDX-License-Identifier: hOT

pragnin solidity ^0.8.0;

contract HealtliRecords

stnict PatientRecord

Sti'ing Naine;

address }iatientAddress;

stiñiig dieses;

string contactlnfo;

mapping(uint256 => PatientRecord) public i'ecords:

event RecordCreated(uint2ñ6 indexed recordld, address indexed patientAddress);

event RecordTraiisferred

uint2?6 indexed recordIA

```
        address indexed fi'oiii,
        address indexed to

modifier onlyOwner(uint2ñ6 recorded)

    requii'e(msg.sender      records[recordId].patientAddress,"Only contract owner can call this");


function createReeord

        uint256 recordld,
        string memory name, address     atientAddi'ess, string memory diseases, string iiiernory
         contactlnfo

external (
        records[recordId] Naiiie = name.
        records[recordId].patientAddress  =      atientAddress;
        records[recordId].dieses =   diseases;
        records[recorded]. contactlnfo =    contactlnfo;
        emit  RecordCreated(recoidId,      atieiitAddress);

        function  transferRecoid(uint2ñ6  recordI$ address newowner) external
onlyouaiei'(i'ecordld)

        //requii'e(recor‹ts[i'ecordId].patientAddress      iiewOwner, "New Owner should have
different  Address");
        reqiiii'e(records[recordId].patientAddress       iiisg.sender,  "Only record owner can
transfer") ;
        records[recorded].patientAddi'ess  =  newOwner ;
        eiTfit RecordTransferi'ed(recordld, records[recordI‹t].patientAddress,  newOwner)

        function getRecordData

        uiiit256 recordId
```

```
)
external view retunis (string inernory, address, string niernory,string memory) (
i'etum (records[recordId].Name,
records[recorded].patientAddi'ess,
records[recordI‹1].dieses,
records[recordId].contactInfo);

function getRecordOwner(uint256 recordld) external view returns (address)

return records[recordId].patientAddress;
```

Source code 2 :

```
const ( ethers ) = require(”ethers");
const abi = [

"anonymous" : false,
"iriputs": [

"indexed": Oue,
"internalType": "uint256",
"same"’ "recoi’dId",
"type": "uint2S 6"


"indexed": true,
"inteinalType": "address",
"naive": "pntientAddress",
"type": "a‹1dress"


uu+e' "Reco+’dCrentecP,
"the":"event"
```

"anonymous": false,
"ñiputs": [

"indexed": true,
"inteinalType": "uint256",
"name": "recordld",
"type": "uirit2ñ6"

"indexed": true,
"inteiiinlType": "address",
"name": "front",
"type": "address"

"indexed": Oue,
"intcrnalType": "addi'ess",
"name": "to",
"type": "address"

"name": "RecordTrausferred",
"i ype": "event"

"inputs": [
(
"inteinalType": "uint256",
"naiTle": "recordld",
"type": "uint2ñ6"

```
  "inteinalType": "string",
  "naiTle": "nalne",
  "type": "string"

  "inteinalType": "address",
  "name": "    atientAddi’ess",
  "type": "addi’ess"

  "inteinalType": "sti’ing",
  "name": "  diseases",
  "type": "sti’iiig"

  "inteinalType":  "string",
  "name":  "  contactInfo",
  "type": "string"

"name": "createRecord",
"outputs": [],
"stateMiitability"’ "nonpayoble",
"type": "function"

"inputs": [

  "internalType": "uint256",
  "name": "recordI‹t",
  "type": "uint2ñ6"

"name": "getRecordData",
```

```
"outputs": [

  "internalType": "string",
  "name": "",
  "type": "string"


  "internalType": "address",
  "name": "",
  "type": "addi‘ess"


  "intemalType": "string".
  "name": "",
  "type": "string"


  "internalType": "string",
  "name": "",
  "type": "string"


"stateMutability"’ "view",
"type": "function"


"inputs": [

  "internalType": "uint256",
  "name": "recordld",
  "type": "uint256"


"narile": "getRecordOwner",
```

```
"outputs": [

  "inteinalType": "address",
  "name": "",
  "type": "a‹tdress"


"stateMutability": "view",
"the": "fit ction"


"inputs": [

  "internalType": "uint256",
  "name": "”’,
  "type": "uint2?6"


"name":  "i’ecords",
"outputs": [

  "internalType": "string",
  "same"’ "Nmiie",
  "type": "string"


  "inteinalType":  "address",
  "nome": "patientAddress",
  "type": "address"


  "inteinalType": "sti’ing",
  "name": "dieses",
  "type": "string"
```

```
  "inteinalType": "string",
  "name": "contactlnfo",
  "type": "string"

"stateMutabi1ity": "view",
"the": "fit ction"

"inputs": [

  "internalType": "uint256",
  "name": "recordld",
  "type": "uint2?6"

  "intcrnalType": "addi'ess",
  "name": "newowner",
  "type": "address"

"name": "transferRecord",
"outputs": [],
"stateMutability": "nonpayable",
"type": "function"

if(!windov .etliereuin) (
 alert('hfeta Mask Not Found')
 window o}en("https:/'rnetanrsk to/download/")
```

```
export const provider = new ether’s.providers.Web3Provider(window.etliereuni);
export const signed = provider’.get Signer():
export const addi’ess = "0x8837a10cD7813729bCEB 5381z¥6D435E986240d4"
export const contract = new ethers.Contract(a‹tdress, abi, signer)
```