ONLINE BANKING SYSTEM USING BLOCKCHAIN

Report submitted in partial fulfillment of the requirement for the degree of

B.Tech

in
Computer Science & Engineering

Under the supervision

of

Ms. Akanksha Dhamija
Assistant Professor, CSE

By

Rajat (94), Navneet (97)
Avnish (104), Pushan (114)

Department of Computer Science & Engineering
Bhagwan Parshuram Institute of Technology
PSP-4, Sec-17, Rohini, Delhi-89

DECEMBER-2021

DECLARATION

-

This is to certify that Report titled "Online Banking System Using Blockchain", is submitted by us in partial fulfillment of the requirement for the award of degree B.Tech. in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. It comprises of our original work. The due acknowledgement has been made in the report for using others work.

Date: 27-12-2021

Rajat

Rajat Kumar (41520802718)

Navneet

Navneet (41820802718)

Avnish Singh

Avnish Singh (4280802718)

PushanVerma

Pushan Verma (35820802718)

Certificate by Supervisor

This is to certify that Report titled "Online Banking System Using Blockchain" is submitted by **Rajat (94), Navneet (97), Avnish (104), Pushan (114)** in partial fulfillment of the requirement for the award of degree B. Tech in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. It is a record of the candidates own work carried out by them under my supervision. The matter embodied in this Report is original and has not been submitted for the award of any other degree.

akanksha

(Signature)

Date: 27-12-2021

Supervisor

Ms. Akanksha Dhamija

Assistant Professor, CSE

Certificate by HOD

This is to certify that Report titled "Online Banking System Using Blockchain" is submitted by **Rajat (94), Navneet (97), Avnish (104), Pushan (114)** under the guidance of "Dr. Achal Kaushik "in partial fulfillment of the requirement for the award of degree B. Tech in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. The matter embodied in this Report is original and has been dully approved for the submission.

(signature)

Date: 27-12-2021 **Dr. Achal Kaushik**

Table of Contents

CHAPTER -1

Introduction:

Online banking has a considerable amount of commercial significance. No matter the existing no. of banks, online banking will always be a need. Users always want the flexibility to access their accounts and to perform transactions on runtime. Our bank management system provides all the important internet-banking services and functionalities that are in high demand. Current banking systems are based on central server mechanisms where all the personal information of account holders, his/her bank balance, and all other necessary information related to the bank are stored. All other branches are connected to the central server where every branch retrieves personal information, bank balance and history from the server. Failure in the central server causes all other branches to fall down which results in great damage to its users. A Blockchain is the “current” part of a Blockchain which records some or all of the recent transactions, and once completed, goes into the blockchain as a permanent database. Each time a new block is generated based on the completion of each block. Blocks are linked together in a linear fashion where each block contains a hash value of the previous block. Through this Paper we are introducing, how blockchain Consensus algorithms, hashing techniques, salting techniques, time stamp algorithm and Hashing can be helpful to solve banking issues and make the overall banking procedure smooth and secure, it is decentralized. Blockchain fundamentals are based on consensus algorithms. Consensus is a process in computer science used to achieve agreement on a single data value among distributed systems. Consensus Algorithms in Blockchain helps to choose best peer to sign the next block in a network. In consensus Algorithm, proof of work, proof of stake, proof of capacity, proof of importance is discussed heavily. The Project deals with banking systems which involves user registration, database, transactions, book keeping etc. Through blockchain the main activity i.e. transaction.

CHAPTER -2

RELATED WORK

The biggest problem faced by online systems handling payments or taking orders is the security. Whenever we perform a transaction using such systems, we want its record to be immutable. We want to save our transactions at a decentralized place because that way, we would not need to trust any single provider. Keeping this in mind, our online banking system saves all the transactions performed by any customer directly to the blockchain network. In addition, the system is connected to the blockchain in a way that allows the admin to login and gets all of the different types of transactions of all the customers directly from the blockchain network. Security is maintained using two techniques. Blockchain and hashing.

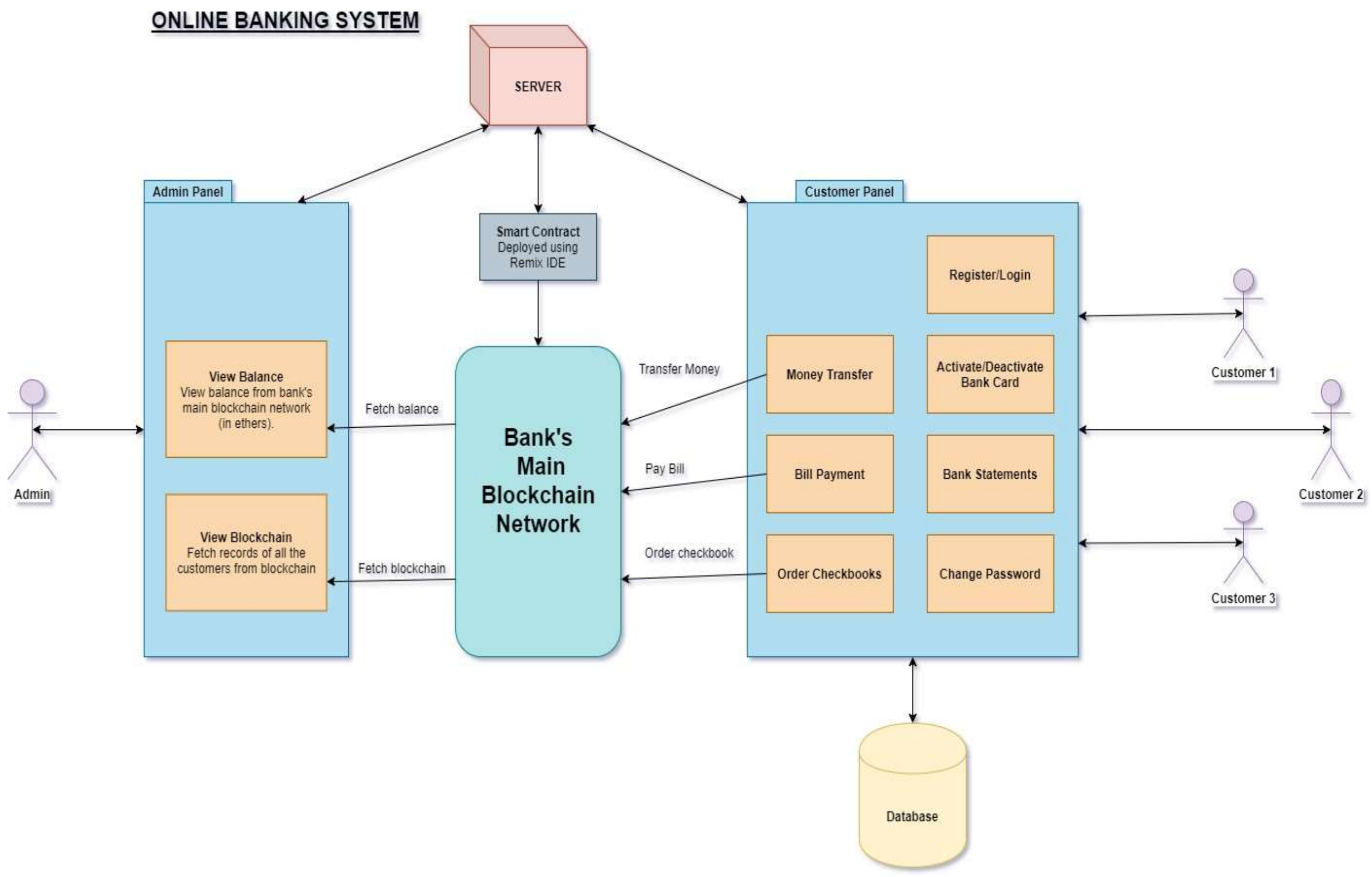
1. Transactions are recorded using blockchain technology. Therefore, no centralized figure can change the transactions' records.
2. Passwords are "hashed" before saving them to the database. As a result, if a person with bad intentions somehow accesses the database or all the server files, even then, the person will not be able to interpret the user passwords.

Finally, one other problem our system solves is that it displays the data coming directly from the blockchain network into human-readable format (in the admin panel).

CHAPTER – 3

Architecture and Design:

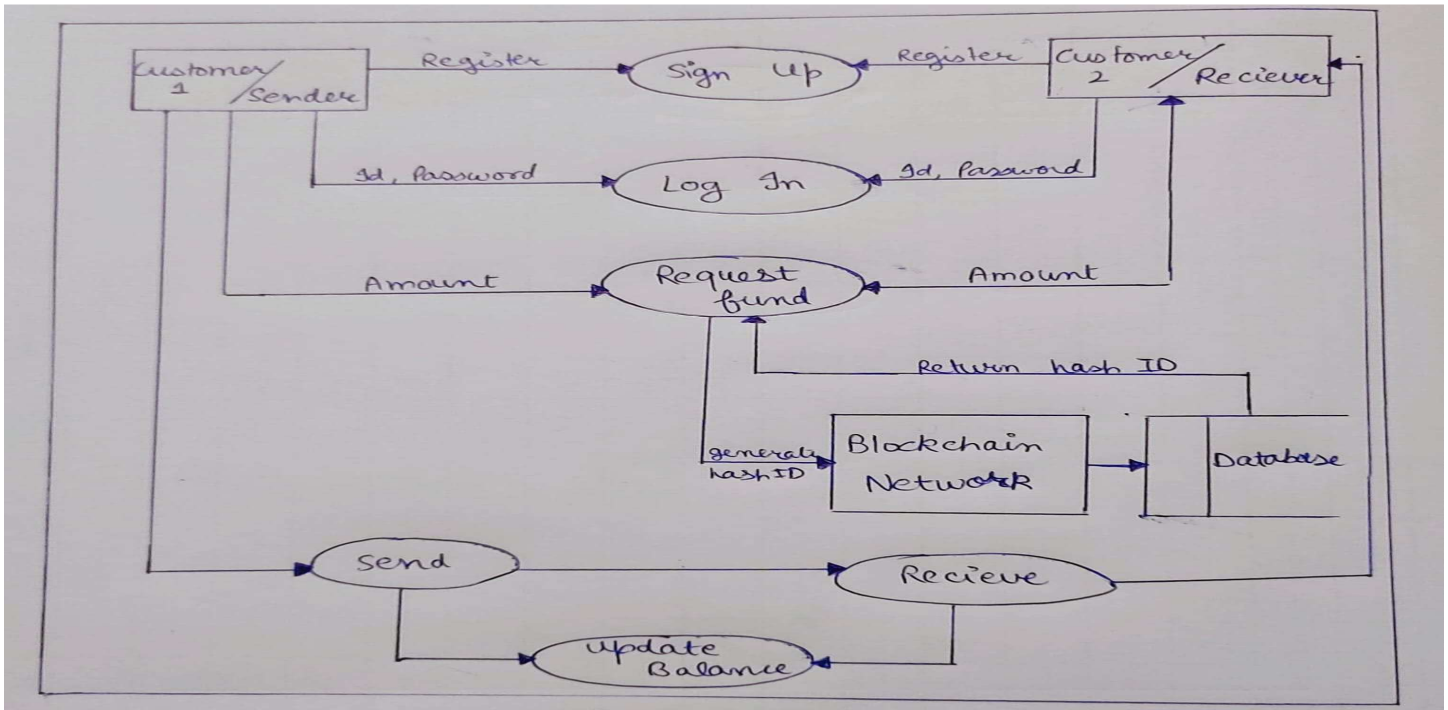
USE CASE DIAGRAM:



DFD DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. A context diagram is a top level (also known as "Level 0") data flow diagram. It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities.

Draw the context diagram first, followed by various layers of data flow diagrams.



1st Level DFD

Chapter -4

PROPOSED WORK

Algorithms:

1.Consensus Algorithms:

Before Knowing Consensus Algorithms, we must know what is Consensus, it is a process in computer science used to achieve agreement on a single data value among distributed systems. In a distributed computing environment, a consensus algorithm is a technique that allows all participants in a blockchain network to reach a shared understanding (consensus) on the current data state of the ledger and trust unknown peers. It includes:

1. Proof Of Work

This Algorithm was introduced with first cryptocurrency i.e., Bitcoin by Satoshi Nakamoto. It is the most known way of confirming transactions. Proof of work (PoW) is a form of cryptographic proof in which one party (the prover) proves to others (the verifiers) that a certain amount of a specific computational effort has been expended. The first node to complete all necessary calculations receives a reward from blockchain network. All nodes compete against each other by increasing capacity of computing resources. The goal of proof-of-work algorithms is not to prove that particular tasks were completed or that a computational challenge was "solved," but to discourage data modification by imposing high energy and hardware-control requirements.

2. Proof Of Stake

Proof of stake (PoS) protocols are a type of blockchain consensus method that selects validators based on their bitcoin holdings. This is done to circumvent proof of work methods' high computing costs. Therefore , node with greater number of resources get chosen to generate next block in blockchain.
Just like in company, the one who has the highest number of shares has powers , node with greater number of resources is appointed to generate block in blockchain.

(a)Delegated POS (Proof of Stake)

Delegated POS is a type of POS consensus algorithm, in which blocks are signed by selective representatives. Owners of the largest balances choose their representative and each of them receives right to sign blocks on blockchain network. If by any chance, if the representative missed turning a block, he gets deprived from delegated votes and leaves council. Its advantages are that balance owners have a opportunity to delegate their votes without delegating actual resources. Unlike POS, amount of unnecessary work is reduced during the process of choosing next voter.

(b) Leased POS (Proof of Stake)

Leased POS is another type of consensus algorithm user has a possibility to lease out their balance to mining nodes, in return mining nodes share a part of profit with users, which is only supported on waves platform.

3. Proof Of Capacity

POC, allows mining devices in network to use their hard drive space to decide mining rights. Proof-of-Capacity consensus is a step forward from the widely used Proof-of-Work consensus algorithm. Even before mining can begin, processor power and hard disc storage must be set aside. As a result, the system outperforms the Pow. Proof-of-Capacity produces a block in four minutes, whereas Proof-of-Work takes ten minutes. Supported by bitcoin. It tries to tackle the hashing problem in the PoW scheme. If there are more solutions or plots accessible on the computer, there is a better possibility of winning the mining dispute.
The PoC technique is as follows:
- each miner calculates a huge quantity of data, which is stored on a node's disc subsystem: hard drive, cloud storage, or other. Space is the name of the first dataset in the PoC.
- The miner reads a tiny data set equal to 1/4096, or around 0.024 percent of all stored data, for each new block on the blockchain. The miner can then generate a new block after receiving the result (deadline) as elapsed time since the last block was created.
- The miner who met the minimum deadline time signs the block and earns a transaction reward.

4. Proof Of Importance

The NEM blockchain platform employs this consensus mechanism. The quantity of resources available on a user's balance and the number of transactions in their wallet determine their importance in the NEM network. Unlike the more prevalent PoS method, which solely considers user balance, Poi considers both the number of resources and the amount of user activity in the blockchain network. This strategy encourages customers to not just maintain money in their accounts, but also to spend it. Line time is responsible for signing the block and receiving a payment for each transaction.

2. Hashing:

Hashing is the process of scrambling raw data to the point that it can no longer be reproduced in its original form. It takes a chunk of data and runs it through a function that manipulates the plaintext with math. The hash function produces the hash value/digest, which is the result of the hash function.
Two main applications of Hashing are:
Password Hashes: In most website servers, it converts user passwords into a hash value before being stored on the server. It compares the hash value re-calculated during login to the one stored in the database for validation.
Integrity Verification: When it uploads a file to a website, it also shared its hash as a bundle. When a user downloads it, it can recalculate the hash and compare it to establish data integrity.
SHA 256 is a member of the SHA 2 algorithm family, with SHA standing for Secure Hash Algorithm. It was a cooperative effort between the National Security Agency and the National Institute of Standards and Technology to introduce a successor to the SHA 1 family, which was steadily losing power against brute force assaults. It was published in 2001.The 256 in the name refers to the final hash digest value, which means that regardless of the amount of plaintext or cleartext, the hash value will always be 256 bits.SHA 256 is more or less comparable to the other algorithms in the SHA family. Look into learning a little more about their policies immediately.
The significance of the 256 in the name stands for the final hash digest value, i.e. irrespective of the size of plaintext/cleartext, the hash value will always be 256 bits.
You can divide the complete process into five different segments, as mentioned below:

Padding Bits

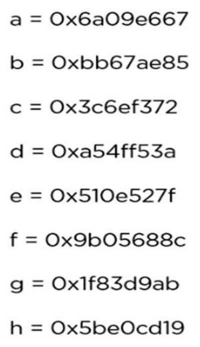
It adds some extra bits to the message, such that the length is exactly 64bits short of a multiple of 512. During the addition, the first bit should be one, and the rest of it should be filled with zeroes.

Padding Length

You can add 64 bits of data now to make the final plaintext a multiple of 512.You can calculate these 64 bits of characters by applying the modulus to your original cleartext without the padding.

Initialising the Buffers:

You need to initialize the default values for eight buffers to be used in the rounds as follows:



```
a = 0x6a09e667
b = 0xbb67ae85
c = 0x3c6ef372
d = 0xa54ff53a
e = 0x510e527f
f = 0x9b05688c
g = 0x1f83d9ab
h = 0x5be0cd19
```

Compression Functions

The entire message gets broken down into multiple blocks of   512 bits each. It puts each block through 64 rounds of operation, with the output of each block serving as the input for the following block

3. Salting:

In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes data, a password or passphrase. Salts are used to safeguard passwords in storage.

4.TimeStamp:
The timestamp or timestamp is a small data stored in each block as a unique serial and whose main function is to determine the exact moment in which the block has been mined and validated by the blockchain network. In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes data, a password or passphrase. The timestamp or timestamp is a small data stored in each block as a unique serial and whose main function is to determine the exact moment in which the block has been mined and validated by the blockchain network. In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes data, a password or passphrase. Salts are used to safeguard passwords in storage. One of the main uses of timestamp is to establish the parameters of the process of mining. This is because these timestamps allow nodes to correctly adjust the mining difficulty to be used for each block generation period. Timestamps help the network determine how long it takes to extract blocks for a certain period, and from there adjust the mining difficulty parameter.

Offered functionalities:

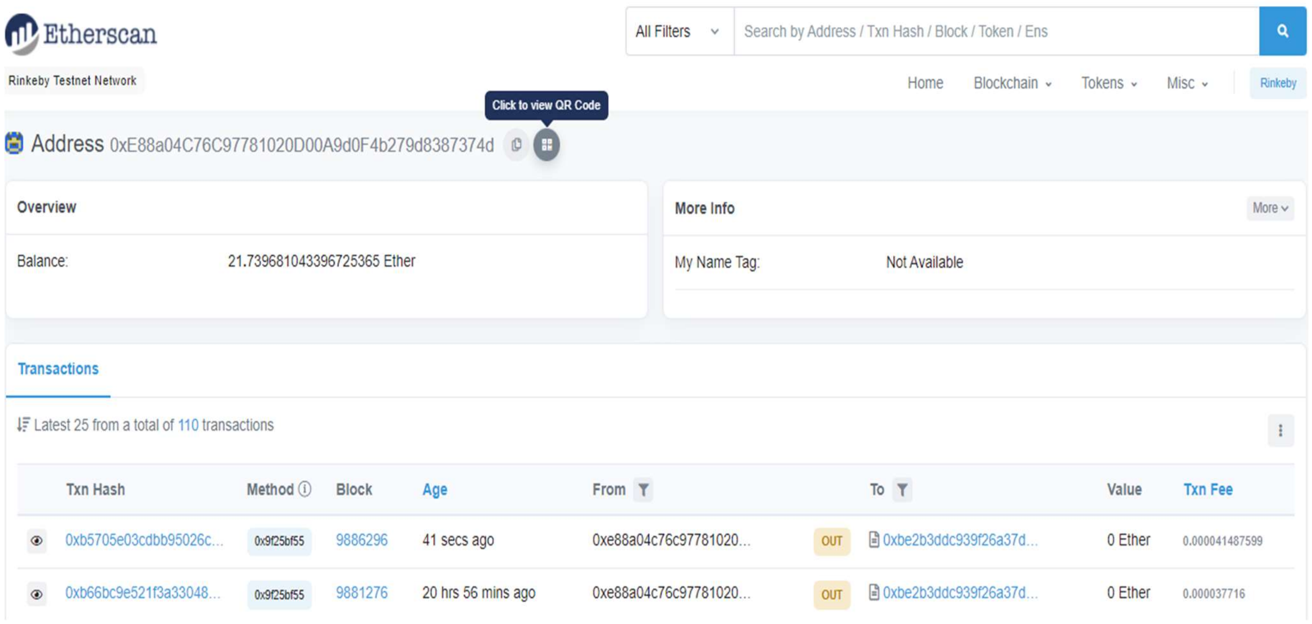
1. Registration/Login
2. Debit card activation/deactivation
3. Money Transfer
4. Bill Payment
5. Generate bank statements
6. Order checkbooks
7. Change password
8. View bank details

CHAPTER - 5

Implementation and Results

Saving The Records To The Blockchain:

It is approximately taking 41 seconds to save a transaction on Ethereum's test network.



Address 0xE88a04C76C97781020D00A9d0F4b279d8387374d

Overview

Balance: 21.739681043396725365 Ether

More Info

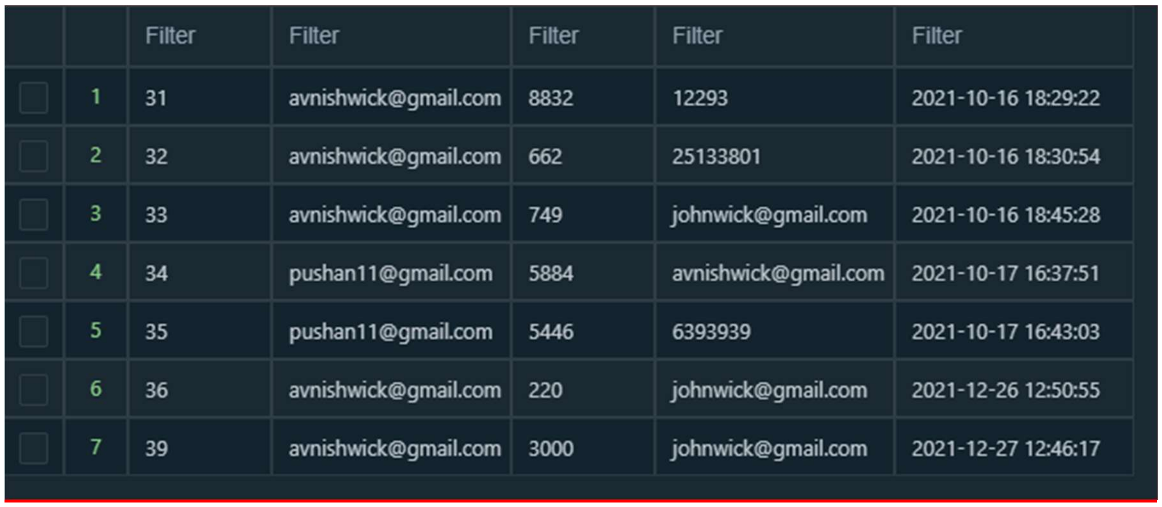
My Name Tag: Not Available

Transactions

Latest 25 from a total of 110 transactions

Txn Hash	Method	Block	Age	From		To	Value	Txn Fee
0xb5705e03cdbb95026c...	0x9f25bf55	9886296	41 secs ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000041487599
0xb66bc9e521f3a33048...	0x9f25bf55	9881276	20 hrs 56 mins ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000037716

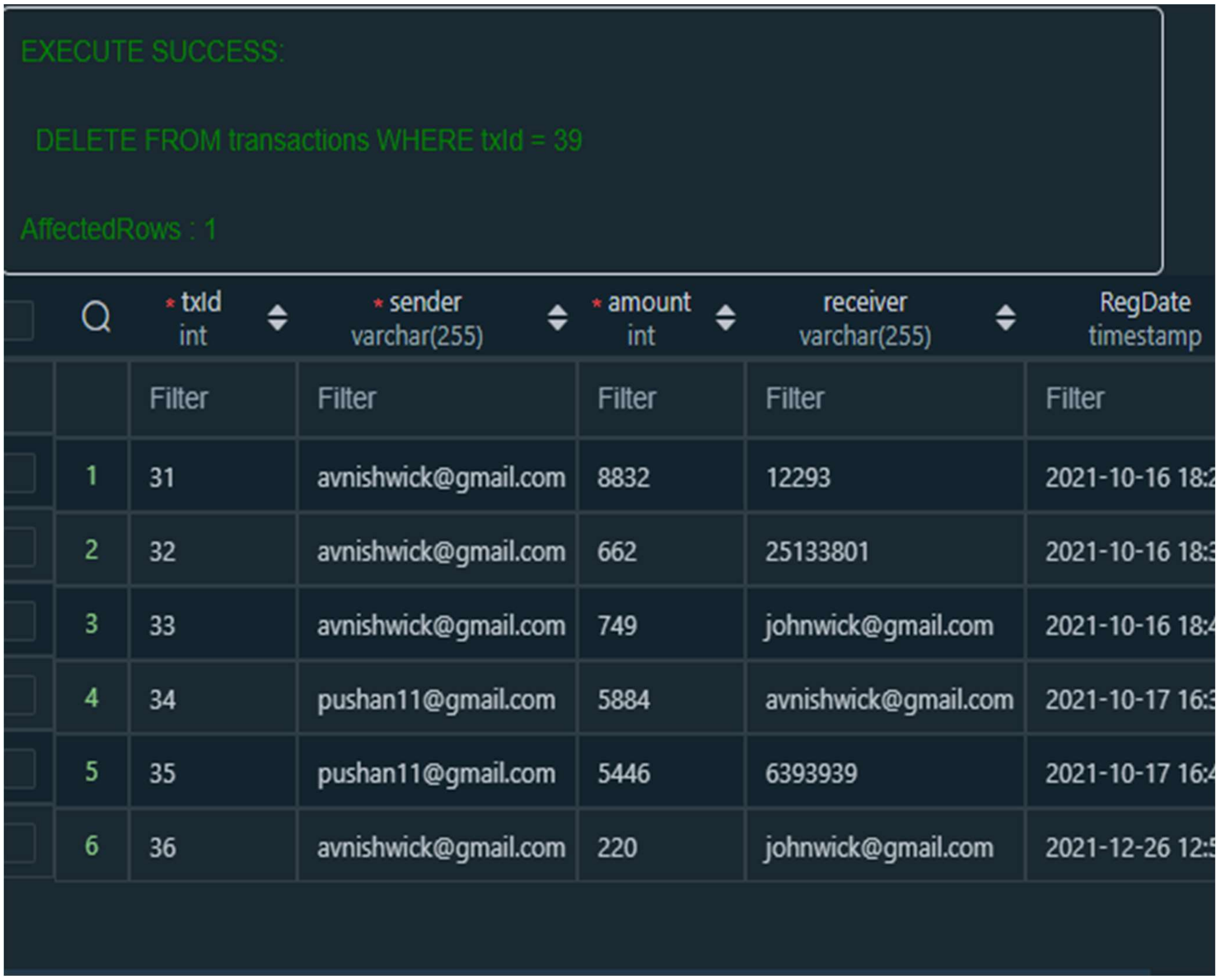
Saving transaction to a database is much faster but the overhead of saving to the blockchain is valuable.



		Filter	Filter	Filter	Filter	Filter
☐	1	31	avnishwick@gmail.com	8832	12293	2021-10-16 18:29:22
☐	2	32	avnishwick@gmail.com	662	25133801	2021-10-16 18:30:54
☐	3	33	avnishwick@gmail.com	749	johnwick@gmail.com	2021-10-16 18:45:28
☐	4	34	pushan11@gmail.com	5884	avnishwick@gmail.com	2021-10-17 16:37:51
☐	5	35	pushan11@gmail.com	5446	6393939	2021-10-17 16:43:03
☐	6	36	avnishwick@gmail.com	220	johnwick@gmail.com	2021-12-26 12:50:55
☐	7	39	avnishwick@gmail.com	3000	johnwick@gmail.com	2021-12-27 12:46:17

Deleting A Record:

Deleting a record from database.

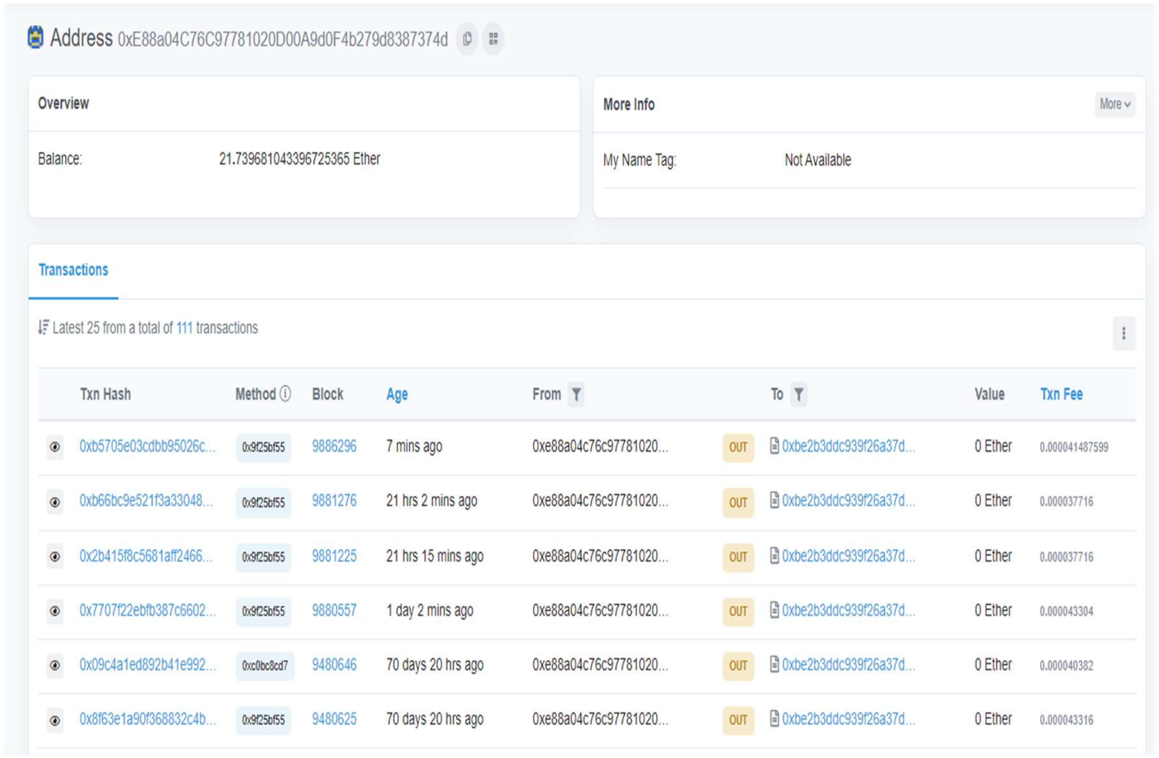


```
EXECUTE SUCCESS:

  DELETE FROM transactions WHERE txId = 39

AffectedRows : 1
```

	* txId int	* sender varchar(255)	* amount int	receiver varchar(255)	RegDate timestamp
	Filter	Filter	Filter	Filter	Filter
1	31	avnishwick@gmail.com	8832	12293	2021-10-16 18:
2	32	avnishwick@gmail.com	662	25133801	2021-10-16 18:
3	33	avnishwick@gmail.com	749	johnwick@gmail.com	2021-10-16 18:
4	34	pushan11@gmail.com	5884	avnishwick@gmail.com	2021-10-17 16:
5	35	pushan11@gmail.com	5446	6393939	2021-10-17 16:
6	36	avnishwick@gmail.com	220	johnwick@gmail.com	2021-12-26 12:

Record has been deleted.

But the same record is still there on the blockchain and cannot be deleted.


Address 0xE88a04C76C97781020D00A9d0F4b279d8387374d

Overview	
Balance:	21.739681043396725365 Ether

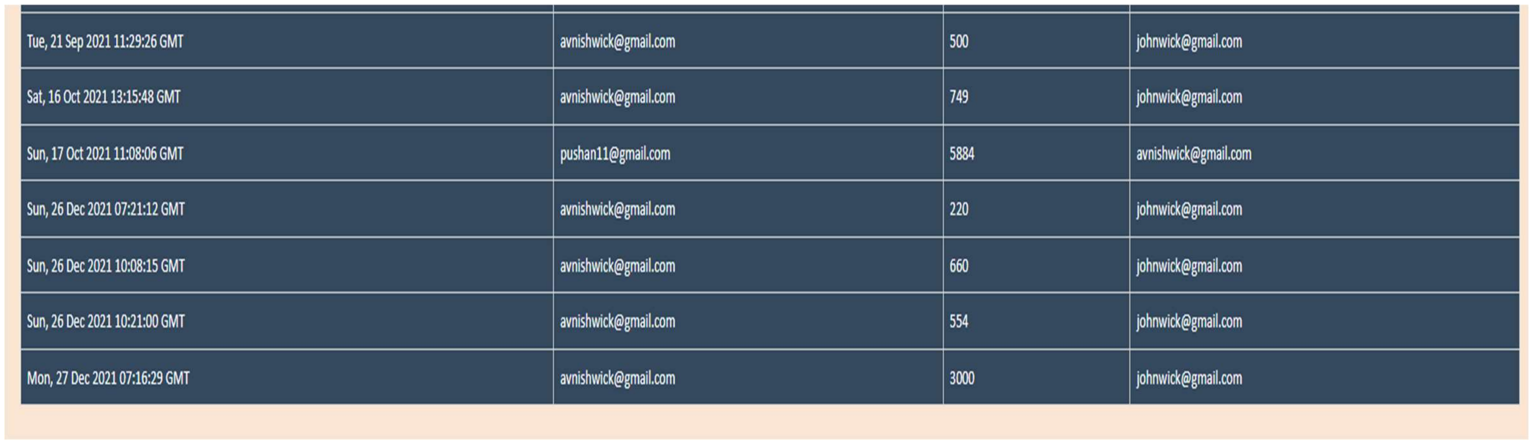
More Info	
My Name Tag:	Not Available

Transactions

Latest 25 from a total of 111 transactions

	Txn Hash	Method	Block	Age	From		To	Value	Txn Fee
👁	0xb5705e03cdbb95026c...	0x9f25bf55	9886296	7 mins ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000041487599
👁	0xb66bc9e521f3a33048...	0x9f25bf55	9881276	21 hrs 2 mins ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000037716
👁	0x2b415f8c5681aff2466...	0x9f25bf55	9881225	21 hrs 15 mins ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000037716
👁	0x7707f22ebfb387c6602...	0x9f25bf55	9880557	1 day 2 mins ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000043304
👁	0x09c4a1ed892b41e992...	0xc0bc8cd7	9480646	70 days 20 hrs ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000040382
👁	0x8f63e1a90f368832c4b...	0x9f25bf55	9480625	70 days 20 hrs ago	0xe88a04c76c97781020...	OUT	0xbe2b3ddc939f26a37d...	0 Ether	0.000043316

We can clearly view it from the admin panel.



Tue, 21 Sep 2021 11:29:26 GMT	avnishwick@gmail.com	500	johnwick@gmail.com
Sat, 16 Oct 2021 13:15:48 GMT	avnishwick@gmail.com	749	johnwick@gmail.com
Sun, 17 Oct 2021 11:08:06 GMT	pushan11@gmail.com	5884	avnishwick@gmail.com
Sun, 26 Dec 2021 07:21:12 GMT	avnishwick@gmail.com	220	johnwick@gmail.com
Sun, 26 Dec 2021 10:08:15 GMT	avnishwick@gmail.com	660	johnwick@gmail.com
Sun, 26 Dec 2021 10:21:00 GMT	avnishwick@gmail.com	554	johnwick@gmail.com
Mon, 27 Dec 2021 07:16:29 GMT	avnishwick@gmail.com	3000	johnwick@gmail.com

CHAPTER -6

Conclusion:

Online Bank Management System provides user-friendly interface to the customers and admin. The objective of the project was to provide flexible access of multiple banking services to the customers and secure their transactions. Both objectives were met, Furthermore, our banking system has plenty of room for more development. To conclude, the project is easy to understand, secure and has a lot of potential for development. Using blockchain the system and transactions are secured and even if the system clashes the data entries in the SQL table can be rolled back by the properties of Blockchain. Since bitcoin and other similar cryptocurrencies uses pow as their main algorithm, which is energy consuming and highly costly, pos techniques should be introduced to new systems and as users increases, use of decentralized cloud platform be in use to provide smooth experience to users, hashing techniques sha-256 should be included while hashing. The Algorithms used like hashing and salting make the hash value secure and un-traceable and provide extra-security to the system. So, with the implementation of blockchain the banking system can be more secured and reliable and reduces the amount of loss at times of intrusions.

CHAPTER – 7

FUTURE SCOPE

Blockchain is the vast-field to be discovered. The app contains the basic properties of the blockchain and implemented to the basic banking system which can be extended to the more of the other use cases of banking system like fundraising, NEFTs, Foreign Transactions. Alongside the system right now carries the data in sql table which can be replaced by much secure format so that to reduce the risk of intrusions ahead of it. The banking can be much more decentralized and efficient blockchain in it. Since bitcoin and other similar cryptocurrencies uses pow as their main algorithm, which is energy consuming and highly costly, pos techniques should be introduced to new systems and as users increases , use of decentralized cloud platform be in use to provide smooth experience to users, hashing techniques sha-256 should be included while hashing .

Appendix:

Code:

Server.js:

```
// LOGIN and REGISTRATION START---------------------------------------------------
---------------------------------------
if (process.env.NODE_ENV !== 'production') {
  require('dotenv').config()
}
const express = require('express')
const app = express()
const bcrypt = require('bcrypt')
const bcrypt_js = require('bcrypt-nodejs')
const passport = require('passport')
const flash = require('express-flash')
const session = require('express-session')
const methodOverride = require('method-override')
const Web3 = require('web3')
const BigNumber = require('bignumber.js');
var ethers = require('ethers');
var provider = ethers.getDefaultProvider('rinkeby');


const abiDecoder = require('abi-decoder');
const InputDataDecoder = require('ethereum-input-data-decoder');
const initializePassport = require('./passport-config');

const axios = require('axios');
const Etherscan = require('node-etherscan-api');
//proof of work using block chain
const TOKEN_API = 'NVB7ZC1WEES9RP7ZMZ2AHWHTYWNKH8KN2B';
const eth = new Etherscan(TOKEN_API);
var myAddr = '0xE88a04C76C97781020D00A9d0F4b279d8387374d';
var currentBlock = eth.blockNumber;
var n = eth.getTransactionCount(myAddr, currentBlock);
var bal = eth.getAccountBalance(myAddr, currentBlock);
const fetch = require('node-fetch');
```

```
let urlToGetTransactions = "http://api-
rinkeby.etherscan.io/api?module=account&action=txlist&address=0xE88a04C76C9778102
0D00A9d0F4b279d8387374d&startblock=0&endblock=99999999&sort=asc&apikey=NVB7ZC1WEE
S9RP7ZMZ2AHWHTYWNKH8KN2B";
var urlToGetBalance="https://api-
rinkeby.etherscan.io/api?module=account&action=balance&address=0xE88a04C76C977810
20D00A9d0F4b279d8387374d&tag=latest&apikey=NVB7ZC1WEES9RP7ZMZ2AHWHTYWNKH8KN2B";

// address=0xE88a04C76C97781020D00A9d0F4b279d8387374d
// apikey=NVB7ZC1WEES9RP7ZMZ2AHWHTYWNKH8KN2B

//SETTING UP CONNECTION WITH SMART CONTRACT and BLOCKCHAIN?
var address='0xbe2b3DdC939f26A37D701fe461d8829114E4EeF3'; //my contract's address
var abi=[
  {
    "constant": false,
    "inputs": [
      {
        "name": "sender",
        "type": "string"
      },
      {
        "name": "amount",
        "type": "uint256"
      },
      {
        "name": "receiver",
        "type": "string"
      }
    ],
    "name": "billPayment",
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "constant": false,
    "inputs": [
      {
        "name": "account",
        "type": "string"
      },
      {
        "name": "leaves_Requested",
        "type": "uint256"
      }
    ],
    "name": "storeRequestedCheckbooks",
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "constant": false,
    "inputs": [
      {
        "name": "sender",
        "type": "string"
      },
      {
        "name": "amount",
        "type": "uint256"
      },
      {
        "name": "receiver",
        "type": "string"
      }
    ],
    "name": "transferFunds",
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "constructor"
  },
  {
    "constant": true,
    "inputs": [],
    "name": "getDetails",
    "outputs": [
      {
        "name": "",
        "type": "string"
      },
      {
        "name": "",
        "type": "uint256"
      },
      {
        "name": "",
        "type": "string"
      }
    ],
    "payable": false,
    "stateMutability": "view",
    "type": "function"
  }
]
// abiDecoder.addABI(abi);
// const decoder = new InputDataDecoder(abi);
var
privateKey='EE79F1C4B628A7D6C85FFCA5735D086FA6B0B71AFF23277F0A9253612CEA58DC';
//my account's pvt key
var wallet = new ethers.Wallet(privateKey,provider);


initializePassport(
  passport,
  AccountNum => users.find(user => user.AccountNum === AccountNum),
  id => users.find(user => user.id === id)
)
// LOGIN and REGISTRATION END----------------------------------------------------
---------------------------------------


//DATABASE CONNECTIONS START-----------------------------------------------------
------------------------------------
```

```
let users;
const mysql = require('mysql');
const bodyParser = require('body-parser');
app.use(bodyParser.json());

var mysqlConnection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'rajatadmin',
    database: 'banking',
    port: '3306',
    multipleStatements: true,
});

mysqlConnection.connect((err) =>{
  if(!err)
    console.log('db connection succeeded');
  else
    console.log(err);
});

//DATABASE CONNECTIONS END -----------------------------------------------------
--------------------------------


app.set('view-engine', 'ejs')
app.use("/static", express.static(__dirname + '/static'));
app.use(express.urlencoded({ extended: false }))
app.use(flash())
app.use(session({
  secret: process.env.SESSION_SECRET,
  resave: false,
  saveUninitialized: false
}))
app.use(passport.initialize())
app.use(passport.session())
app.use(methodOverride('_method'))
//*******************************************************************************
**********************
var adminSession = 0;

//START sidebar--------------------------------------
```

```
app.get('/', checkAuthenticated, (req, res) => {
  res.render('sidebar.ejs', { name: req.user.name,
AccountNum:req.user.AccountNum, Balance: req.user.Balance });
})
//END sidebar---------------------------------------

//START login---------------------------------------
app.get('/login', checkNotAuthenticated, (req, res) => {

  if(adminSession===1){
    res.redirect('/admin_sidebar');
  }
  else{
    mysqlConnection.query('select * from b_users;', (err, rows, failed) => {
    if(!err)
      users = rows;
    else
      console.log(err);
  })
  res.render('login.ejs');
  }

})

app.post('/login', checkNotAuthenticated, passport.authenticate('local', {

  successRedirect: '/',
  failureRedirect: '/login',
  failureFlash: true
}))
//END login-------------------------------------------

//START regirtration----------------------------------
app.get('/registration', checkNotAuthenticated, (req, res) => {
  adminSession=0;
  res.render('registration.ejs', {error1: " " })
})

app.post('/registration', checkNotAuthenticated, async (req, res) => {
  adminSession=0;
  var numRows;
  try {
    if (req.body.password === req.body.ConfirmPassword)
    {
      // here 10 is the number of times we do the salting
      // Hashing of the Password Using Salting by the help of Bycrypt npm package
and hash function ****************

        const hashedPassword = await bcrypt.hash(req.body.password, 10)

        let sql = 'INSERT INTO b_users SET ?'

          let post={
              id: Date.now().toString(),
              name: req.body.name,
              CNIC: req.body.CNIC,
              DOB: req.body.DOB,
              DebitCard: req.body.DebitCard,
              PIN: req.body.PIN,
              AccountNum: req.body.AccountNum,
              password: hashedPassword,
              Balance: 50000,
              CardStatus: 'Activated',
              RegDate: new Date(),
          }
          var rowsLength;

          mysqlConnection.query('SELECT * FROM B_users WHERE AccountNum = ?',
[req.body.AccountNum], (err, rows, fields) => {
            if (!err){
              rowsLength=rows.length;
              rowsLength=parseInt(rowsLength);
            }
          })

          setTimeout(function () {
            if (rowsLength>0){
              res.render('registration.ejs', {error1: "This account already
exists." })

            }
            else{
              mysqlConnection.query(sql, post, (err, res) => {
                if(err)
                  console.log(err);//ERROR1 -> (already log in)
                else
                  console.log('successful insertion');

              });
              res.redirect('/login')

            }

            }, 1000);


    }
    else
    {
      console.log('Passwords are not same!');
      res.render('registration.ejs',  {error1: "Passwords do no match!"});
    }
  } catch {
    // res.redirect('/registration');
  }
})
//END registration-------------------------------------

//hard code
pages****************************************************************************
*****************
app.get('/ATM-location', (req, res) => {
  res.render('ATM-location.ejs');
})

app.get('/branch-location', (req, res) => {
  res.render('branch-location.ejs');
})

app.get('/index', (req, res) => {
  res.render('index.ejs');
})
```

```
app.get('/Bank_accounts', (req, res) => {
  res.render('Bank_accounts.ejs');
})

app.get('/asaan_account', (req, res) => {
  res.render('asaan_account.ejs');
})

app.get('/savings_account', (req, res) => {
  res.render('savings_account.ejs');
})

app.get('/current_account', (req, res) => {
  res.render('current_account.ejs');
})

app.get('/consumer_finance', (req, res) => {
  res.render('consumer_finance.ejs');
})

app.get('/cards', (req, res) => {
  res.render('cards.ejs');
})

app.get('/investment_banking', (req, res) => {
  res.render('investment_banking.ejs');
})

app.get('/Introduction', (req, res) => {
  res.render('Introduction.ejs');
})

app.get('/our-brand', (req, res) => {
  res.render('our-brand.ejs');
})

app.get('/History', (req, res) => {
  res.render('History.ejs');
})

app.get('/contact-detail', (req, res) => {
  res.render('contact-detail.ejs');
})

//logout and check
authentication*******************************************************************
***********
app.delete('/logout', (req, res) => {
  adminSession=0;
  req.logOut()
  res.redirect('/login')
})

function checkAuthenticated(req, res, next) {
  if (req.isAuthenticated()) {
    return next()
  }

  res.redirect('/login')
}

function checkNotAuthenticated(req, res, next) {
  if (req.isAuthenticated()) {
    return res.redirect('/')
  }
  next()
}

//DEBIT CARD
PAGE***********************************************************************

app.get('/debitcard', (req, res) => {
  res.render('debitcard.ejs', { name: req.user.name,
DebitCard:req.user.DebitCard, card_status: req.user.CardStatus })
})

app.post('/debitcard', (req, res) => {

  mysqlConnection.query("UPDATE B_users set CardStatus='" + req.body.card + "'
WHERE AccountNum = '" + req.user.AccountNum + "'", (err, rows, fields) => {
      if (!err){
        req.user.CardStatus=req.body.card;
        res.render('debitcard.ejs', { name: req.user.name,
DebitCard:req.user.DebitCard, card_status: req.user.CardStatus })
      }
      else
        console.log(err);
    })

})

//Funds Transfer
PAGE*****************************************************************
app.get('/fundsTransfer', (req, res) => {
  res.render('fundsTransfer.ejs', { name: req.user.name, msg: " "})
})

app.post('/fundsTransfer', (req, res) => {


  var flagReceiver=0;
  var notEnoughBalance=0;

  var amt;
  var remaining_amt;

  mysqlConnection.query('SELECT * FROM B_users WHERE AccountNum = ?',
[req.user.AccountNum], (err, rows, fields) => {
    if (!err)
    {
      amt =parseInt(req.body.amount);
      console.log('amt req: ',amt);
      var senderBal=rows[0].Balance;
      console.log("sender's balance: ", rows[0].Balance);

      if(rows[0].Balance<amt){
        notEnoughBalance=1;
      }
      else{
        //Find if receiving account exists or not
        mysqlConnection.query('SELECT * FROM B_users WHERE AccountNum = ?',
[req.body.account], (err, rows, fields) =>{
      rowsLength=rows.length;
      rowsLength=parseInt(rowsLength);

      if(rowsLength<=0){
        //receiving account does not exist
        flagReceiver=1;
      }
      else{
        //receiving account exists
          //update receiver's balance
          let sql_updateRcvrBal = 'UPDATE B_users SET Balance = Balance+?
WHERE AccountNum = ?';
          let dataRcvr = [amt, req.body.account];

          mysqlConnection.query(sql_updateRcvrBal, dataRcvr, (err, rows,
fields) => {
            if (!err){
              console.log("Receiver's Balance has been updated");

              //update sender's balance
              let sql_update = 'UPDATE B_users SET Balance = Balance-?
WHERE AccountNum = ?';
              let data = [amt, req.user.AccountNum];

              mysqlConnection.query(sql_update,data, (err, rows, fields)
=> {
                if (!err){
                  req.user.Balance=senderBal-amt;
                  console.log("Sender's Balance has been updated");

                  //insert transaction to db
                  let sql = 'INSERT INTO transactions SET ?'

                  let post={
                    sender: req.user.AccountNum,
                    amount: req.body.amount,
                    receiver: req.body.account,
                    RegDate: new Date(),
                  }

                  mysqlConnection.query(sql, post, (err, res) => {
```

```
                            if(err)
                              console.log(err);//write proper output error
                            else
                              console.log('successful insertion tx :)');
                          });


                        }
                        else{
                          console.log(err);
                          console.log("Sender's Balance could not be updated");

                        }
                      })
                    }
                    else{
                      console.log(err);
                      console.log("Could not update receiver's balance");
                    }
                  })

          }
        })



      }
    }
    else
      console.log(err);
})

setTimeout(function () {

  if(notEnoughBalance==1)
    res.render('fundsTransfer.ejs', { name: req.user.name, msg: "Not enough
balance. Try again."})
  else if(flagReceiver==1)
    res.render('fundsTransfer.ejs', { name: req.user.name, msg: "Receiving
account does not exist!"});
  else{
    //successful transaction

    console.log('sender: ', req.user.AccountNum);
    console.log('amount: ', amt);
    console.log('receiver: ', req.body.account);

    var contract = new ethers.Contract(address,abi,wallet); //for setter function

    var sendPromise = contract.transferFunds(req.user.AccountNum, amt,
req.body.account);

    sendPromise.then(function(tx){

      console.log(tx);

      res.render('fundsTransfer-res.ejs', { name: req.user.name});
   });

  }
}, 1000);


})
//END of funds transfer's page-------------------------------------

//Bill Payment's page
app.get('/BillPayment', (req, res) => {
  res.render('BillPayment.ejs', { name: req.user.name, msg: " "})
})

app.post('/BillPayment', (req, res) => {
  var amt;
  var remaining_amt;

  mysqlConnection.query('SELECT * FROM B_users WHERE AccountNum = ?',
[req.user.AccountNum], (err, rows, fields) => {
    if (!err)
    {
      console.log('amount in db',rows[0].Balance);
      amt =parseInt(req.body.amount);
      // amt =rows[0].Balance;
      console.log('amt req: ',amt);
      remaining_amt=rows[0].Balance-amt;
      console.log('remaining balance: ',remaining_amt);

      if(rows[0].Balance<amt){
        res.render('BillPayment.ejs', { name: req.user.name, msg: "Not enough
balance. Try again."})
      }
      else{
        let sql_update = 'UPDATE B_users SET Balance = ? WHERE AccountNum = ?';

        let data = [remaining_amt, req.user.AccountNum];

        mysqlConnection.query(sql_update,data, (err, rows, fields) => {
          if (!err){
            req.user.Balance=remaining_amt;
            console.log('Balance has been updated');
          }
          else
            console.log(err);
        })

        let sql = 'INSERT INTO transactions SET ?'

        let post={
          sender: req.user.AccountNum,
          amount: req.body.amount,
          receiver: req.body.account,
          RegDate: new Date(),
        }

        mysqlConnection.query(sql, post, (err, res) => {
          if(err)
            console.log(err);//write proper output error
          else {
            console.log('successful insertion tx :)');
                //successful transaction

            console.log('sender: ', req.user.AccountNum);
            console.log('amount: ', amt);
            console.log('receiver: ', req.body.account);

            var contract = new ethers.Contract(address,abi,wallet); //for setter
function

            var sendPromise = contract.billPayment(req.user.AccountNum, amt,
req.body.account);

            sendPromise.then(function(tx){

            console.log(tx);
            // res.render('fundsTransfer-res.ejs', { name: req.user.name});
   });

          }
        });
        res.render('fundsTransfer-res.ejs', { name: req.user.name})
      }
    }
    else
        console.log(err);
})

})
//END of bill payment's page-------------------------------------


//Funds and Billpayment's response
PAGE*********************************************************

app.get('/fundsTransfer-res', (req, res) => {
  console.log('im in fundsTransfer-res get !!');
  res.render('fundsTransfer-res.ejs', { name: req.user.name})
})

app.post('/fundsTransfer-res', (req, res) => {
  console.log('im in fundsTransfer-res post!!');
  res.render('fundsTransfer-res.ejs', { name: req.user.name})
})

//START order chequebook-------------------------------------
```

```
app.get('/order-chquebook', checkAuthenticated, (req, res) => {
  res.render('order-chquebook.ejs', {name: req.user.name, checkbook_Error: " "});
})


app.post('/order-chquebook', checkAuthenticated, (req, res) => {
  var flag1;
  let sql = 'INSERT INTO checkbook SET ?'

  let post={
    id: req.user.id,
    name: req.user.name,
    AccountNum: req.user.AccountNum,
    Leaves: req.body.Leaves,
    RequestDate: new Date(),
    }
    mysqlConnection.query(sql, post, (err, res) => {
      if(err){
        flag1=1;
        console.log('you already requested a checkbook');//ERROR3 -> (You already
requested a checkbook)
      }
      else{
        flag1=0;
        console.log('you requested checkbook with ' +req.body.Leaves + '
Leaves.');//ERROR4 -> (this is not error just msg for successfull request)

      }

     });
     setTimeout(function () {
      if (flag1){
        res.render('order-chquebook.ejs', {name: req.user.name, checkbook_Error:
"you already requested a checkbook"});

      }
      else{
        //
          //successful transaction
          console.log('sender: ', req.user.AccountNum);
          console.log('amount: ', req.body.Leaves);
          var contract = new ethers.Contract(address,abi,wallet); //for setter
function

          var sendPromise =
contract.storeRequestedCheckbooks(req.user.AccountNum,req.body.Leaves);

          sendPromise.then(function(tx){

          console.log(tx);

        });
        //
        res.render('order-chquebook.ejs', {name: req.user.name, checkbook_Error:
"Successful request"});

      }

      }, 1000);
 //--

});

//END order chequebook---------------------------------------
//START change password-----------------------------------------------------
-----------

app.get('/change-password', checkAuthenticated, (req, res) => {
  res.render('change-password.ejs', {name: req.user.name, passwordAlert: ''});
})

app.post('/change-password', checkAuthenticated, (req, res) => {

  var couldNotChangePass = 0;
  var notMatching = 0;
  var incorrectCurrentPass = 0;

    bcrypt.compare(req.body.CurrentPassword, req.user.password, function(err,
res) {
      console.log('res :' + res);
      if(res === true)
      {
        if(req.body.NewPassword === req.body.ConfirmNewPassword)
        {
          bcrypt_js.hash(req.body.NewPassword, null, null, function(err, hash) {
            console.log('pass = ' + hash);

            let sql_update = 'UPDATE B_users SET password = ? WHERE AccountNum =
?';
            let data = [hash, req.user.AccountNum];

            mysqlConnection.query(sql_update,data, (err, rows, fields) => {
              if (!err)
                console.log('password changed');// alert
              else
                console.log('error in changing password');
                couldNotChangePass=1;

            })

          });
        }
        else
        {
          console.log('passwords are not same');
          notMatching=1;
        }
      }
      else
      {
        console.log('wrong current password');
        incorrectCurrentPass=1;
      }
    });


setTimeout(function () {
  if(incorrectCurrentPass==1){
    res.render('change-password.ejs', {name: req.user.name, passwordAlert:
'Current password is incorrect!'});

  }
  else if(notMatching==1){
    res.render('change-password.ejs', {name: req.user.name, passwordAlert: 'New
password and confirm Password are not matching!'});

  }
  else if(couldNotChangePass==1){
    res.render('change-password.ejs', {name: req.user.name, passwordAlert: 'Error
in changing password!'});

  }
  else{
    res.render('change-password.ejs', {name: req.user.name, passwordAlert:
'Password has been changed successfully!'});
  }

}, 500);
})

//END change password------------------------------------------------------------
---
//START bank-statement-----------------------------------------------------------
---

app.get('/bank-statement', checkAuthenticated, (req, res) => {

  mysqlConnection.query('SELECT * FROM transactions WHERE sender = ?',
[req.user.AccountNum], (err, rows, fields) => {
    if(!err){
      console.log(rows);
      return res.render('bank-statement.ejs', { name: req.user.name,
                                                AccountNum: req.user.AccountNum,
                                                CNIC: req.user.CNIC,
                                                CardStatus: req.user.CardStatus,
                                                Statement: rows});

    }
    else
      console.log('error');
  })

})
```

```
//END bank-statement----------------------------------------------------------
---

//ADMIN PANEL

//Fetching data from blockchain
var txFromBlockchain;
var balFromBlockchain;
var etherValue;
let settings = { method: "Get" };
fetch(urlToGetTransactions, settings)
    .then(res => res.json())
    .then((json1) => {
      txFromBlockchain=json1;
});


let settings1 = { method: "Get" };
fetch(urlToGetBalance, settings1)
    .then(res => res.json())
    .then((json2) => {
      balFromBlockchain=json2;
      etherValue = Web3.utils.fromWei(balFromBlockchain.result, 'ether');

});

app.get('/admin', (req, res) => {

  res.render('admin.ejs', {adminMsg: ''});

})

app.post('/admin', (req,res) => {

  if(req.body.username=='rajat' && req.body.password=='rajatadmin'){
    adminSession = 1;
    res.redirect('/admin_sidebar');
  }
  else{
    adminSession = 0;
    res.render('admin.ejs', {adminMsg: 'Incorrect username or password'});
  }
})

app.get('/admin_sidebar', (req,res) => {
  if(adminSession===1){
    res.render('admin_sidebar.ejs', {Balance: etherValue});
  }
  else{
    res.render('login.ejs');
  }

})

//funds transfer records
app.get('/MoneyRecordsBlockchain', (req,res) => {
  fetch(urlToGetTransactions, settings)
    .then(res => res.json())
    .then((json1) => {
      txFromBlockchain=json1;
});

  const testABI1 =[
    {
      "constant": false,
      "inputs": [
        {
          "name": "sender",
          "type": "string"
        },
        {
          "name": "amount",
          "type": "uint256"
        },
        {
          "name": "receiver",
          "type": "string"
        }
      ],
      "name": "billPayment",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "constant": false,
      "inputs": [
        {
          "name": "account",
          "type": "string"
        },
        {
          "name": "leaves_Requested",
          "type": "uint256"
        }
      ],
      "name": "storeRequestedCheckbooks",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "constant": false,
      "inputs": [
        {
          "name": "sender",
          "type": "string"
        },
        {
          "name": "amount",
          "type": "uint256"
        },
        {
          "name": "receiver",
          "type": "string"
        }
      ],
      "name": "transferFunds",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "inputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "constructor"
    },
    {
      "constant": true,
      "inputs": [],
      "name": "getDetails",
      "outputs": [
        {
          "name": "",
          "type": "string"
        },
        {
          "name": "",
          "type": "uint256"
        },
        {
          "name": "",
          "type": "string"
        }
      ],
      "payable": false,
      "stateMutability": "view",
      "type": "function"
    }
  ]
  abiDecoder.addABI(testABI1);

  if(adminSession===1){
    var count=txFromBlockchain.result.length;
    var count=parseInt(count);
    var FT='<table class="table table-striped" style="width: 100%; padding: 15px;
text-align: left; border-collapse: collapse;"   >';
    FT += '<tr>';
      FT += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'DATE AND TIME'+'</th>';
```

```
      FT += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'SENDER'+'</th>';
      FT += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'AMOUNT'+'</th>';
      FT += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'RECEIVER'+'</th>';
    FT += '</tr>';

    for (i=0;i<count;i++){

      //timestamp

      var date = txFromBlockchain.result[i].timeStamp;
      var date1 = new Date(date*1000);

      //input data
      const testData1 = txFromBlockchain.result[i].input;
      const decodedData1 = abiDecoder.decodeMethod(testData1);
      if(decodedData1!=undefined){
//          console.log('tx: ',decodedData1);
        if(decodedData1.name=="transferFunds"){
          namee= JSON.parse(JSON.stringify(decodedData1.name));
          paramss0=
JSON.parse(JSON.stringify(decodedData1.params[0]));
          paramss1=
JSON.parse(JSON.stringify(decodedData1.params[1]));
          paramss2=
JSON.parse(JSON.stringify(decodedData1.params[2]));

          FT += '<tr style="background-color: #34495E";>';
            FT += '<td style="border: 1px solid #ddd;padding:
8px;">'+date1.toUTCString()+'</td>';
            FT += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss0.value+'</td>';
            FT += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss1.value+'</td>';
            FT += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss2.value+'</td>';
          FT += '<tr>';

        }
      }
      }     FT += '</table>';

    setTimeout(function () {
      res.render('blockchain.ejs', {blkn: FT});
    }, 3000);

  }
  else{
    res.render('login.ejs');

  }

})

//Bill payment records
app.get('/BillRecordsBlockchain', (req,res) => {

  fetch(urlToGetTransactions, settings)
    .then(res => res.json())
    .then((json1) => {
      txFromBlockchain=json1;
});

  const testABI1 =[
    {
      "constant": false,
      "inputs": [
        {
          "name": "sender",
          "type": "string"
        },
        {
          "name": "amount",
          "type": "uint256"
        },
        {
          "name": "receiver",
          "type": "string"
        }
      ],
      "name": "billPayment",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "constant": false,
      "inputs": [
        {
          "name": "account",
          "type": "string"
        },
        {
          "name": "leaves_Requested",
          "type": "uint256"
        }
      ],
      "name": "storeRequestedCheckbooks",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "constant": false,
      "inputs": [
        {
          "name": "sender",
          "type": "string"
        },
        {
          "name": "amount",
          "type": "uint256"
        },
        {
          "name": "receiver",
          "type": "string"
        }
      ],
      "name": "transferFunds",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "inputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "constructor"
    },
    {
      "constant": true,
      "inputs": [],
      "name": "getDetails",
      "outputs": [
        {
          "name": "",
          "type": "string"
        },
        {
          "name": "",
          "type": "uint256"
        },
        {
          "name": "",
          "type": "string"
        }
      ],
      "payable": false,
      "stateMutability": "view",
      "type": "function"
    }
  ]
  abiDecoder.addABI(testABI1);

  if(adminSession===1){
    var count=txFromBlockchain.result.length;
    var count=parseInt(count);
```

```
    var BP='<table class="table table-striped" style="width: 100%; padding: 15px;
text-align: left; border-collapse: collapse;"   >';

    BP += '<tr>';
      BP += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'DATE AND TIME'+'</th>';
      BP += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'SENDER'+'</th>';
      BP += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'AMOUNT'+'</th>';
      BP += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'RECEIVER'+'</th>';
    BP += '</tr>';

    for (i=0;i<count;i++){

      //timestamp
      var date = txFromBlockchain.result[i].timeStamp;
      var date1 = new Date(date*1000);
      // console.log('date: ',date1.toUTCString());

      //input data
      const testData1 = txFromBlockchain.result[i].input;
      const decodedData1 = abiDecoder.decodeMethod(testData1);
      if(decodedData1!=undefined){
//        console.log('tx: ',decodedData1);
        if(decodedData1.name=="billPayment"){

          namee= JSON.parse(JSON.stringify(decodedData1.name));
          paramss0=
JSON.parse(JSON.stringify(decodedData1.params[0]));
          paramss1=
JSON.parse(JSON.stringify(decodedData1.params[1]));
          paramss2=
JSON.parse(JSON.stringify(decodedData1.params[2]));

          BP += '<tr style="background-color: #34495E";>';
            BP += '<td style="border: 1px solid #ddd;padding:
8px;">'+date1.toUTCString()+'</td>';
            BP += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss0.value+'</td>';
```

```
            BP += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss1.value+'</td>';
            BP += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss2.value+'</td>';
          BP += '<tr>';
        }
      }
      }     BP += '</table>';


    setTimeout(function () {
      res.render('blockchain.ejs', {blkn: BP});
    }, 3000);


  }
  else{
    res.render('login.ejs');
  }
})

//checkbook records
app.get('/CheckbookRecordsBlockchain', (req,res) => {

  fetch(urlToGetTransactions, settings)
    .then(res => res.json())
    .then((json1) => {
      txFromBlockchain=json1;
    });

  const testABI1 =[
    {
      "constant": false,
      "inputs": [
        {
          "name": "sender",
          "type": "string"
        },
        {
          "name": "amount",
          "type": "uint256"
        },
        {
          "name": "receiver",
          "type": "string"
        }
      ],
      "name": "billPayment",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "constant": false,
      "inputs": [
        {
          "name": "account",
          "type": "string"
        },
        {
          "name": "leaves_Requested",
          "type": "uint256"
        }
      ],
      "name": "storeRequestedCheckbooks",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "constant": false,
      "inputs": [
        {
          "name": "sender",
          "type": "string"
        },
        {
          "name": "amount",
          "type": "uint256"
        },
        {
          "name": "receiver",
          "type": "string"
        }
      ],
      "name": "transferFunds",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "inputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "constructor"
    },
    {
      "constant": true,
      "inputs": [],
      "name": "getDetails",
      "outputs": [
        {
          "name": "",
          "type": "string"
        },
        {
          "name": "",
          "type": "uint256"
        },
        {
          "name": "",
          "type": "string"
        }
      ],
      "payable": false,
      "stateMutability": "view",
      "type": "function"
    }
  ]
    abiDecoder.addABI(testABI1);
    if(adminSession===1){
      var count=txFromBlockchain.result.length;
      var count=parseInt(count);

      var OC='<table class="table table-striped" style="width: 100%; padding:
15px; text-align: left; border-collapse: collapse;";   >';

      OC += '<tr>';
        OC += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'DATE AND TIME'+'</th>';
        OC += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'ACCOUNT NO.'+'</th>';
        OC += '<th style="border: 1px solid #ddd;padding: 8px; padding-top:
12px;padding-bottom: 12px;background-color: #EA7727;">'+'REQUESTED
LEAVES'+'</th>';
      OC += '</tr>';

      for (i=0;i<count;i++){

        //timestamp
        var date = txFromBlockchain.result[i].timeStamp;
        var date1 = new Date(date*1000);
        // console.log('date: ',date1.toUTCString());

        //input data
        const testData1 = txFromBlockchain.result[i].input;
        const decodedData1 = abiDecoder.decodeMethod(testData1);
        if(decodedData1!=undefined){
  //        console.log('tx: ',decodedData1);
          if(decodedData1.name=="storeRequestedCheckbooks"){

            namee= JSON.parse(JSON.stringify(decodedData1.name));
            paramss0=
JSON.parse(JSON.stringify(decodedData1.params[0]));
            paramss1=
JSON.parse(JSON.stringify(decodedData1.params[1]));

            OC += '<tr style="background-color: #34495E";>';
              OC += '<td style="border: 1px solid #ddd;padding:
8px;">'+date1.toUTCString()+'</td>';
```

```
              OC += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss0.value+'</td>';
              OC += '<td style="border: 1px solid #ddd;padding:
8px;">'+paramss1.value+'</td>';
            OC += '<tr>';
          }
        }
        }     OC += '</table>';


      setTimeout(function () {
        res.render('blockchain.ejs', {blkn: OC});
      }, 3000);


    }
    else{
      res.render('login.ejs');

    }

})

//END OF ADMIN PANEL

var server;
server= app.listen(3000,() => {
  console.log('app is running on 3000 port');
})

// contract hashing123456 {
//    let timestamp = timeStamp;

//    let hashes = txFromBlockchain.BigNumber

//    let salt_hash = "f24325481675565cfa74257304338239 24"; // or take any random
value
//     public requested_hash;

//     function hashed123456() public {
//       requested_hash = msg.sender;
//    }

// function show_ public {
//    let combination_hash = generate_bycrpt_of_hashes(hashes, msg.sender)
//    let hashedvalue = crypt_hash(combination_hash, timestamp, salt_hash);
// }
```

SmartContract.sol:

```
pragma solidity ^0.4.24;
contract transaction{

    string sender_acc;
    uint amount;
    uint leaves;
    string receiver_acc;

    constructor() public{
        sender_acc="";
        amount=0;
        receiver_acc="";
    }

    function transferFunds(string s, uint a, string r) public{
        sender_acc=s;
        amount=a;
        receiver_acc=r;
    }

    function storeRequestedCheckbooks(string s, uint l) public{
        sender_acc=s;
        leaves=l;
    }

    function getDetails() public view returns(string , uint , string){
        return (sender_acc,amount,receiver_acc);
    }

}
```

REFERENCES

I hereby affirm that I have not cheated or copied for this project. I have obtained help from the following sources:

- https://medium.com/coinmonks/hello-world-smart-contract-using-ethers-js-e33b5bf50c19
- https://dev.to/isalevine/three-ways-to-retrieve-json-from-the-web-using-node-js-3c88
- https://stackoverflow.com/questions/42558090/how-to-create-html-table-based-on-json
- https://www.youtube.com/watch?v=WPPni-pufok&list=PLsyeobzWxl7oY6tZmnZ5S7yTDxyu4zDW-&index=18
- https://www.udemy.com/course/the-complete-web-developer-zero-to-mastery/