ONLINE BANKING SYSTEM USING BLOCKCHAIN

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

B.Tech

in

Computer Science & Engineering

[](http://www.google.co.in/url?url=http://egov.engineeringwatch.in/bhagwan-parshuram-institute-of-technology-new-delhi-north-west-rohini/&rct=j&frm=1&q=&esrc=s&sa=U&ei=Qcc4VaDfFcOUsAHi54DwDg&ved=0CBUQ9QEwAA&sig2=VGYgGxagdSfRMxejTdK0Yg&usg=AFQjCNFN993UjYXRhdtuExrXVrIGKtxPmw)

Under the supervision

of

Akansha Dhamija

Associate 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 ,41520802718**

**Navneet ,41820802718**

**Avnish ,4280802718**

**Pushan ,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.

  (Signature)

**Date: 27-12-2021 Supervisor**

**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**

**ACKNOWLEDGEMENT**

(Signature of the students)

Table of Contents

List of Figures (with page numbering in ROMAN)

List of Tables (with page numbering in ROMAN)

Abstract (with page numbering in ROMAN)

Page No

Chapter 1 : Introduction……………………………………..

Chapter 2 : Related Work……………………………………

Chapter 3 : System Analysis and Design

3.1 : Software Requirement Specifications (H/W and S/W

requirements)

3.2 : Use Case Diagrams

3.3 : Flow Chart/ Activity Diagram

Chapter 4 : Proposed Work (Your Algorithm )

Chapter 5 : Implementation and Results (50 % implementation)

Chapter 6 : Conclusion

Chapter 7 : Future Work

REFERENCES

**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.

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

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

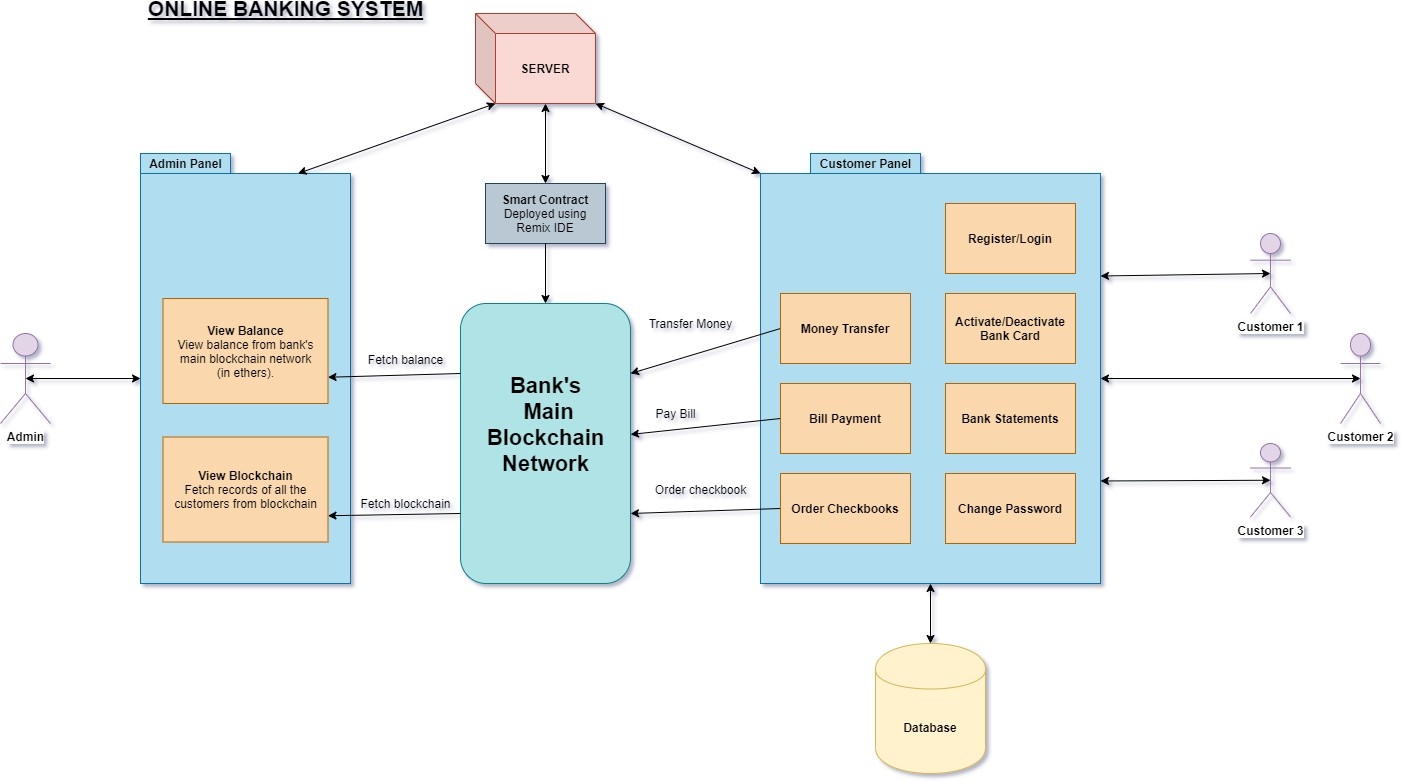
with bad intentions somehow accesses the database or all the server files, even then,

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).

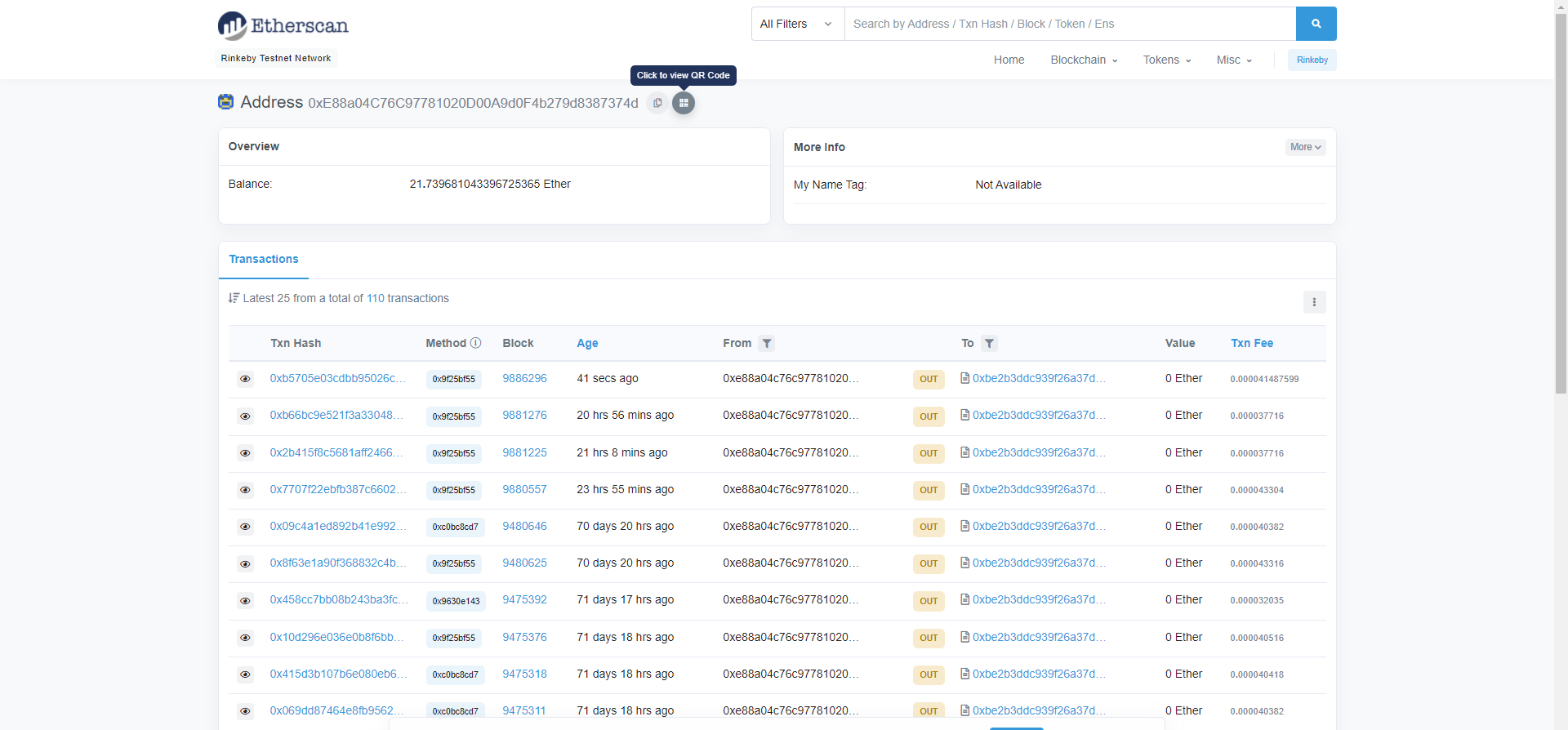
**Architecture and Design:**



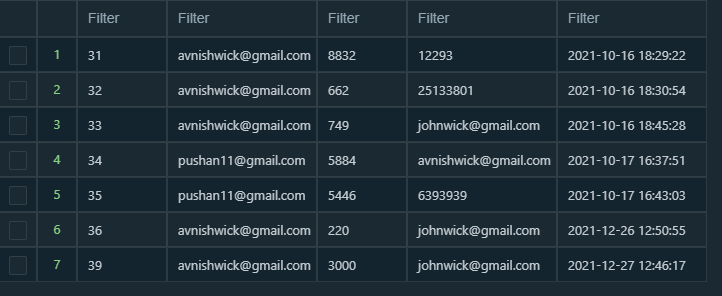
**Results:**

**Saving The Records To The Blockchain:**

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

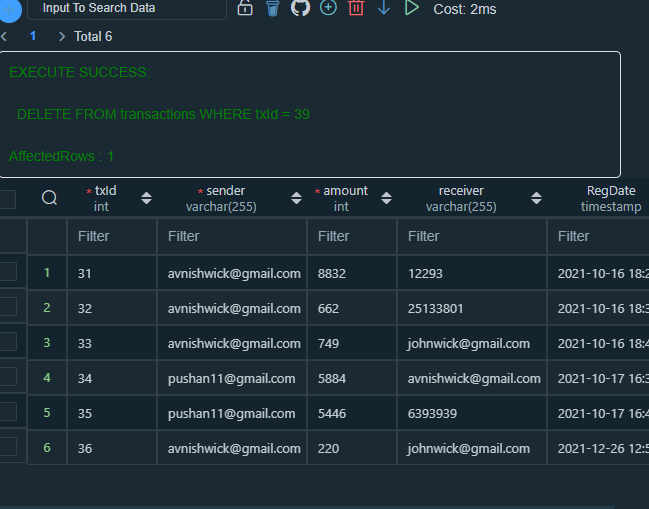


Saving transaction to a database is much faster but the overhead of saving to the blockchain is valuable.



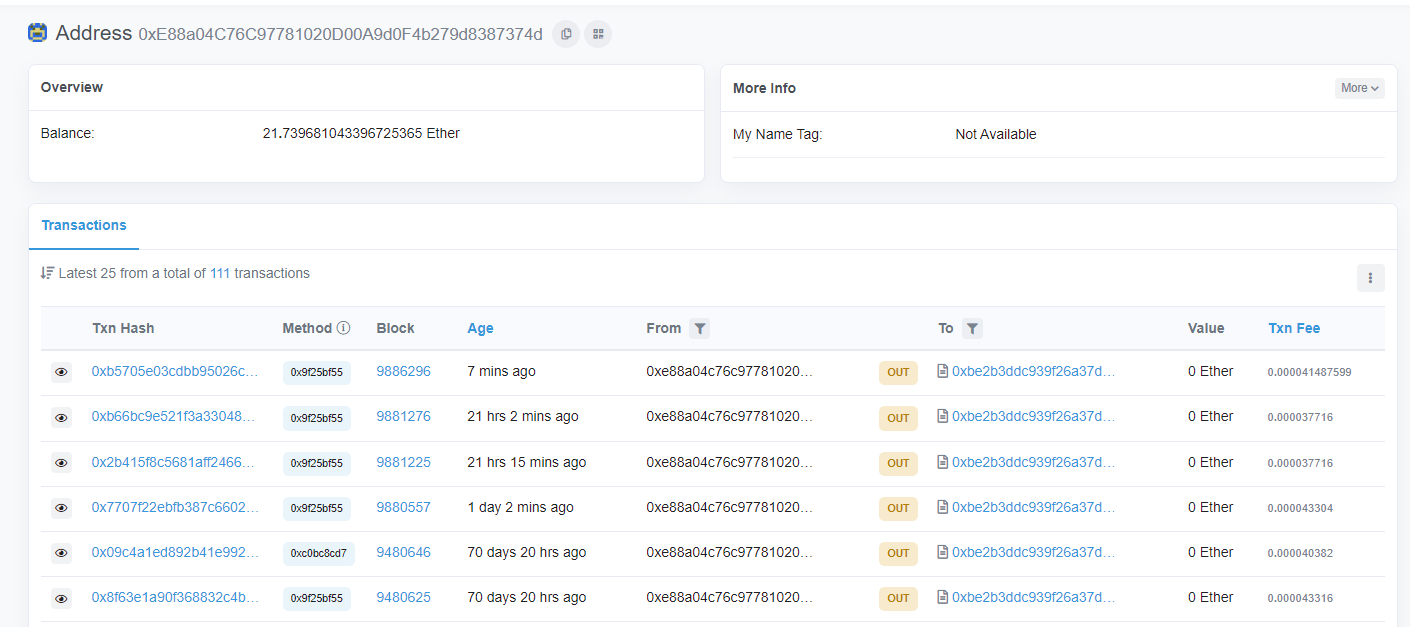
Deleting A Record:

Deleting a record from database.



Record has been deleted.

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



We can clearly view it from the admin panel.



**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.

**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=0xE88a04C76C97781020D00A9d0F4b279d8387374d&startblock=0&endblock=99999999&sort=asc&apikey=NVB7ZC1WEES9RP7ZMZ2AHWHTYWNKH8KN2B";

var urlToGetBalance**=**"https://api-rinkeby.etherscan.io/api?module=account&action=balance&address=0xE88a04C76C97781020D00A9d0F4b279d8387374d&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 = "f2432548167565cfa7425730433823924"; // 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);*

*// }*

*// }*

**Honour Statement:**

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/>