CS482 Introduction to Block chain and Cryptocurrency

PROJECT REPORT

Online Banking System

Instructor	**Sir Jawwad Shamsi**
Project Team	**Fatima Ibrahim (17K-3639)** **Hina Waheed (17K-3862)**

Department Of Computer Science

FAST-National University of Computer Emerging Sciences, Karachi.

Table of Contents

1. Introduction:

Online banking has considerable amount of commercial significance. No matter the existing no. of banks, online banking will always be a need. Users always want 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:

- Registration/Login
- Debit card activation/deactivation
- Money Transfer
- Bill Payments
- Generate bank statements
- Order checkbooks
- Change password
- View bank details

2. Motivation:

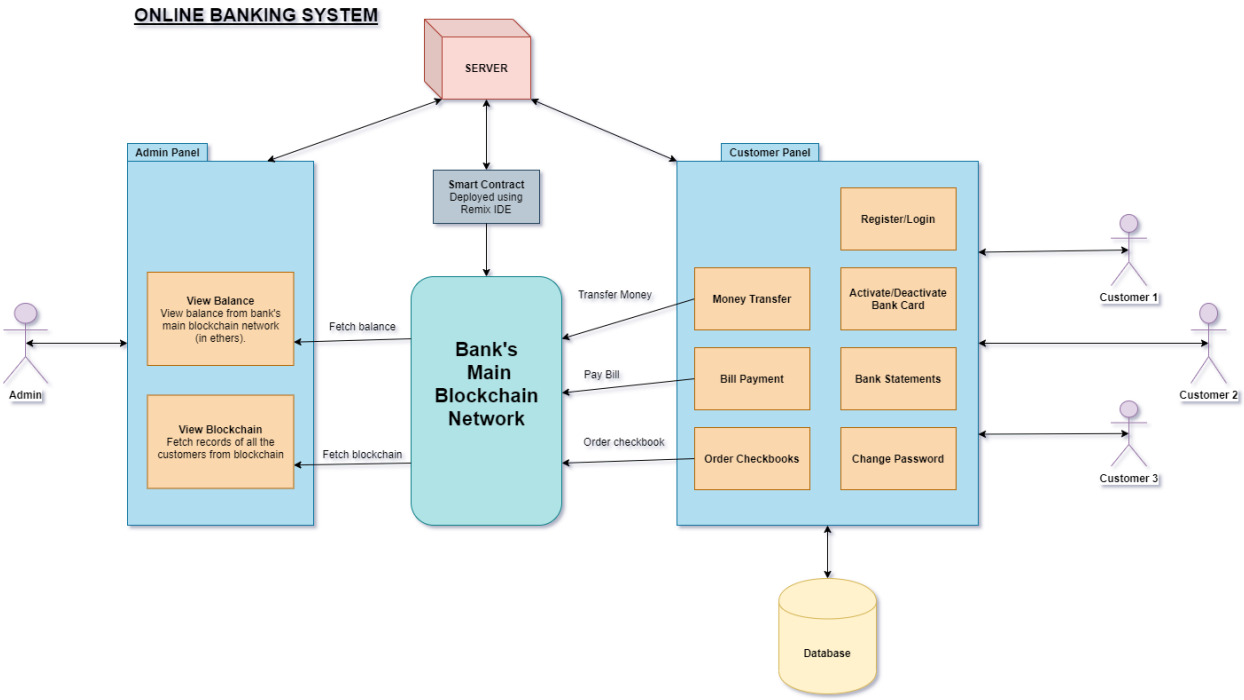
The biggest problem faced by online systems handling payments or taking orders is the security. Whenever we perform a transaction using such systems, we want its record to be immutable. We want to save our transactions at a decentralized place because that way, we would not need to trust any single provider.
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 get all of the different types of transactions of all the customers directly from the blockchain network.

Security is maintain 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 admin panel).

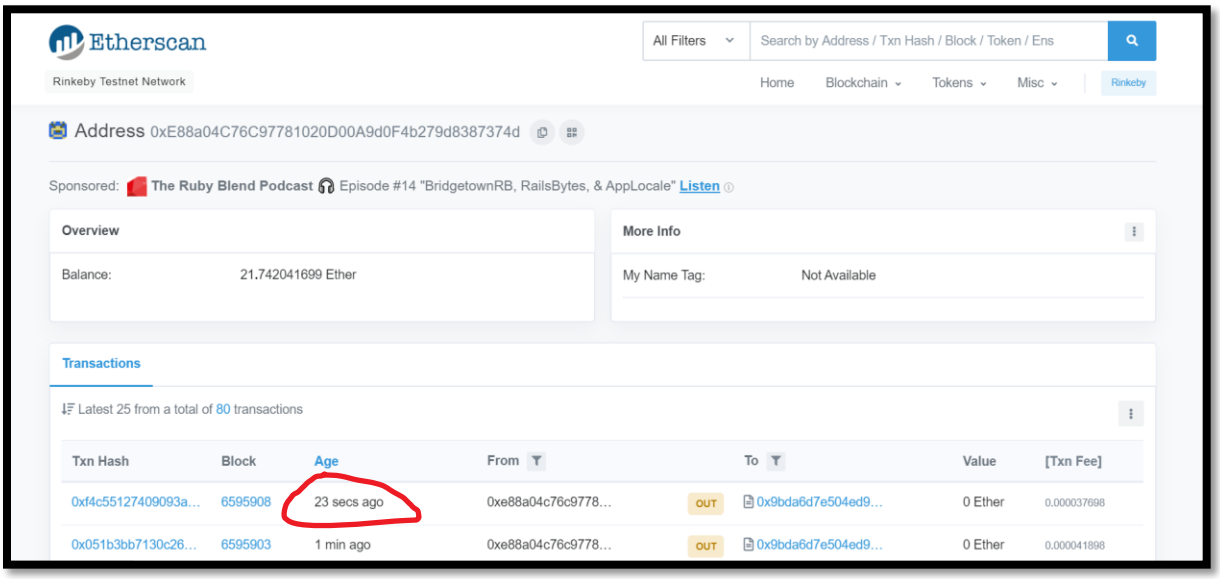
3. Architecture and Design:



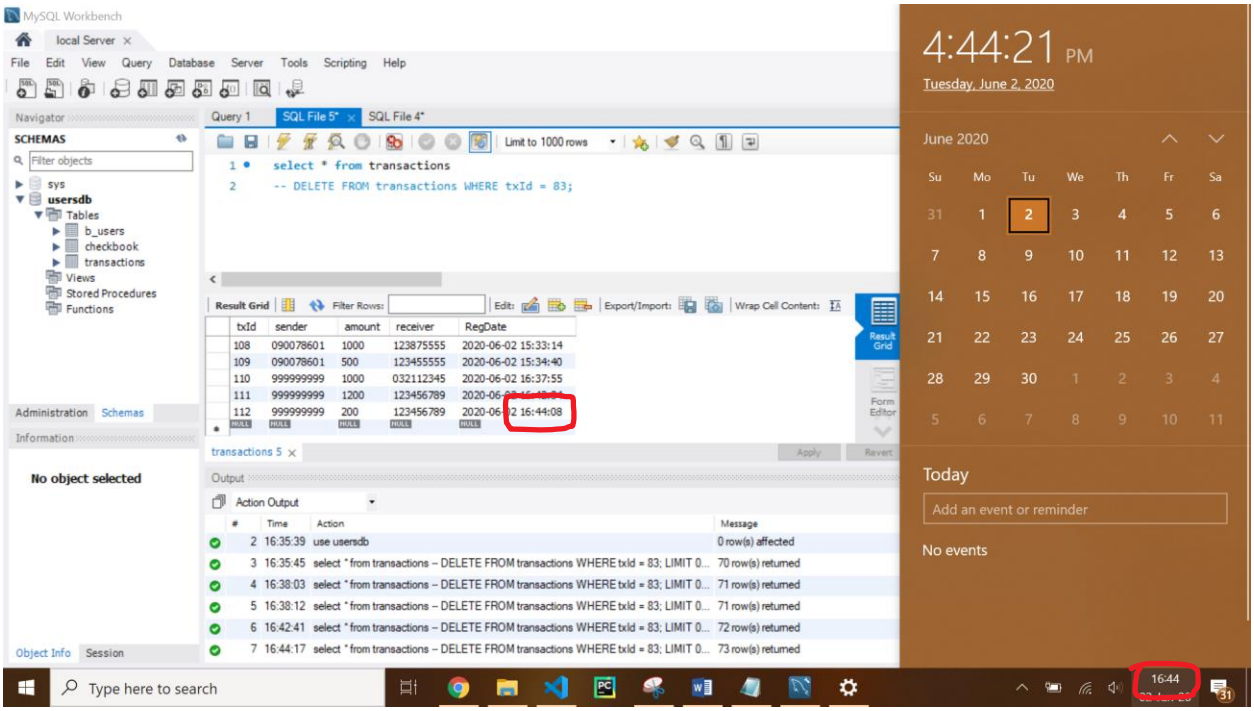
4. Results:

Saving The Records To The Blockchain:

- It is approximately taking 23 seconds to save a transaction on Ethereum's test network.

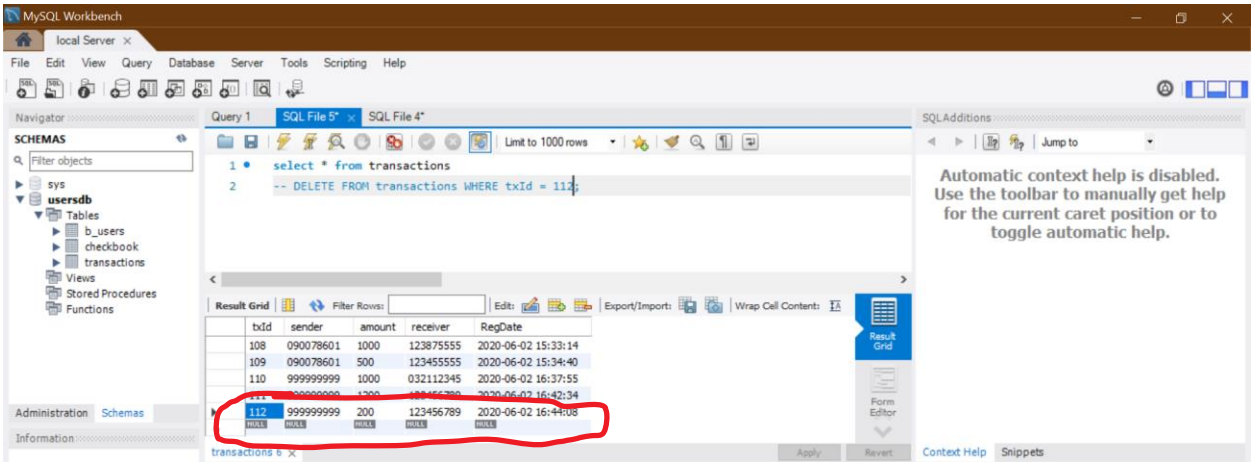


- Saving transaction to a database is much faster but the overhead of saving to the blockchain is valuable.

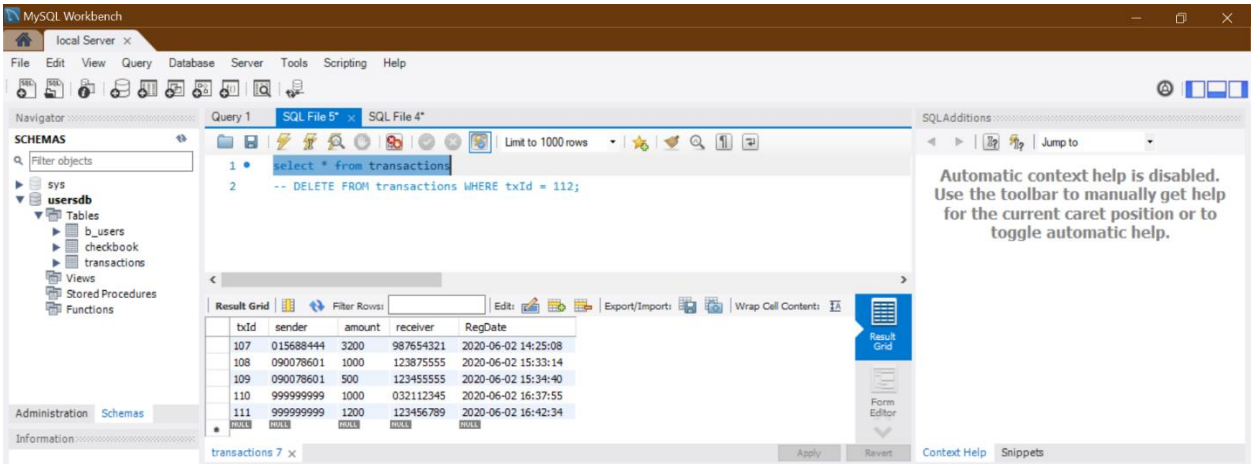


Deleting A Record:

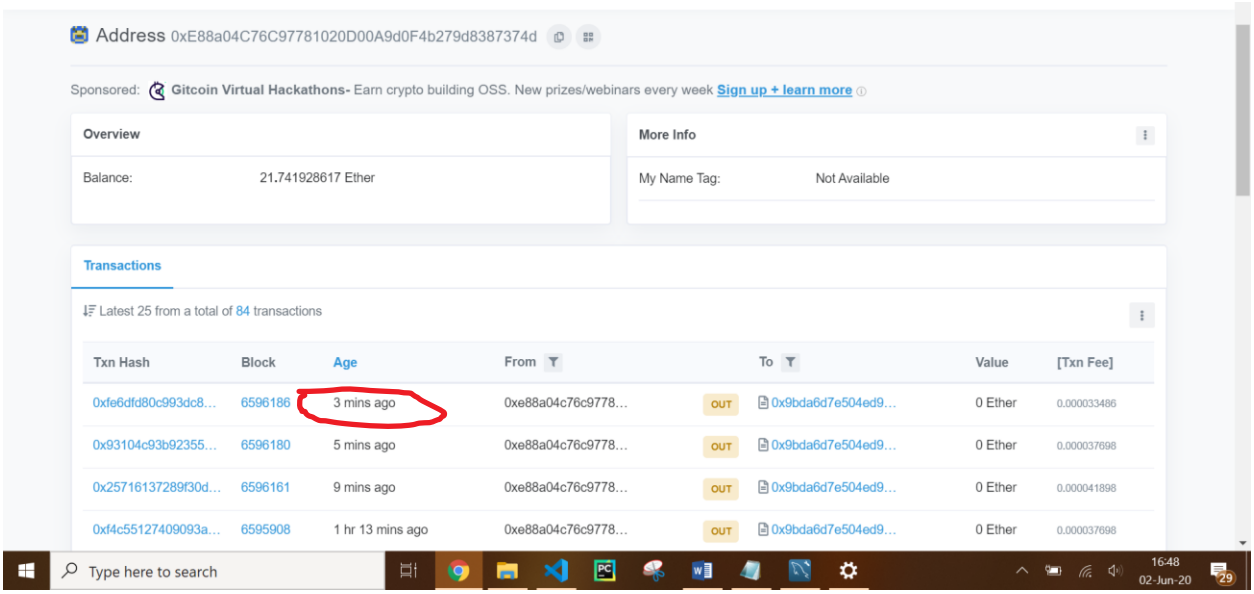
- Deleting a record from database.



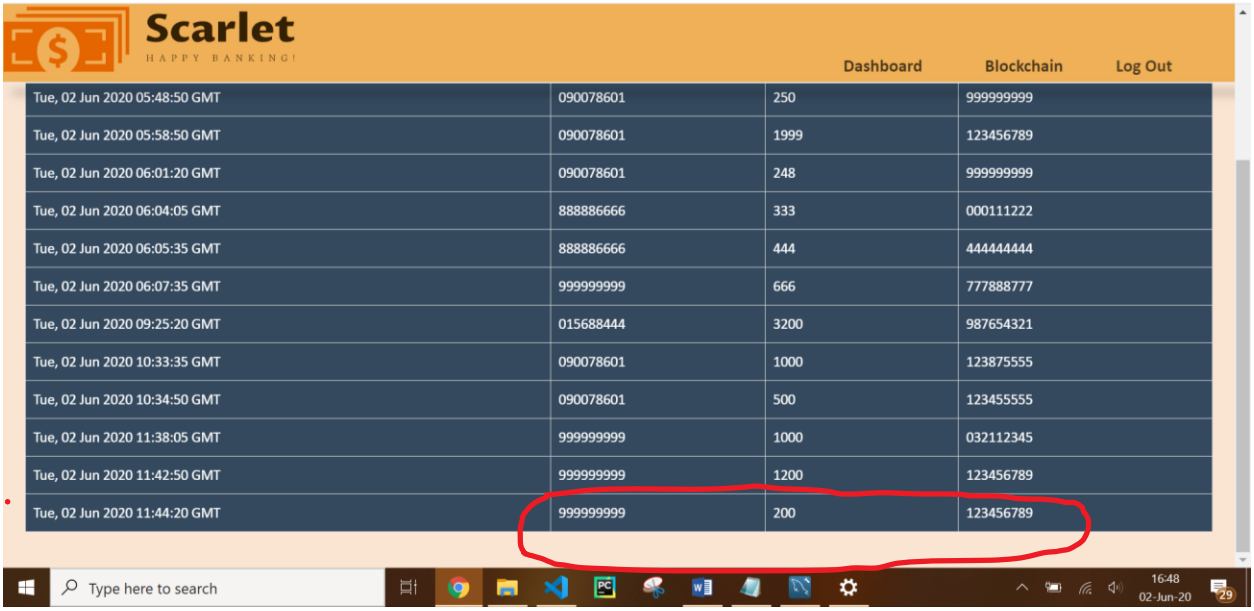
- Record has been deleted.



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



- We can clearly view it from the admin panel.



Tue, 02 Jun 2020 05:48:50 GMT	090078601	250	999999999
Tue, 02 Jun 2020 05:58:50 GMT	090078601	1999	123456789
Tue, 02 Jun 2020 06:01:20 GMT	090078601	248	999999999
Tue, 02 Jun 2020 06:04:05 GMT	888886666	333	000111222
Tue, 02 Jun 2020 06:05:35 GMT	888886666	444	444444444
Tue, 02 Jun 2020 06:07:35 GMT	999999999	666	777888777
Tue, 02 Jun 2020 09:25:20 GMT	015688444	3200	987654321
Tue, 02 Jun 2020 10:33:35 GMT	090078601	1000	123875555
Tue, 02 Jun 2020 10:34:50 GMT	090078601	500	123455555
Tue, 02 Jun 2020 11:38:05 GMT	999999999	1000	032112345
Tue, 02 Jun 2020 11:42:50 GMT	999999999	1200	123456789
Tue, 02 Jun 2020 11:44:20 GMT	999999999	200	123456789

5. 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.

Appendix A:

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');
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=0xE88a04C76C977810
20D00A9d0F4b279d8387374d&startblock=0&endblock=99999999&sort=asc&apikey=NV
B7ZC1WEES9RP7ZMZ2AHWHTYWNKH8KN2B";
var urlToGetBalance="https://api-
rinkeby.etherscan.io/api?module=account&action=balance&address=0xE88a04C76C977
81020D00A9d0F4b279d8387374d&tag=latest&apikey=NVB7ZC1WEES9RP7ZMZ2AHWHT
YWNKH8KN2B";

//SETTING UP CONNECTION WITH SMART CONTRACT and BLOCKCHAIN
var address='0x9BDA6d7e504eD98B1cbda3A338668d54763e5c0c'; //my contract's addr
ess
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='EE79F1C4B628A7D6C85FFCA5735D086FA6B0B71AFF23277F0A9253612
CEA58DC'; //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: 'fire1996',
    database: 'usersdb',
    port: '3306',
    multipleStatements: true
});

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

//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, Bal
ance: 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)
    {
        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.bo
dy.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_s
tatus: 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, car
d_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.Ac
countNum], (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.bod
y.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 Acc
ountNum = ?';
            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 n
ot 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.accou
nt);

  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.Ac
countNum], (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.acc
ount);

          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.');//ERROR
4 -> (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 chec
kbook"});

     }
     else{

        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 Passw
ord 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 succe
ssfully!'});
  }

}, 500);
})

//END change password-----------------------------------------------------------

//START bank-statement----------------------------------------------------------

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

  mysqlConnection.query('SELECT * FROM transactions WHERE sender = ?', [req.user.Acc
ountNum], (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=='admin' && req.body.password=='0000000'){
    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()+'</t
d>';
           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.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="";
      leaves=0;
   }

   function transferFunds(string sender, uint amount, string receiver) public{
      sender_acc=sender;
      _amount=amount;
      receiver_acc=receiver;
   }

   function billPayment(string sender, uint amount, string receiver) public{
      sender_acc=sender;
      _amount=amount;
      receiver_acc=receiver;
   }

   function storeRequestedCheckbooks(string account, uint leaves_Requested) public{
      sender_acc=account;
      leaves=leaves_Requested;
   }

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

Appendix B:

Honour Statement:

I hereby affirm that I have not cheated or copied for this project. I have obtained help from 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/