A Project Report On

Blockchain Technology (LP3)

CLASS: BE-1

GUIDED BY
Prof. Shweta Shah

DEPARTMENT OF COMPUTER ENGINEERING
PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DHANKAWADI, PUNE-43

SAVITRIBAI PHULE PUNE UNIVERSITY
2022-23

Gayatri Godbole – 41128
Pushkar Jain – 41134

Title:

D-app for E-Voting System

Problem Definition:

Develop a Blockchain based application dApp for e-voting system.

Objective

1) To understand and explore the working of Blockchain technology and its applications.

2) To create a blockchain application d-App for e-voting system.

Requirements:

An Ubuntu 20.04 environment

Node

Ethereum

Metamask

Google Chrome

Theory

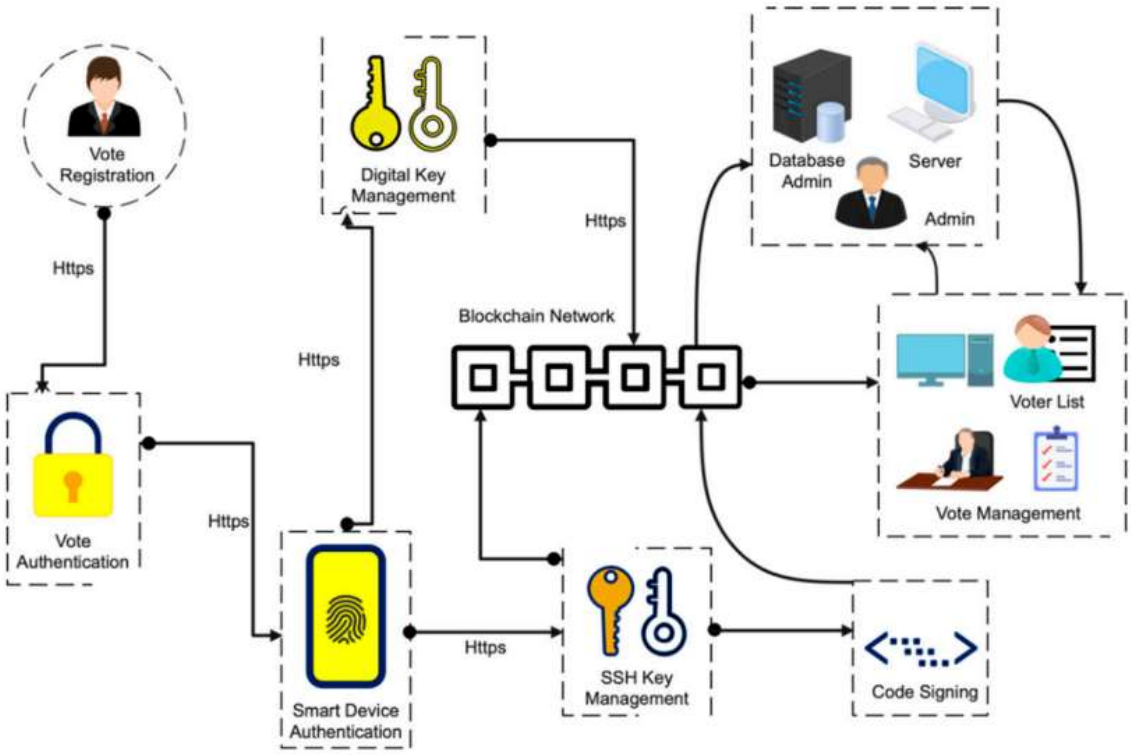
Blockchain works by creating decentralized distributed ledgers that are circulated over all devices participating in the system. It allows to share records based on peer-to-peer replication, and processing by all nodes in the network namely, transacting nodes and validating nodes. When records are placed in the ledger, all nodes in the network go through encryption procedures and are processed by all miners. The blockchain provides highly reliable storage
of data and makes use of consensus strategy, digital signatures, and hash of each block.

Blockchain provides many features such as integrity, traceability, safety, and security for protection of data. Blockchain is applied in both private and government sectors. Usually, an organization has a database where it stores information about their employees and workers. The system will use information from this database so that we can have a list of valid voters and valid candidates in our system the system flowchart for user casting the vote. The user would be directed to a login portal where the user would need to enter some credential to authenticate the user. The user interface will be a web application. Once the user enters the credentials correctly, i.e., the user is successfully able to log in into system, the user would be directed to the voting portal.

In the voting portal, there would be a list of candidates. The voter needs to give preference to the candidates with the lowest number given to the most preferred candidate and the largest number given to the least preferred candidate. In our system , each preferred list by a voter is taken as one transaction.

Thus whenever a user or voter gives preference to the candidates and submit his list, a transaction is added to the blockchain network.

Thus, each voter would be able to cast only one vote.



Description :

- The authority must login first with the provided session ID.
- The voter can now begin the process of voting with proper authentication through OTP(one time password) on the respective linked mobile number.
- If the voter is valid then the system will check for for the voters age and the address to which he can give vote.
- the voting pallete will be opned with candidate names,their parties and logos.
- Now the voter can give his vote by clicking vote button.
- one voter can give his vote only once,i.e after one time voting buttons are disabled and the vote is automatically loged out.
- Same process continiues for many more votters irrespective of their voting wards.

Installing and Running Project

Clone Project

git clone git@github.com:sanattaori/techdot.git && cd techdot
Install Dependencies

npm install
Running Project

node index.js
If dependency problem occurs delete package.json, Run

npm init
Again Install dependencies and run project.

Running Project

Step 1 - Setting up Environment Instead of developing the app against the live Ethereum blockchain, we have used an in-memory blockchain (think of it as a blockchain simulator) called testrpc.

npm install ethereumjs-testrpc web3
Step 2 - Creating Voting Smart Contract

npm install solc

Step 3 - Testing in node console
Code:-

Index.js

@@ -6,13 +6,17 @@ var passwordHash = require('password-hash');

var bodyParser = require('body-parser');

var cookieParser = require('cookie-parser');

var request = require('request');

var fs = require('fs');

Web3 = require('web3')

solc = require('solc')

var app = express();

app.use(bodyParser.json())

app.use(cookieParser());

app.use(morgan('combined'));

app.use("/", express.static("ui"));

@@ -59,10 +63,11 @@ app.get('/app', function(req, res){

```
        var cookie_otp = req.cookies['show'];
        if (passwordHash.verify('password', cookie_pass) && cookie_otp != null) {
        res.sendFile(path.join(__dirname, 'ui', 'clist.html'));
        //res.sendFile(path.join(__dirname, 'ui', 'clist.html'));
        res.redirect('/info');
        } else if (cookie_otp == null || cookie_otp == '') {
        } else if (cookie_otp == null && passwordHash.verify('password',
cookie_pass)) {
        res.sendFile(path.join(__dirname, 'ui', 'app.html'));
        }
        else {
@@ -71,12 +76,27 @@ app.get('/app', function(req, res){
});
// app.post('/getaddress',function(req,res){
// });
app.get('/info', function(req, res){
        var cookie_pass = req.cookies['auth'];
        var cookie_otp = req.cookies['show'];
        if (cookie_pass == null || cookie_pass == '' || cookie_otp == null || cookie_otp
== '') {
        res.redirect('/app');
        } else {
        web3 = new Web3(new
Web3.providers.HttpProvider("http://localhost:8545"));
        code = fs.readFileSync('Voting.sol').toString()
        compiledCode = solc.compile(code)
        abiDefinition = JSON.parse(compiledCode.contracts[':Voting'].interface)
```

```
        VotingContract = web3.eth.contract(abiDefinition)

        byteCode = compiledCode.contracts[':Voting'].bytecode

        deployedContract =
VotingContract.new(['Sanat','Aniket','Mandar','Akshay'],{data: byteCode, from:
web3.eth.accounts[0], gas: 4700000})

        contractInstance = VotingContract.at(deployedContract.address)

        res.sendFile(path.join(__dirname, 'ui', 'clist.html'));

        }

});

        var port = 8080;

        app.listen(8080, function () {

        console.log(`app listening on port ${port}!`);

        });
```

Package.json

```
{

"name": "techdot",

"version": "1.0.0",

"description": "",

"main": "index.js",

"dependencies": {

"cookie-parser": "^1.4.5",

"express": "^4.16.3",

"morgan": "^1.10.0",

"password-hash": "^1.2.2",

"request": "^2.88.2",

"solc": "^0.4.11",
```

"web3": "^0.19.1"

},

"devDependencies": {},

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"repository": {

"type": "git",

"url": "git+https://github.com/sanattaori/techdot.git"

},

"author": "",

"license": "ISC",

"bugs": {

"url": "https://github.com/sanattaori/techdot/issues"

},

"homepage": "https://github.com/sanattaori/techdot#readme"

}

Solidity Program :

```
pragma solidity ^0.4.11;
// We have to specify what version of compiler this code will compile with
contract Voting {
/* mapping field below is equivalent to an associative array or hash.
The key of the mapping is candidate name stored as type bytes32 and value is
an unsigned integer to store the vote count
```

```
*/
mapping (bytes32 => uint8) public votesReceived;
/* Solidity doesn't let you pass in an array of strings in the constructor (yet).
We will use an array of bytes32 instead to store the list of candidates
*/
bytes32[] public candidateList;
/* This is the constructor which will be called once when you
deploy the contract to the blockchain. When we deploy the contract,
we will pass an array of candidates who will be contesting in the election
*/
function Voting(bytes32[] candidateNames) {
candidateList = candidateNames;
}
// This function returns the total votes a candidate has received so far
function totalVotesFor(bytes32 candidate) returns (uint8) {
if (validCandidate(candidate) == false) throw;
return votesReceived[candidate];
}
// This function increments the vote count for the specified candidate. This
// is equivalent to casting a vote
function voteForCandidate(bytes32 candidate) {
if (validCandidate(candidate) == false) throw;
votesReceived[candidate] += 1;
}
function validCandidate(bytes32 candidate) returns (bool) {
```

```
for(uint i = 0; i < candidateList.length; i++) {
if (candidateList[i] == candidate) {
return true;
}
}
return false;
}
}
```

ui/js/app.js :

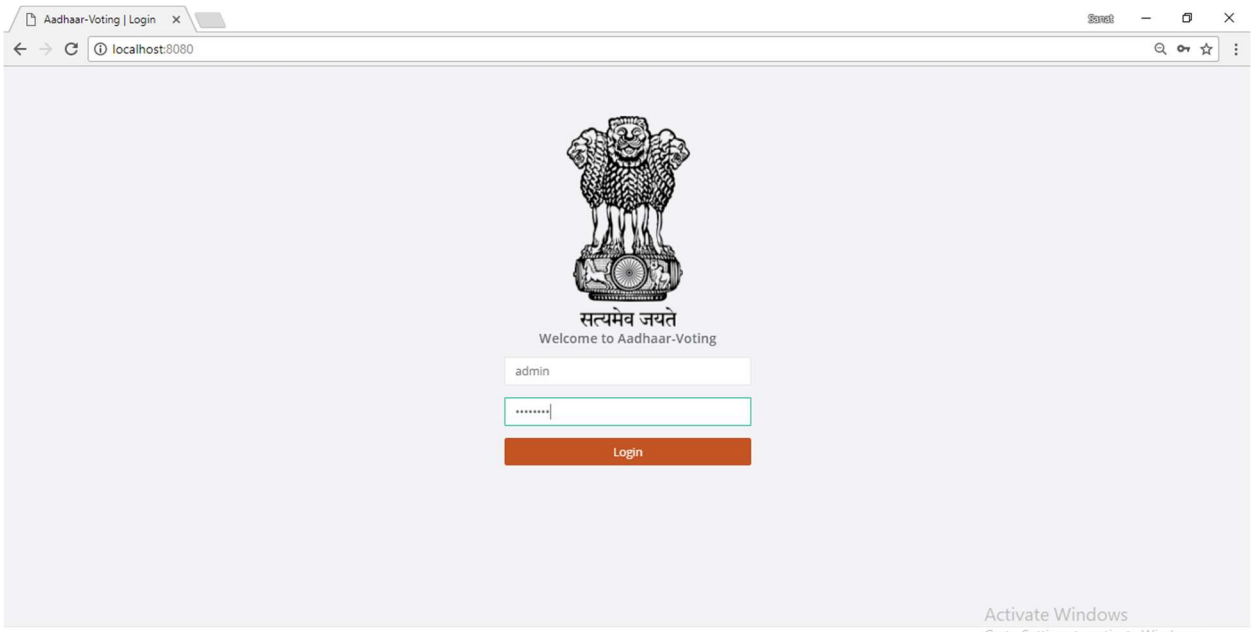
```
@@ -34,8 +34,9 @@ $('#errorbox').hide()
                });
        var aadhaar_no_phone_no = {
                "738253790005": "9158018030",
                "300000000000": "7276478489"
                "7382537xxxxx": "915801xxxx",
                "300000000000": "7276xxxxxx",
                "<replace your aadhaar no here>": "<your phone number>",
                }
                function onSignInSubmit() {
                window.signingIn = true;
                $('#errorbox').hide();
                // updateSignInButtonUI();
                var phoneNumber = "+91" + aadhaar_no_phone_no[$('#aadhaar_no').val()];
                //console.log(phoneNumber);
                var d = new Date();
```

```
                d.setTime(d.getTime() + (1*24*60*60*1000));
                var expires = "expires="+ d.toUTCString();
                document.cookie = 'aadhaar' + "=" + $('#aadhaar_no').val() + ";" + expires +
";path=/";

                $('#verifyc').text('Enter verification code send to '+phoneNumber)
                var appVerifier = window.recaptchaVerifier;
                firebase.auth().signInWithPhoneNumber(phoneNumber, appVerifier)
                .then(function (confirmationResult) {

                // SMS sent. Prompt user to type the code from the message, then sign the
                // user in with confirmationResult.confirm(code).
                window.confirmationResult = confirmationResult;
                window.signingIn = false;
// updateSignInButtonUI();
                // $('.verification-code-form').show()
                // $('#hidepf').hide()
                $('#enter_aadhaarno').hide()
                $('#verify_otp_model').show()
                console.log('otp');

                }).catch(function (error) {
                // Error; SMS not sent
                // $('.main_loader').hide()

                //console.error('Error during signInWithPhoneNumber', error);
```

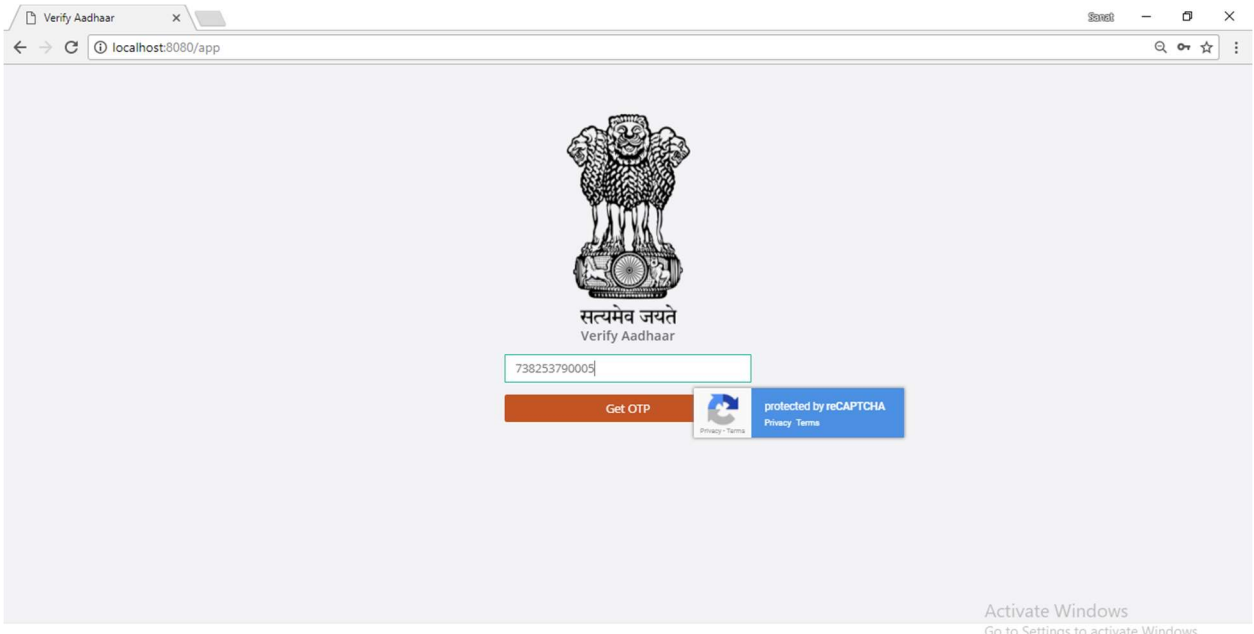
```
                window.alert('error\n\n'+error);
                window.signingIn = false;
                //updateSignInFormUI();
                //updateSignInButtonUI();
                $('.verification-code-form').hide()
                });
                }
                // Phone auth end //

$(verifyotp).click(function(){
                var code = $('#verify_otp').val()
                confirmationResult.confirm(code).then(function (result) {
                // User signed in successfully.
                var user = result.user;
                window.verifyingCode = false;
                //login success
                console.log(user.uid);
                var d = new Date();
                d.setTime(d.getTime() + (1*24*60*60*1000));
                var expires = "expires="+ d.toUTCString();
                document.cookie = 'show' + "=" + user.uid + ";" + expires + ";path=/";
                window.location = '/info'

                }).catch(function (error) {
                // User couldn't sign in (bad verification code?)
```

```
            console.error('Error while checking the verification code', error);
            window.alert('Error while checking the verification code:\n\n'
            + error.code + '\n\n' + error.message);
        window.verifyingCode = false;
            $('#errorbox').show()
            $('#error').text('Enter valid OTP')
            });
            });
            $(getotp).click(function(){
            if ($('#aadhaar_no').val()=="") {
            $('#errorbox').show()
            $('#error').text('Please Enter Aadhaar No')
            }
            else{
            onSignInSubmit();
            $('#errorbox').hide()
            }
            });
```

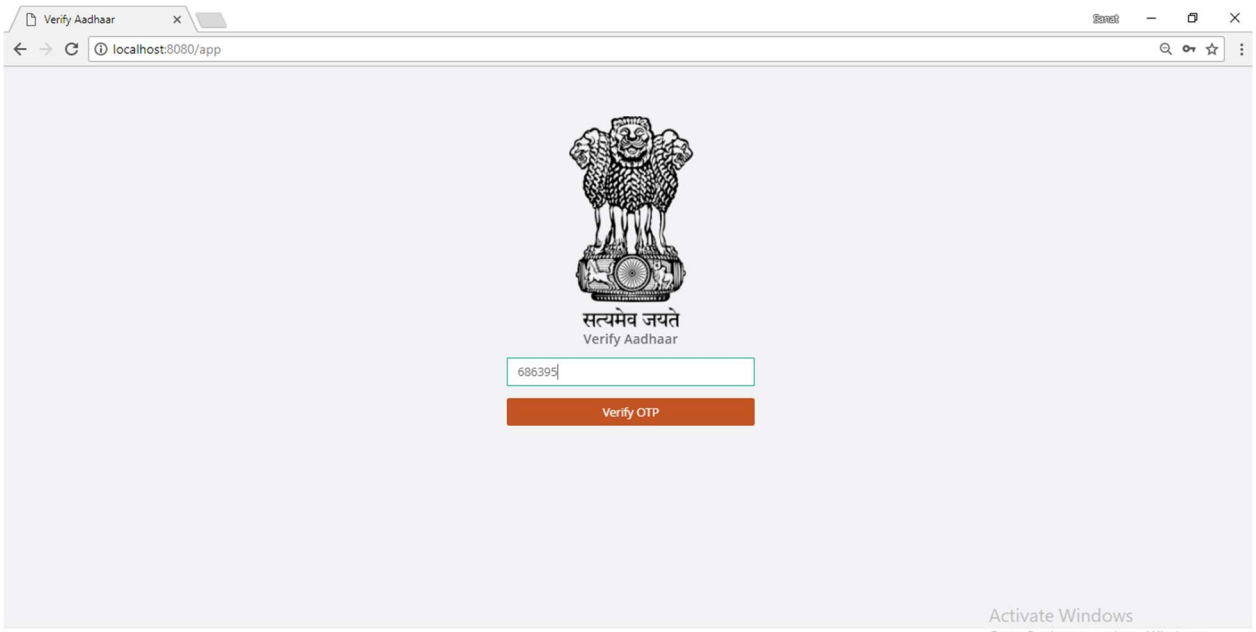
OUTPUT :



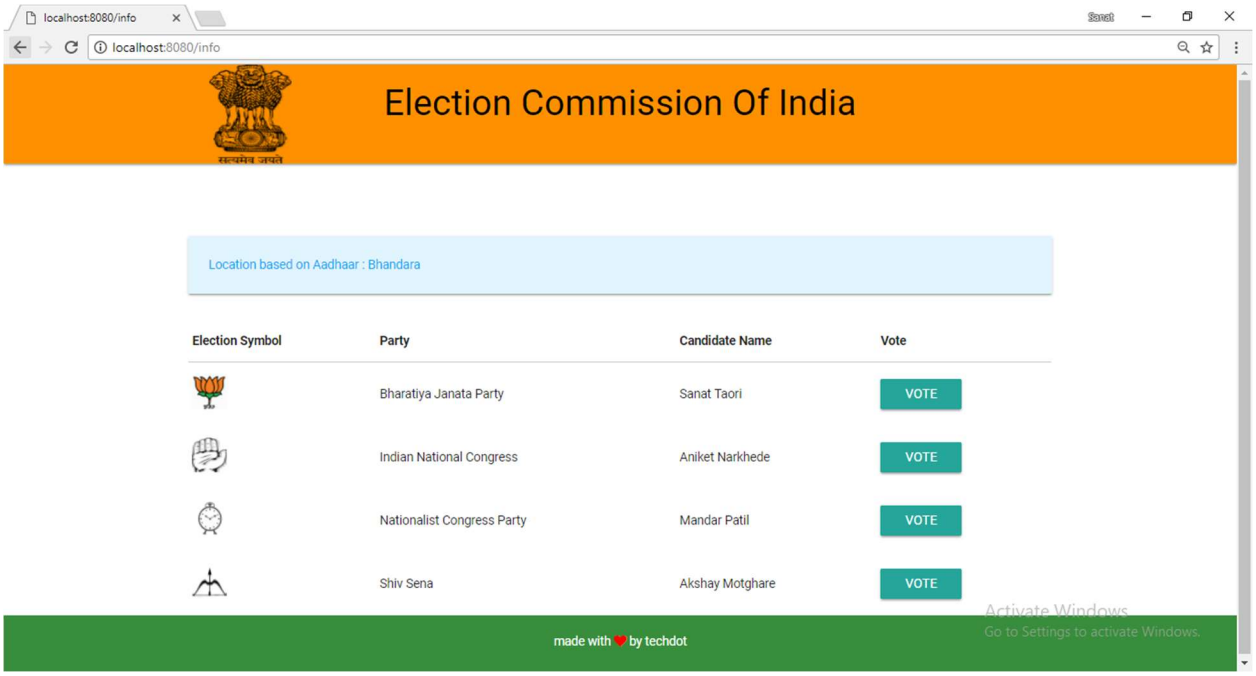
Made with ♥ by Techdot

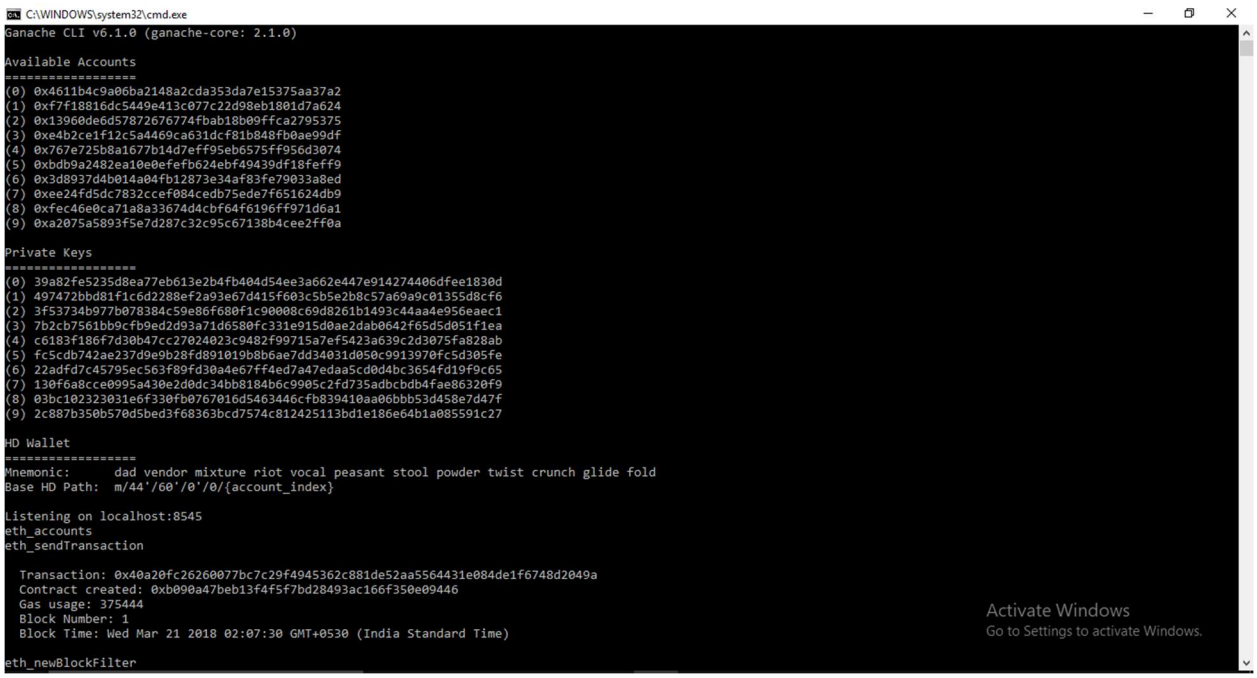


Made with ♥ by Techdot



Made with ❤ by Techdot





Conclusion :

Our voting system uses the concept of preference based voting, where instead of giving vote to particular candidate, the voters provide with preference for the candidates. The votes are counted, and if we have a clear majority as defined by the organization, then our system will declare the winner. But if we don't have clear winner, the votes of the last candidate are distributed according to the preferences given by the voters for that candidate.Thus, the last candidate is removed from the competition and its votes distributed to other candidates according to the preferences given in the votes of the eliminating candidates. This process continues as long as we don't get a clear majority winner. Ethereum platform has been used which runs smart contract on it and is known to be one of the immensely secured blockchain platform.