**COMSATS University Islamabad,   
Park Road, Chak Shahzad, Islamabad Pakistan**

**SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)**

**for**

**BitVote a Block chain Voting System**  
Version 1.0

***By***

**SYED SAQLAIN SHAH CIIT/SP16-BSE-016/ISB**

**MUHAMMAD HASNAT CIIT/SP16-BSE-023/ISB**

***Supervisor*Mr. QASIM MALIK**

***Bachelor of Science in Computer Science (2016-2020)***

**Table of Contents**

**Revision History iii**

**1.** **Introduction 1**

**2.** **Design Methodology and software process model 2**

**3.** **System Overview 2**

3.1 Architectural Design 3

3.2 Process Flow/Representation 4

**4.** **Design Models 7**

**5.** **Data Design 7**

5.1 Data Dictionary 9

**6.** **Algorithm & Implementation 9**

**7.** **Software Requirements Traceability Matrix 13**

**8.** **Human Interface Design 14**

8.1 Screen Images 14

8.2 Screen Objects and Actions 14

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |
|  |  |
|  |  |

**Supervised by**

**Mr. Qasim Malik**

Signature\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction**

In today’s world, widespread mistrust towards the government and interference in countries ‘processes by external actors have made the democratic process of voting more critical than ever. Democratic countries have been experiencing dictatorial regimes which have introduced widespread terror among their people. People have had their human rights violated and their fundamental freedoms provided by their constitution taken away. In such an atmosphere, having a fair and transparent election is something that is paramount for the freedom most people enjoy today.

The pitfalls of the current system of ballot voting are being taken advantage of by people or organizations looking to gain power. In the African countries of Uganda and Kenya there has been widespread controversy over their elections in recent years. The election of 1946 in Romania was heavily rigged. The communists took over Romania and abolished the multi-party system to gain complete control of the country.

These instances of controversial elections could all have been avoided if the counting process was fair, transparent and verifiable. The current ballot system does offer anonymity to the voter but the counting process is not transparent. People are supposed to trust the result which is provided by an Election commission or a government body. This makes the process of counting, a major vulnerability in the current process. There are also other major electoral scams such as voter fraud, ballot stuffing and booth capturing. All these make it very difficult for organizers of an election to distinguish between the actual votes and votes added without authorization.

The system that is being proposed solves most of the issues mentioned above and can be implemented in the current world environment.

**Design methodology and software process model:**

* **Design Methodology**

We will use Object Oriented Methodology and Block Chain Methodology for developing software systems. The idea behind Block chain is the decentralized network of the nodes operating dependently. Data can be organized into objects and classes easily.

* **Process Methodology**

We will use Iterative and Incremental process model for our system. The reason behind selecting iterative model is that it slices the system functionality into increments. Requirements are not completely known. After each iteration, system can be evaluated, and can be implemented in the next iteration according to the changing requirements. As it requires a lot of research to implement, where requirements are being changed by time, **“Registering to Vote”** will be developed according to this process model.

**System overview**

This voting system is going to solve many problems in current voting System. This system is designed on block chain methodology which is till now the most secured system in now days.

In our proposed system the verification of the Voter is done by using finger print scanning and matching it to the person data in the database. The backend voting process is handled by using block chain technology. Every vote is being added to the specific node by using the concept of smart contract.

The counting and recounting of the Votes can be easily done by using Block Chain technology.

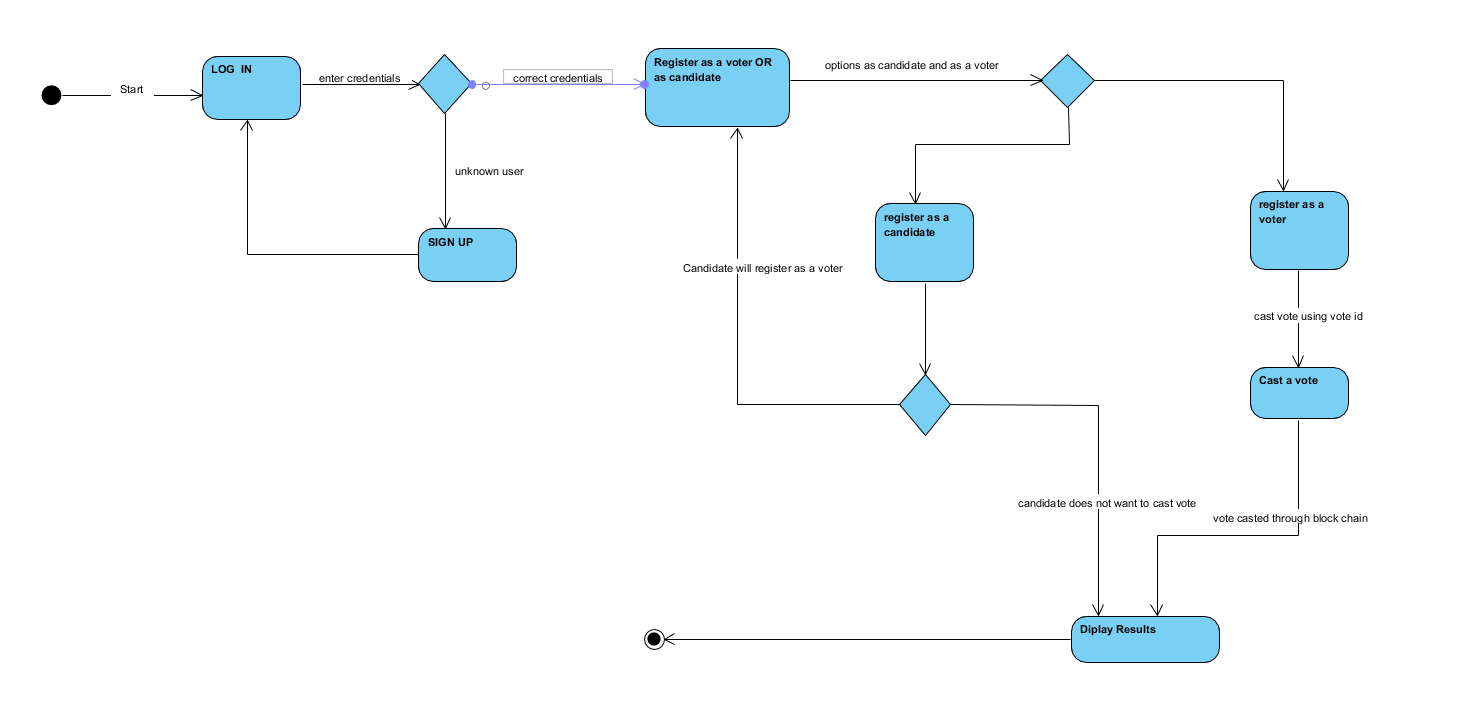
It is secure and safe. The integrity of the data is maintained. Our Proposed system maintains the integrity of the data. It is reliable, fast easy and secure. We are developing this system to maintain transparency in the Voting process.

**Architectural design**

****

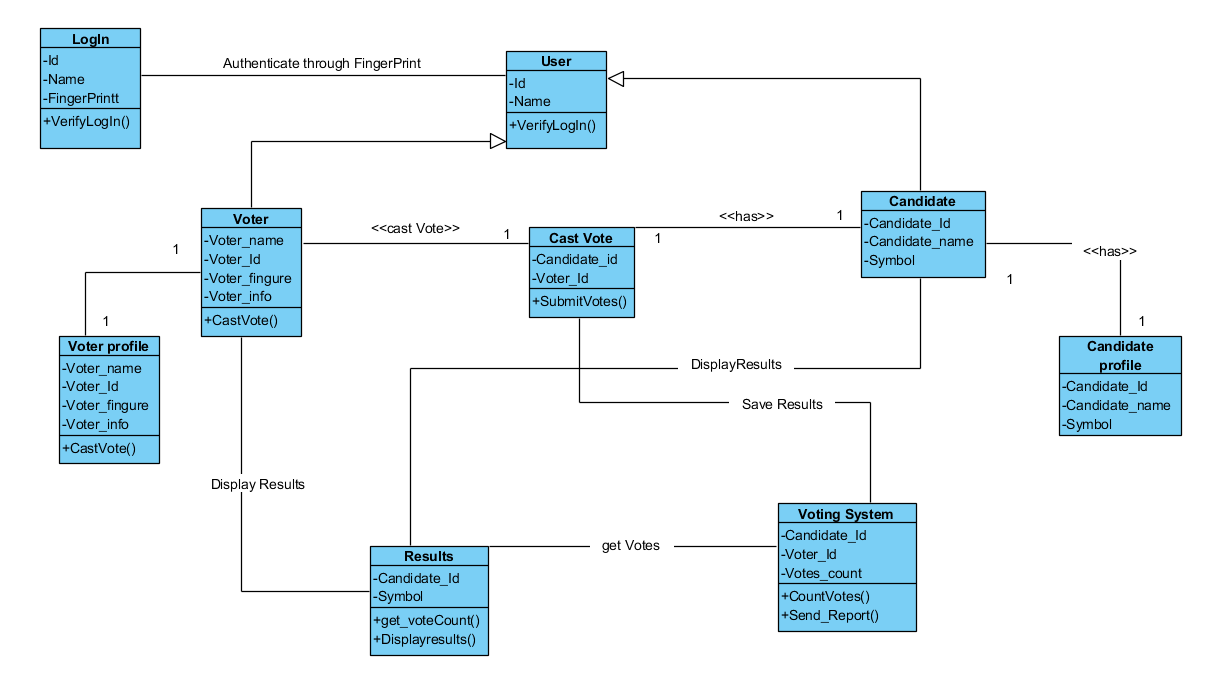
This is the overall overview of the system which we are going to develop. In this system the user first entered their information and stored in back end database. Then during voting hours the user will first log in and will authenticate through Authentication Server by matching user figure print in database. Then if the user is Authenticated then the user will cast a vote and vote will be casted through Arbitration Server then voting and counting will occur in private block chain server, then the results will displayed to voter at the end of voting hours.

**Process flow/Representation:**

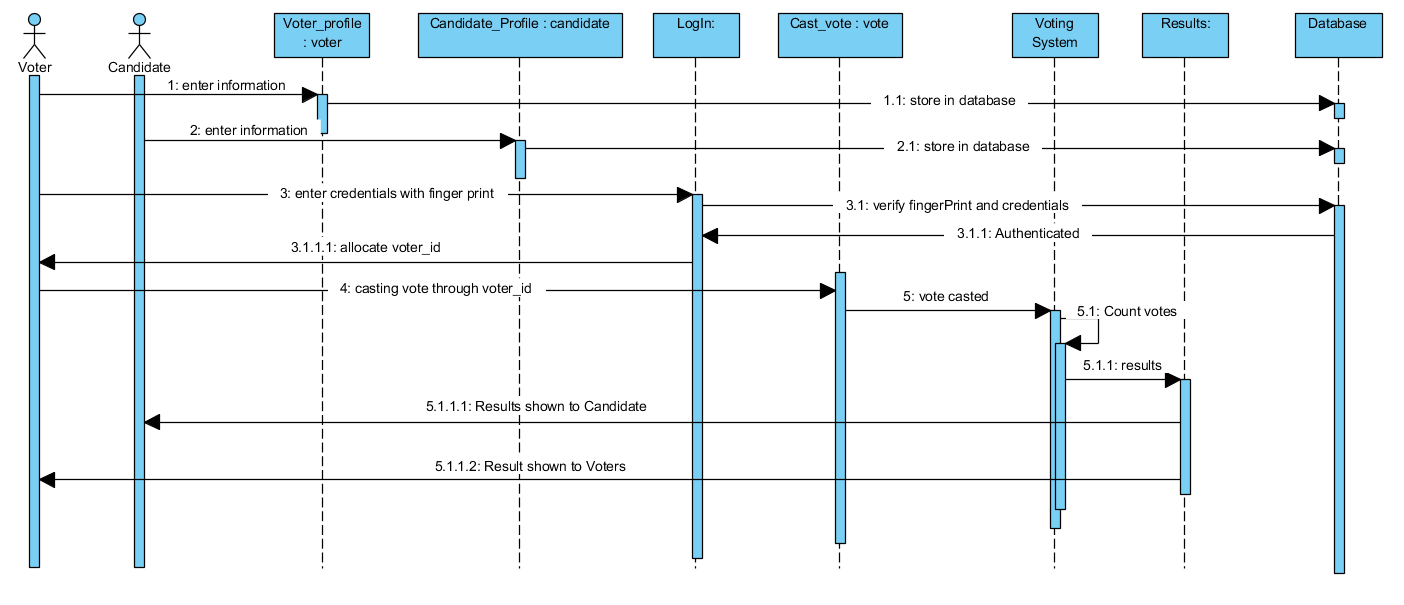


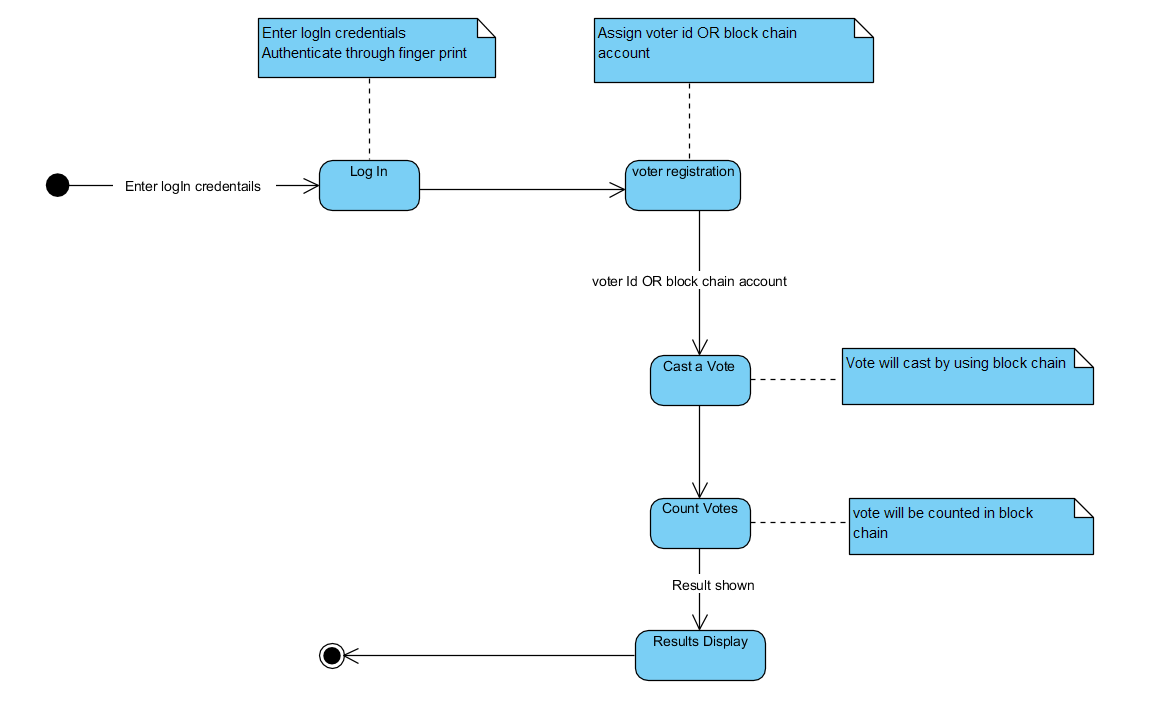
**Design models**

1. **Class Diagram**



1. **Sequence Diagram**



1. **State Transition Diagram**

**Data design**

The voter’s and candidate’s data will be stored in firebase backend database which will be connected to android and web base application.

**J-son Schema:**

{

"Voters":{

"Voter\_name":{

"type":"string",

"required":true

},

"email":{

"type":"string",

"required": true,

"unique": true

},

"password":{

"type":"string",

"required":true

}

},

"Candidate":{

"Candidate\_name":{

"type":"string",

"required":true

},

"email":{

"type":"string",

"required": true,

"unique": true

},

"password":{

"type":"string",

"required":true

}

},

}

**Data dictionary:**

|  |  |
| --- | --- |
| **Sr. No** | **Node Name** |
| **1** | **Voter** |
| **2** | **Candidate** |

**Voter:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Voter\_Name | String | 50 | Voter’s real name |
| Email | String | 50 | Voter’s Email |
| Password | String | 20 | Voter’s Password |

**Candidate:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Candidate\_Name | String | 50 | Candidate’s name |
| Email | String | 50 | Candidate’s Email |
| Password | String | 20 | Candidate’s Password |

**Algorithm & Implementation**

**SMART CONTRACT ALGORITHM:**

pragma solidity 0.4.25;

contract Election {

// Model a Candidate

struct Candidate {

uint id;

string name;

uint voteCount;

}

// Store accounts that have voted

mapping(address => bool) public voters;

// Store Candidates

// Fetch Candidate

mapping(uint => Candidate) public candidates;

// Store Candidates Count

uint public candidatesCount;

// voted event

event votedEvent (

uint indexed \_candidateId

);

constructor () public {

addCandidate("Candidate 1");

addCandidate("Candidate 2");

}

function addCandidate (string \_name) private {

candidatesCount ++;

candidates[candidatesCount] = Candidate(candidatesCount, \_name, 0);

}

function vote (uint \_candidateId) public {

// require that they haven't voted before

require(!voters[msg.sender]);

// require a valid candidate

require(\_candidateId > 0 && \_candidateId <= candidatesCount);

// record that voter has voted

voters[msg.sender] = true;

// update candidate vote Count

candidates[\_candidateId].voteCount ++;

// trigger voted event

emit votedEvent(\_candidateId);

}

}

**Finger Print Detection Algorithm:**

// Start by reading in an image

Mat input = imread("/data/fingerprints/image1.png", CV\_LOAD\_GRAYSCALE);

// Binarize the image, through local thresholding

Mat input\_binary;

threshold(input, input\_binary, 0, 255, CV\_THRESH\_BINARY | CV\_THRESH\_OTSU);

#include

#include

using namespace std;

using namespace cv;

// Perform a single thinning iteration, which is repeated until the skeletization is finalized

void thinningIteration(Mat& im, int iter)

{

Mat marker = Mat::zeros(im.size(), CV\_8UC1);

for (int i = 1; i (i-1, j);

uchar p3 = im.at(i-1, j+1);

uchar p4 = im.at(i, j+1);

uchar p5 = im.at(i+1, j+1);

uchar p6 = im.at(i+1, j);

uchar p7 = im.at(i+1, j-1);

uchar p8 = im.at(i, j-1);

uchar p9 = im.at(i-1, j-1);

int A = (p2 == 0 && p3 == 1) + (p3 == 0 && p4 == 1) +

(p4 == 0 && p5 == 1) + (p5 == 0 && p6 == 1) +

(p6 == 0 && p7 == 1) + (p7 == 0 && p8 == 1) +

(p8 == 0 && p9 == 1) + (p9 == 0 && p2 == 1);

int B = p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9;

int m1 = iter == 0 ? (p2 \* p4 \* p6) : (p2 \* p4 \* p8);

int m2 = iter == 0 ? (p4 \* p6 \* p8) : (p2 \* p6 \* p8);

if (A == 1 && (B >= 2 && B (i,j) = 1;

}

}

im &= ~marker;

}

// Function for thinning any given binary image within the range of 0-255. If not you should first make sure that your image has this range preset and configured!

void thinning(Mat& im)

{

// Enforce the range tob e in between 0 - 255

im /= 255;

Mat prev = Mat::zeros(im.size(), CV\_8UC1);

Mat diff;

do {

thinningIteration(im, 0);

thinningIteration(im, 1);

absdiff(im, prev, diff);

im.copyTo(prev);

}

while (countNonZero(diff) > 0);

im \*= 255;

}

Apply thinning algorithm

Mat input\_thinned = input\_binary.clone();

thinning(input\_thinned);

Mat harris\_corners, harris\_normalised;

harris\_corners = Mat::zeros(input\_thinned.size(), CV\_32FC1);

cornerHarris(input\_thinned, harris\_corners, 2, 3, 0.04, BORDER\_DEFAULT);

normalize(harris\_corners, harris\_normalised, 0, 255, NORM\_MINMAX, CV\_32FC1, Mat());

float threshold = 125.0;

vector keypoints;

Mat rescaled;

convertScaleAbs(harris\_normalised, rescaled);

Mat harris\_c(rescaled.rows, rescaled.cols, CV\_8UC3);

Mat in[] = { rescaled, rescaled, rescaled };

int from\_to[] = { 0,0, 1,1, 2,2 };

mixChannels( in, 3, &harris\_c, 1, from\_to, 3 );

for(int x=0; x(y, x) > threshold ){

// Draw or store the keypoint location here, just like you decide. In our case we will store the location of the keypoint

circle(harris\_c, Point(x, y), 5, Scalar(0,255,0), 1);

circle(harris\_c, Point(x, y), 1, Scalar(0,0,255), 1);

keypoints.push\_back( KeyPoint (x, y, 1) );

}

}

}

// Imagine we have a vector of single entry descriptors as a database

// We will still need to fill those once we compare everything, by using the code snippets above

vector database\_descriptors;

Mat current\_descriptors;

// Create the matcher interface

Ptr matcher = DescriptorMatcher::create("BruteForce-Hamming");

// Now loop over the database and start the matching

vector > all\_matches;

for(int entry=0; i matches;

matcheràmatch(database\_descriptors[entry], current\_descriptors, matches);

all\_matches.push\_back(matches);

}

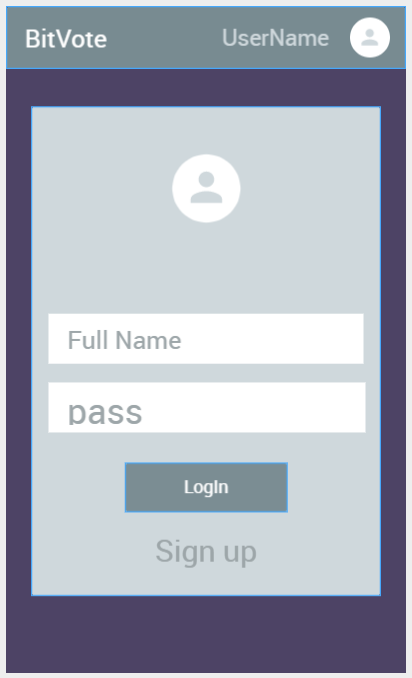
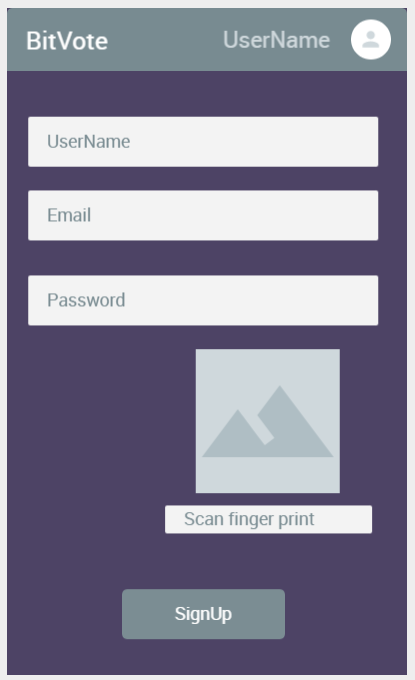
**Software requirements traceability matrix**

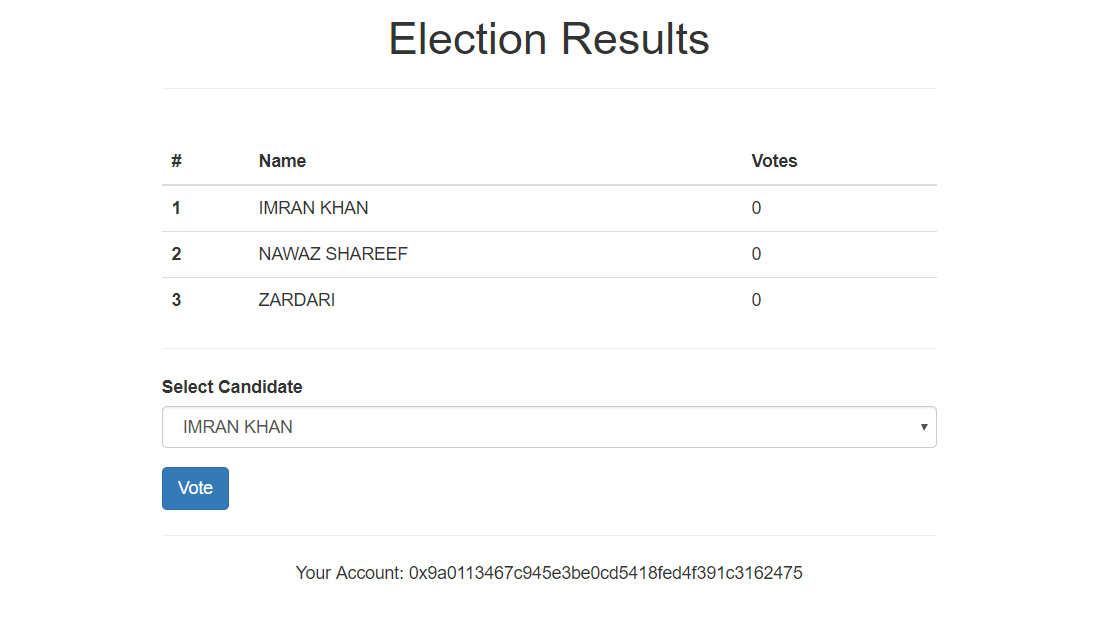
**Table 1 Requirements Traceability Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. Number** | **Ref. Item** | **Design Component** | **Component Items** |
| FR01 | Voter and candidate registration | Sequence Diagram | Registration |
| FR02 | Storing Registered voter information | Sequence Diagram | Database |
| FR03 | Fill all Data fields | Class diagram | Candidate profile,  Voter profile |
| FR04 | Finger Print Capturing | State Transition Diagram | Log In |
| FR05 | Facial Recognition Capturing | State Transition Diagram | Log In |
| FR06 | Cast a Vote | Class Diagram | Cast Vote |
| FR07 | Display Candidates | Class Diagram | Voting System |
| FR08 | Cast Vote to Candidate | Class Diagram | Voting System |
| FR09 | Vote Confirmation | Class Diagram | Voting System |
| FR10 | Verify casted Vote | Class Diagram | Voting System |
| FR11 | Counting Votes in Block chain | Class Diagram | Voting System |
| FR12 | Display Results | Class Diagram | Results |

**Human interface design**

* The user will Log In through authenticate by figure print.
* The user will register their self for as a voter by providing his/her information.
* The voter will get voter id.
* The voter will enter a voter id before casting vote.
* The voter will cast a vote to his/her favorite candidate.
* The result will be shown to voters and candidate.

****