Decentralized E-Voting System Using Blockchain

A project report was submitted in partial fulfillment of the requirements for B.Tech. Project

B.Tech.

by

Abhay Chaurasiya (2018IMT-005)

ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT

GWALIOR-474 015

2021

CANDIDATES DECLARATION

I hereby certify that the work, which is being presented in the report, entitled **Decentralized E-Voting System Using Blockchain**, in partial fulfillment of the requirement for the award of the Degree of Bachelor of Technology and submitted to the institution is an authentic record of our work carried out during the period *June 2021* to *October 2021* under the supervision of **Dr. Anuraj Singh**. I also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Date: Signatures of the Candidates

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: Signatures of the Research Supervisors

ABSTRACT

Voting is a fundamental right for every nation. An Electronic Voting (E-Voting) system is a voting system in which the election process is notated, saved, stored, and processed digitally, which makes the voting management task better than the traditional paper-based method. Blockchain is offering new opportunities to develop new types of digital services. Because of the properties such as transparency, decentralization, irreversibility, nonrepudiation, etc., blockchain is not only a fundamental technology of great interest in its own right but also has large potential when integrated into many other areas. In this project, the concept of developing an electronic voting system using blockchain technology is implemented. The two-level architecture provides a secure voting process without the redundancy of existing (not based on blockchain) systems. The blockchain-based voting project has two modules to make the whole project integrated and work along. One will be the Election Commission who will be responsible for creating elections, adding registered parties, and candidates contesting for the election added under the smart contracts. The other end will be the voter's module where each individual can cast a vote for their respective Assembly Constituency and the vote will be registered on the blockchain to make it tamper-proof. In previous research because of the transparency property from blockchain, ballots are visible when they are cast to the blockchain network. This exposes the progress of the election during the voting phase, and may greatly influence the outcome of the election. Here, we come up with a solution by simply employing a permissioned blockchain for the election. The attention-seeking idea is to use two linked side chains, a one-way pegged sidechain. The first sidechain records the voting operation of voters and the second sidechain counts the votes assigned to the various candidates.

ACKNOWLEDGEMENTS

I am highly indebted to **Dr. Anuraj Singh** and are obliged for giving us the autonomy of functioning and experimenting with ideas. We would like to take this opportunity to express our profound gratitude to them not only for their academic guidance but also for their above interest in our project and constant support coupled with confidence-boosting and motivating sessions which proved very fruitful and were instrumental in infusing self-assurance and trust within us. The nurturing and blossoming of the present work is mainly due to their valuable guidance, suggestions, astute judgment, constructive criticism, and an eye for perfection. Our mentor always answered a myriad of our doubts with smiling graciousness and prodigious patience, never letting us feel that I am novices by always lending an ear to our views, appreciating and improving them, and by giving us a free hand in our project. It's only because of their overwhelming interest and helpful attitude, the present work has attained the stage it has.

Finally, I am grateful to our Institution and colleagues whose constant encouragement served to renew our spirit, refocus our attention and energy, and help us in carrying out this work.

(Abhay Chaurasiya)
2018IMT-005

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

API	Application Programming Interface
EC	Election Commision
ETH	Ethereum
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
SDLC	Software Development Lifecycle
STLC	Software Testing Life Cycle

CHAPTER 1

INTRODUCTION AND LITERATURE SURVEY

1.1 INTRODUCTION

Current majority rule governments are based upon the customary voting form or electronic democratic (e-voting). As of late, gadgets known as EVMs are immensely censored because of unpredictable reports of the political decision results. There have been many inquiries in regards to the plan and inner design of these gadgets and how they very well may be powerless to assaults. Internet casting a ballot is pushed as an expected answer for drawing in the youthful residents and the aliens of the country. While votes are typically recorded, coordinated, checked, and confirmed by a focal power, a blockchain-based electronic democratic framework could decentralize controls, making citizens assume control over specific assignments while holding a duplicate of the electing register.4 Blockchain can assist with carrying out an electronic democratic framework that is permanent, straightforward, and can't be hacked into to change the outcomes. Blockchain casting a ballot is a compelling way to lead reasonable races. Utilizing a political decision's REST API facilitated on Ethereum's Blockchain, the subtleties are displayed at the front-finish of the elector for making the choice. Then, at that point, while surveying the vote is put away on our blockchain system of which the Election Commission brings the vote check.

1.2 Background

1.2.1 Existing System

In India, before 2004 there was a paper-based democratic framework. This is called the polling form Paper framework. Electors needed to go to surveying stalls and cast their vote by stamping on the seal before the image of a contender for which they needed to project their decisions on polling form paper. Results were reported by tallying the votes. The greatest vote gainer was pronounced as champ. India has a populace in excess of 120 crores the polling form paper casting a ballot isn't much dependable, tedious, and extremely challenging to check the vote and there are additional issues like supplanting of polling form paper boxes with copy, harm of polling form paper, checking stamp seal for more than one competitor consequently, there is a solid need to beat these issues. To beat these issues Electronic Voting Machines Were presented.

Electronic Voting Machine (EVM's) are fundamentally comprised of two segments:

1. Control Unit: It stores and collects votes, utilized by survey laborers.
2. Ballot Unit: It is set in the political race corner and is utilized by the electors.

Both the units are associated through a 5m link and one finish of the link is forever fixed to the voting form unit. The control unit has a battery pack inside, which mechanizes the framework. The polling form unit has 16 applicant catches and the unused catches are covered with a plastic veiling tab inside the unit. An extra polling form unit can be associated when there are more than 16 applicants. The extra voting form unit can be associated with a port on the underside of the principal voting form unit. EVM's are globally known as DRE's (Direct Recording Electronic). EVM's have been utilized in India since the overall appointment of 2004 when polling forms were totally out of the pattern. They have been utilized in all the gathering surveys and general appointments of 2009. By utilizing Evm's, Votes are accurately recorded and there is no issue in checking, adaptability, Accuracy, quick announcement of results, and vigor of the framework. The primary problem lies in confirmation, the individual who is casting a ballot may not be the real individual. Different issues like catching corners by ideological groups, projecting votes by underage individuals, and misrepresentation casting a ballot might happen. An individual is furnished with the elector id card as proof of character, given by the Indian government. Parcel of issues are found in elector id cards like name misprinting, missing of name, no unmistakable photograph on personal id card, and so on

1.2.2 Proposed System

A few examinations have been done on utilizing PC advancements to further develop decisions. These examinations tell about the dangers of taking on electronic democratic frameworks, on account of the product challenges, insider dangers, network weaknesses, and the difficulties of reviewing.

I have proposed to plan the current web-based democratic framework which is coordinated with the Blockchain innovation. The proposed framework enjoys the accompanying benefits when contrasted with the current framework:

- Here I used a permission blockchain that does not allow ballots to be visible during vote cast.
- Clients' can cast a ballot from any place on the planet until they have citizenship of the country.
- The democracy is put away in the Blockchain which makes it sealed.
- As there's no remaining in line for making choice it will save a great deal of time and lessen the responsibility

1.3 LITERATURE REVIEW

This paper [2], has featured the serious issue in casting ballot security wherein the 2016 US Presidential Elections, EVM's were probably going to be captured and cast ballots were altered. The investigation tracked down that this old democratic gear isn't simply more inclined to disappointments and crashes but on the other hand is famously simple to hack and mess with. In this examination [3] by Ayed, Ahmed, et alIt has been proposed as an electronic democratic framework dependent on Blockchain innovation. The framework is decentralized and doesn't depend on trust. Any enrolled elector can cast a ballot utilizing any gadget associated with the Internet. The Blockchain will be freely evident and appropriated such that nobody will want to ruin it. Rifa and Budi have resolved that the utilization of hash esteems in recording the democratic consequences of each surveying station connected makes this recording framework safer and the utilization of computerized marks makes the framework more dependable. The utilization of the grouping proposed in the blockchain creation measure in this framework thinks about that in an electing framework not needed for mining as in the Bitcoin framework because the elector information and numbers are clear and are not permitted to choose more than once, the proposed succession guarantees that all hubs Which are legitimately associated and can stay away from the crash in transportation [4]. Canister, Joseph, et al., has resolved that the current blockchain casting a ballot framework can't give the complete security elements, and a large portion of them are stage subordinate, we have proposed a blockchain-based democratic framework that the electors' protection and casting a ballot accuracy are ensured by homomorphic encryption, linkable ring mark, and PoKs between the citizen and blockchain.

1.4 Objectives

- To use the permission blockchain so that ballots are not visible during vote cast.
- To reduce the workload of setting up an election booth and conducting elections in physical form.
- Non-Resident Indians can cast their votes as it is online.
- I am supposed to learn the concept of Blockchain and how it can be utilized to work in different sectors.

CHAPTER 2

DESIGN DETAILS AND IMPLEMENTATION

The proposed e-casting ballot framework is considered ElectoBlock, a web application. It can help in carrying out an electronic democratic framework that is permanent, straightforward, and can't be hacked into to change the outcomes. The framework has been intended to help a democratic application in a real climate considering explicit necessities like security, qualification, accommodation, receipt-freeness, and obviousness. The proposed framework means to accomplish secure computerized casting of a ballot without undermining its convenience. Inside this specific circumstance, the framework is planned to utilize an online interface to work with client commitment. Besides, the situation permits all elector's equivalent privileges of interest and fosters a reasonable and solid rivalry among every one of the up-and-comers while keeping the secrecy of the citizens protected.

2.1 Design Goals

Plan objectives are significant properties of the framework to be improved and might influence the general plan of the framework. There is a scarce difference between the framework plan and prerequisites. Necessities incorporate explicit qualities that should be met altogether for the item to be adequate to the customer, while plan objectives are properties that the planners endeavor to make "comparable to conceivable", without explicit rules for agreeableness.

2.2 Modularization Details

The task has been partitioned into numerous modules where for each usefulness I have assigned modules. Any product involves numerous frameworks which contain a few sub-frameworks and those sub-frameworks further contain their sub-frameworks. In this way, planning a total framework in one go consisting of every single required usefulness is chaotic work and the interaction can have numerous mistakes because of its tremendous size.

The proposed structure is isolated into the following modules:-

1. Political Decision Commission: In this module, a substance named Election Commission will be mindful to set up the shrewd agreement and register up-and-comers, gatherings and get going

in a political race.

2. Political decision Test: This is the module to test our quick understanding where we use Mocha Framework to perform unit tests on our application.

3. Elector Module: In this module, residents who have been outfitted with the individual ETH wallet will import onto the vote-based passage using the Metamask extension and cast their vote.

2.3 Design Details

2.3.1 Product Perspective

The product item is an independent framework and not a piece of a bigger framework. The framework will comprise two sections. One will be utilized for general purposes by the EC, like survey competitors, enrolled parties, and past years' political decision results. The citizens will arrive at the framework associated with another module through pages by utilizing internet browsers like Mozilla, Internet Explorer and Google Chrome. On the final voting day, the elector needs to import his/her Ethereum's wallet and get confirmed likewise. The electors cast their votes utilizing the interface that is given. These votes are acknowledged by the blockchain and afterward tossed into the worker. The EC arranges the entire framework as indicated by its requirements on the worker.

2.3.2 Product Features

1. Eligibility: This property expresses that lone qualified clients can cast a ballot. The individuals who are furnished with verification by the Election Commission.

2. Privacy: Protection is perhaps the main part of popularity-based democracy. Electors' protection ought to be kept up with. Nobody ought to have the option to realize how a specific individual cast a ballot or to whom the specific citizen cast a ballot.

3. Coercion Resistance: Nobody ought to have the option to drive the citizen and ought not to recognize whether the elector cast a ballot the same way he/she was told to cast a ballot.

4. Verifiability: This property expresses that everybody engaged with the democratic cycle ought to have the option to confirm the outcomes. This gets straightforward about the political race. Likewise, an individual citizen ought to have the option to confirm whether his/her vote is checked or not.

5. Immutability: The elector's vote ought to be changeless. Nobody ought to have the option to change the vote of any elector without the appropriate worry of the citizen. Every one of the records ought to be changeless.

2.3.3 Requirements, Assumptions, and Dependencies

The framework empowers electors to make their choice from any place and is confirmed by EC and given the ETH wallet address and private key. Security and namelessness are the most essential basics of this blockchain casting a ballot framework.

For the appropriate working of the framework we can list our suspicions and conditions as follows:

• **Metamask Browser Extension:** Metamask permits clients to oversee accounts and their keys in an assortment of ways, including equipment wallets while disconnecting them from the site setting.

• **Ganache:** It is an individual blockchain for quick Ethereum and Corda dispersed application improvement.

• **Truffle:** An elite improvement climate, testing system, and resource pipeline for blockchains utilizing the Ethereum Virtual Machine (EVM), meant to make life as an engineer simpler.

• **NodeJS:** It is a JavaScript runtime based on Chrome's V8 JavaScript motor.

2.3.4 Technologies Use

Software	Type
Ganache	Ethereum Blockchain Server
Truffle	Development framework For ETH
Remix	Solidity’s IDE
VS Code	Integrated Development Environment
Metamask	Ethereum Wallet
Mx Linux	Operating System
Node	JavaScript Runtime

2.4 Non-functional Requirements

2.4.1 Performance Requirements

The framework is relied upon to have a sensible brief time frame of reaction. The elector

ought to have the option to import his/her wallet given by the Election Commission inside a couple of moments remembering the state of organization steadiness. The framework's exhibition is distinctive as per its modes:

(i) Election Mode: In this stage, the normal opportunity to convey the brilliant agreements relies on t the excavators associated with the blockchain and the measure of GAS we choose to close down the exchange to set apart as approved one however as I am working locally, it is simply a question of a large portion of a moment or somewhere in the vicinity.

ii) Voting Mode: In this stage, the framework will react inside the space of seconds as we don't need to close down exchanges just to bring the rundown of possibility for the races however relying upon the organization soundness and web3 association the above exhibition may be deferred. Then, in the wake of making the choice, it may require a little while to close down the exchange contingent on the excavators and GAS limit.

2.4.2 Security Requirements

• The information exchange among customers and the blockchain worker should be done over HTTPS to stay away from blended substance assault.

• The reentrancy on a solitary capacity must be limited while conveying the shrewd agreement.

• To address the number flood blunder, checking the votes have been done inside a particular occasion answerable for it.

2.4.3 Reliability

• **In Election Mode:** The framework should be kept up with occasionally as though the keen agreement which is to be conveyed experiences any bugs, it should be fixed to forestall votes erroneous conclusion and exchange blunder taking care of.

• **In Voting Mode:** As the support, part is in the Election Mode, in case there's any mistake in web3 association the interoperability status may change in any case the framework will work faultlessly constantly.

2.4.4 Usability

•The framework will have a negligible and basic User Interface.

• To direct the clients interestingly utilizing it, there will be directly identified with the utilization

of the framework.

2.5 Data Model And Description

2.5.1 Sequence Diagram

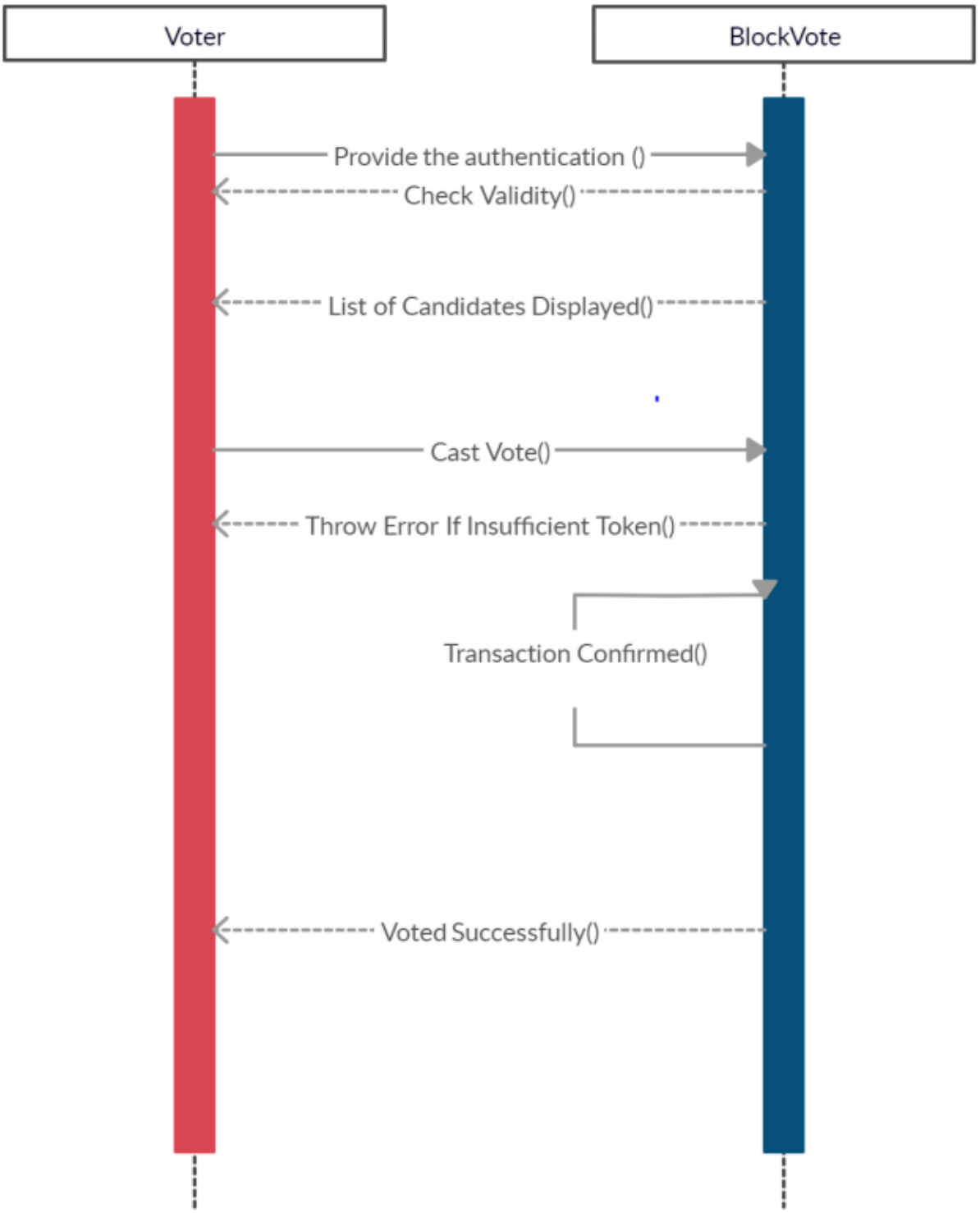


Figure 1: Sequence Diagram

2.5.2 Flowchart

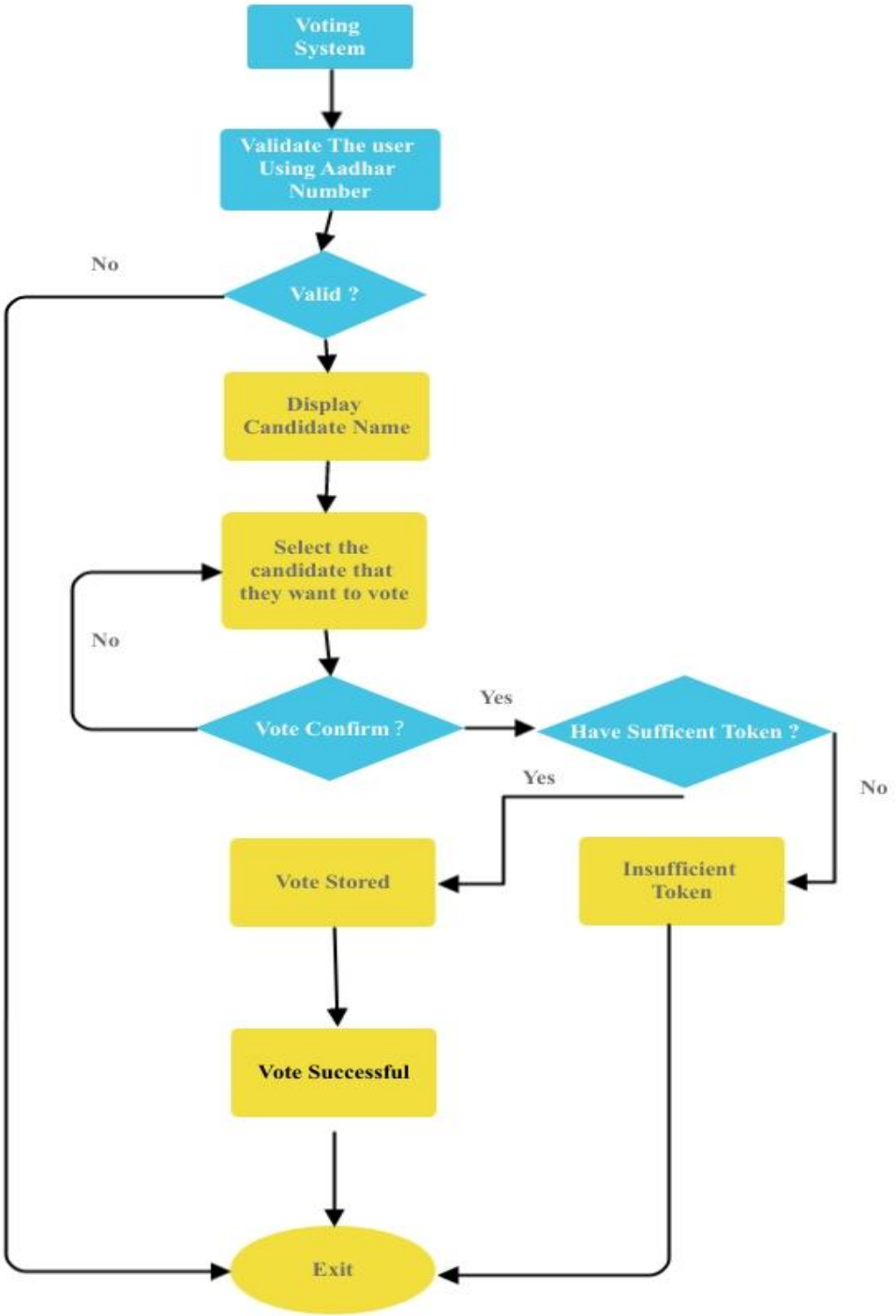


Figure 2: Flow Chart

2.6 Implementations

The tiers given below alludes to different levels or layers where activities occur.

Client: Client is any client or program that needs to play out an activity over the framework. Customers connect with the framework through a Presentation layer.

Presentation Layer: This layer is liable for the introduction of information at the customer side, i.e., it gives an interface to the end client into the application to project the votes.

Resource manager: The asset supervisor manages the association (stockpiling, ordering, and recovery) of the information important to help the application rationale. This asset administrator here is the Local Blockchain worker kept up with by Ganache.

Application logic: The application rationale sorts out what the framework does. It deals with carrying out the business leads and building up the business measures. Blockchain casting a ballot framework is planned and carried out as indicated by the three-level design.

2.6.1 Coding

The framework configuration is converted into a machine-decipherable structure which is named coding. It is fundamentally deciphering the intelligible arrangement to a machine-accommodating one. The code age step plays out in this undertaking.

The following points are considered while converting the system design into coding:

• Are the initializations correct?
• Are the data types properly assigned?
• Is memory leak being dealt with?
• Does it comply with the coding standard?

2.6.2 Coding Standardization

Coding Standardization is essentially the proficiency of our code that has been changed over from the framework plan. The effectiveness essentially relies on:

• **Readability:** The code should be readable with proper indentation and spacing to make the contents clear of all the modules.
• **Portability:** The code is portable enough as it will work on the various platforms given all the necessary dependencies are installed.
• **Debug Easily:** The coding should be error-free as much as possible.

2.6.3 Source Code:

```
pragma solidity >=0.5.16;

//name of the Contract

contract Migrations {
  address public owner;
  uint public last_completed_migration;

  modifier restricted() {
    if (msg.sender == owner) _;
  }

//assigning the sender of the transaction to be owner
  constructor () public{
    owner = msg.sender;
  }

//Setting up the migration for the first time to be deployed on the Blockchain
  function setCompleted(uint completed) public{
    last_completed_migration = completed;
  }

//On necessary changes, upgrade function is triggered
  function upgrade(address new_address) public {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
  }
}
```

Figure 3: Migration Smart Contract

```
pragma solidity >=0.5.16;

contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        string party;
        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
    );

    //Adding Election Candidates along with the parties
    constructor () public {
        addCandidate("Raju Bista","Bharatiya Janata Party");
        addCandidate("Sankar Malakar","Indian National Congress");
        addCandidate("Saman Pathak","Communist Party Of India (Marxist)");
        addCandidate("Amar Singh Rai","All India Trinamool Congress");
        addCandidate("Sudip Mandal","Bahujan Samaj Party");
        addCandidate("NOTA","None of the above");
    }

    //Function to trigger the adding candidates
    function addCandidate (string memory name,string memory party) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, name,party, 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);
    }
}
```

Figure 4: Election Smart Contract

```
- arguments: [
  - {
      argumentTypes: null,
      hexValue: "52616a75204269737461",
      id: 28,
      isConstant: false,
      isLValue: false,
      isPure: true,
      kind: "string",
      lValueRequested: false,
      nodeType: "Literal",
      src: "587:12:0",
      subdenomination: null,
    - typeDescriptions: {
        typeIdentifier: "t_stringliteral_a846bf2ab39e91538fb3d470483720053f3816d751cd534e01487163f7d794f6",
        typeString: "literal_string "Raju Bista""
      },
      value: "Raju Bista"
    },
  - {
      argumentTypes: null,
      hexValue: "426861726174697961204a616e617461205061727479",
      id: 29,
      isConstant: false,
      isLValue: false,
      isPure: true,
      kind: "string",
      lValueRequested: false,
      nodeType: "Literal",
      src: "600:24:0",
      subdenomination: null,
    - typeDescriptions: {
        typeIdentifier: "t_stringliteral_34b76bc47431a027fdaf3fbf527be0755a20fb4fd744ccab14ec8f1036aaf4db",
        typeString: "literal_string "Bharatiya Janata Party""
      },
      value: "Bharatiya Janata Party"
    }
  ],
- expression: {
  - argumentTypes: [
    - {
        typeIdentifier: "t_stringliteral_a846bf2ab39e91538fb3d470483720053f3816d751cd534e01487163f7d794f6",
        typeString: "literal_string "Raju Bista""
```


Figure 5: Election API

```
initWeb3: function () {

  // TODO: refactor conditional
  if (typeof web3 !== 'undefined') {

    // If a web3 instance is already provided by Meta Mask.
    App.web3Provider = web3.currentProvider;
    web3 = new Web3(web3.currentProvider);
  } else {

    // Specify default instance if no web3 instance provided
    App.web3Provider = new Web3.providers.HttpProvider('http://localhost:7545');
    ethereum.enable();
    web3 = new Web3(App.web3Provider);
  }
  return App.initContract();
},
```

Figure 6: Initializing the web3 connection on Front-End

```
initContract: function () {
  $.getJSON("Election.json", function (election) {

    // Instantiate a new truffle contract from the artifact
    App.contracts.Election = TruffleContract(election);

    // Connect provider to interact with contract
    App.contracts.Election.setProvider(App.web3Provider);

    //invokes listen for Events
    App.listenForEvents();
    App.listenForAccountChange();

    return App.render();
  });
},
```

Figure 7: Initializing the smart contract

```
App.contracts.Election.deployed().then(function (instance) {
    electionInstance = instance;
    return electionInstance.candidatesCount();
  }).then(function (candidatesCount) {
    var candidatesResults = $("#candidatesResults");
    candidatesResults.empty();

    var candidatesSelect = $('#candidatesSelect');
    candidatesSelect.empty();

    for (var i = 1; i <= candidatesCount; i++) {
      electionInstance.candidates(i).then(function (candidate) {
        var id = candidate[0];
        var name = candidate[1];
        var voteCount = candidate[3];
        var party = candidate[2];
        // Render candidate Result
        var candidateTemplate =
          `<tr><th>${id}</th><td>${name}</td><td>${party}</td><td>${voteCount}</td></tr>`;
        candidatesResults.append(candidateTemplate);

        // Render candidate ballot option
        var candidateOption = `<option value="${id}"> ${name} (${party}) </option>`
        candidatesSelect.append(candidateOption);
      });
```

Figure 8: Trigger voted Events

```
castVote: function () {
    var candidateId = $('#candidatesSelect').val();
    App.contracts.Election.deployed().then(function (instance) {
      return instance.vote(candidateId, { from: App.account });
    }).then(function (result) {
      // Wait for votes to update
      $("#content").hide();
      $("#loader").show();
      alert("Thanks for voting")
    }).catch(function (err) {
      console.error(err);
    });
  }
```

Figure 9: CastVote function to vote

CHAPTER 3

RESULT AND TESTING

3.1 Testing

This undertaking utilizes Mocha as the testing system to unit test and joining test all of our experiments for the application. Following methodologies are utilized:

(i) **Unit Testing:** This is the first and the main degree of testing. Its need starts from the second a developer fosters a unit of code. Each unit is tried for different situations. Distinguishing and fixing bugs during the beginning phases of the Software Lifecycle decreases expensive fixes later on. It is significantly more conservative to discover and wipe out bugs during the beginning phases of the application building process. Henceforth, Unit Testing is the most significant of all the testing levels. As the product project advances it turns out to be increasingly more expensive to discover and fix the bugs.

Steps for Unit Testing are:-

Step 1: Creation of a Test Plan
Step 2: Creation of Test Cases and the Test Data
Step 3: Creation of scripts to run the test cases wherever applicable
Step 4: Execution of the test cases, once the code is ready
Step 5: Fixing of the bugs if present and re-testing of the code
Step 6: Repetition of the test cycle until the Unit is free from all types of bugs.

(ii) **Integration Testing:** Joining methodology represents how individual modules will be consolidated during Integration testing. The singular modules can be consolidated in one go, or they can be gone along with individually. A choice on the most proficient method to assemble the pieces is known as the Integration Strategy.

We have utilized a base-up mix way to deal with coordinate test our application.

In Bottom-Up Integration, we move from the base to the top for example the parts underneath are first composed and these are coordinated first. The incorporation occurs from base to top. If the calling party is yet to be created, it is supplanted by an uncommonly composed part called a Driver

3.2 Testing Design

```
//Checking the candidate count
it("initializes with six candidates along with the parties", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
    }).then(function(count) {
      assert.equal(count,6); //asserting the value
    });
  });
```

Figure 10: Candidate Count Unit Test

```
//Checks for double voting by a voter

it("throws an exception for double voting", function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      candidateId = 2;
      electionInstance.vote(candidateId, { from: accounts[1] });
      return electionInstance.candidates(candidateId);
    }).then(function(candidate) {
      var voteCount = candidate[3];
      assert.equal(voteCount, 1, "accepts first vote");
      // Try to vote again
      return electionInstance.vote(candidateId, { from: accounts[1] });
    }).then(assert.fail).catch(function(error) {
      assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
      return electionInstance.candidates(1);
    }).then(function(candidate1) {
      var voteCount = candidate1[3];
      assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
      return electionInstance.candidates(2);
    }).then(function(candidate2) {
      var voteCount = candidate2[3];
      assert.equal(voteCount, 1, "candidate 2 did not receive any votes");
    });
```

Figure 11: Double Voting Unit Test

```
//Casting the vote unit testing

it("allows a voter to cast a vote", function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      candidateId = 1;
      return electionInstance.vote(candidateId, { from: accounts[0] });
    }).then(function(receipt) {
      assert.equal(receipt.logs.length, 1, "an event was triggered");
      assert.equal(receipt.logs[0].event, "votedEvent", "the event type is correct");
      assert.equal(receipt.logs[0].args._candidateId.toNumber(), candidateId, "the candidate id is
correct");
      return electionInstance.voters(accounts[0]);
    }).then(function(voted) {
      assert(voted, "the voter was marked as voted");
      return electionInstance.candidates(candidateId);
    }).then(function(candidate) {
      var voteCount = candidate[3];
      assert.equal(voteCount, 1, "increments the candidate's vote count");
    })
  });
```

Figure 12: Invalid Candidate Unit Test

```
//Candidate Initialization Unit Testing

it("it initializes the candidates with the correct values", function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      return electionInstance.candidates(1);
    }).then(function(candidate) {
      assert.equal(candidate[0], 1, "contains the correct id");
      assert.equal(candidate[1], "Raju Bista", "contains the correct name");
      assert.equal(candidate[2], "Bharatiya Janata Party", "contains the correct party");
      assert.equal(candidate[3], 0, "contains the correct votes count");
      return electionInstance.candidates(2);
    }).then(function(candidate) {
      assert.equal(candidate[0], 2, "contains the correct id");
      assert.equal(candidate[1], "Sankar Malakar", "contains the correct name");
      assert.equal(candidate[2], "Indian National Congress", "contains the correct party");
      assert.equal(candidate[3], 0, "contains the correct votes count");
      return electionInstance.candidates(3);
    }).then(function(candidate) {
      assert.equal(candidate[0], 3, "contains the correct id");
      assert.equal(candidate[1], "Saman Pathak", "contains the correct name");
      assert.equal(candidate[2], "Communist Party Of India (Marxist)", "contains the correct party");
      assert.equal(candidate[3], 0, "contains the correct votes count");
      return electionInstance.candidates(4);
    }).then(function(candidate) {
      assert.equal(candidate[0], 4, "contains the correct id");
      assert.equal(candidate[1], "Amar Singh Rai", "contains the correct name");
      assert.equal(candidate[2], "All India Trinamool Congress", "contains the correct party");
      assert.equal(candidate[3], 0, "contains the correct votes count");
      return electionInstance.candidates(5);
    }).then(function(candidate) {
      assert.equal(candidate[0], 5, "contains the correct id");
      assert.equal(candidate[1], "Sudip Mandal", "contains the correct name");
      assert.equal(candidate[2], "Bahujan Samaj Party", "contains the correct party");
      assert.equal(candidate[3], 0, "contains the correct votes count");
      return electionInstance.candidates(6);
    }).then(function(candidate) {
      assert.equal(candidate[0], 6, "contains the correct id");
      assert.equal(candidate[1], "NOTA", "contains the correct name");
      assert.equal(candidate[2], "None of the above", "contains the correct party");
      assert.equal(candidate[3], 0, "contains the correct votes count");
    });
  });
```

Figure 13: Candidate Initialization Unit Test

3.3 Test Report

```
C:\Users\root\Desktop\blockvote-final-year-project> truffle test
Using network 'development'.


Compiling your contracts...
===========================

> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\root\AppData\Local\Temp\test-202069-286100-17j3ypn.net2
> Compiled successfully using:

   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



  Contract: Election

    √ initializes with six candidates along with the parties (355ms)
    √ it initializes the candidates with the correct values (2155ms)
    √ allows a voter to cast a vote (1193ms)
    √ throws an exception for invalid candiates (4578ms)
    √ throws an exception for double voting (1507ms)


  5 passing (10s)
```

Figure 14: Test Report

3.4 Reports

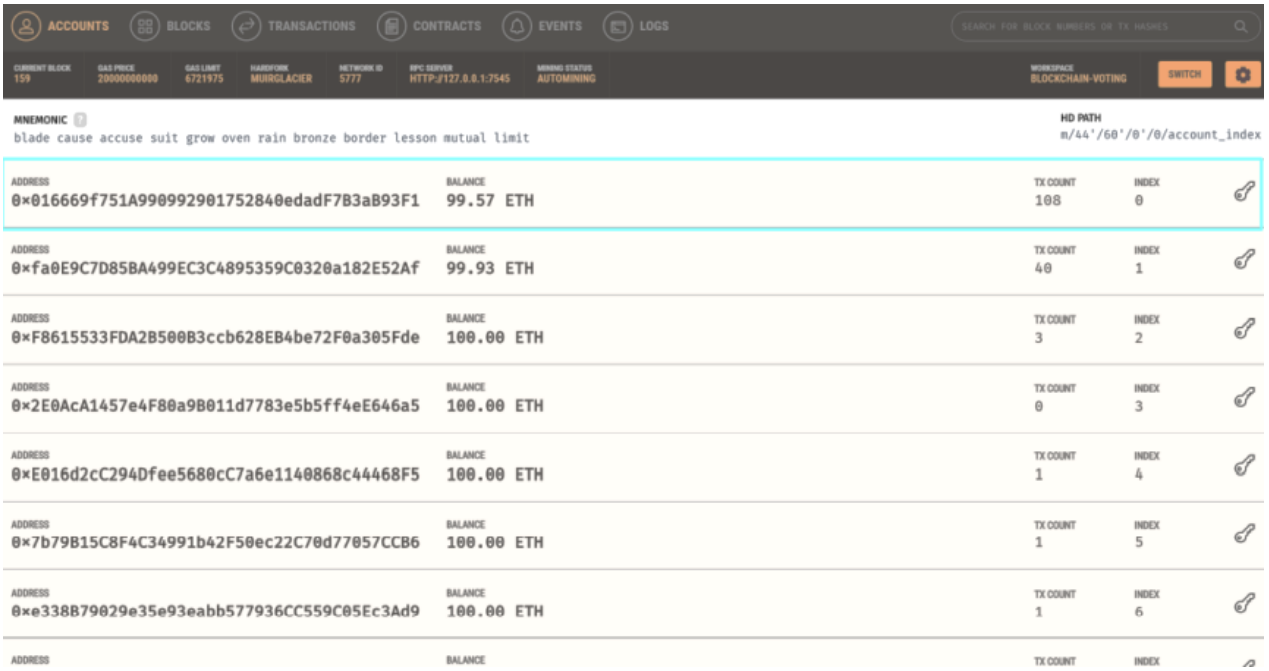


Figure 15: Smart Contract Owner Account

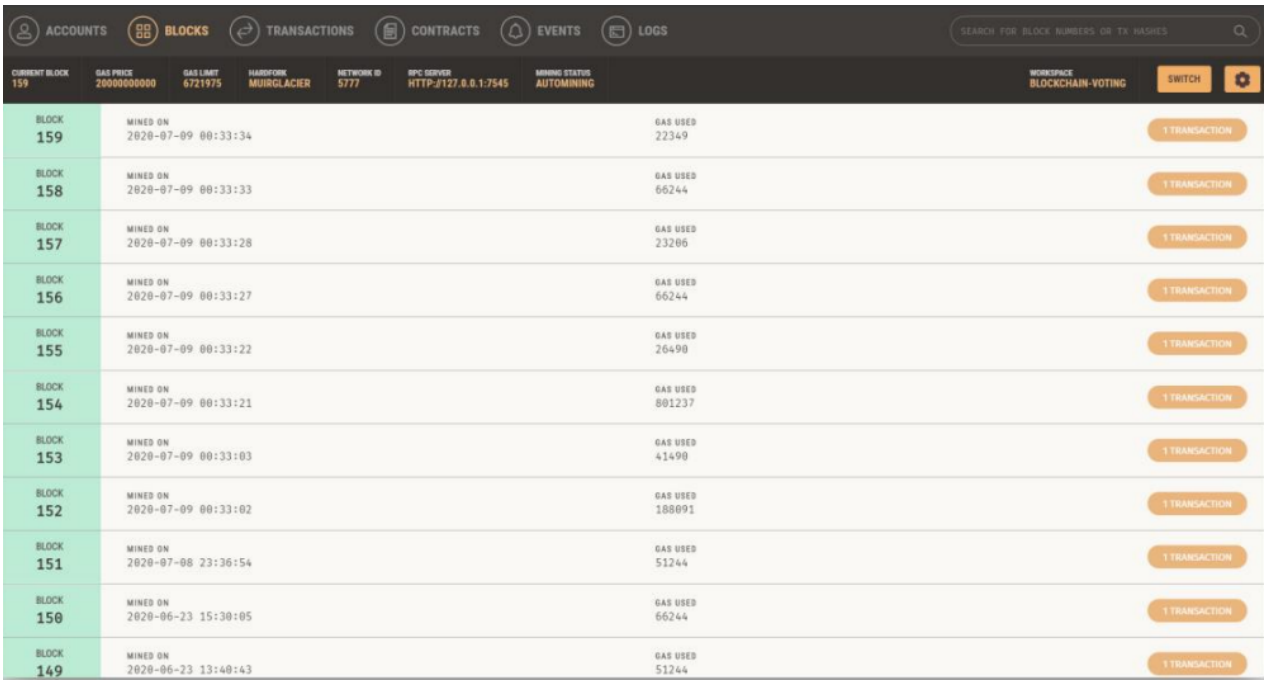


Figure 16: Blocks Mined after Transactions

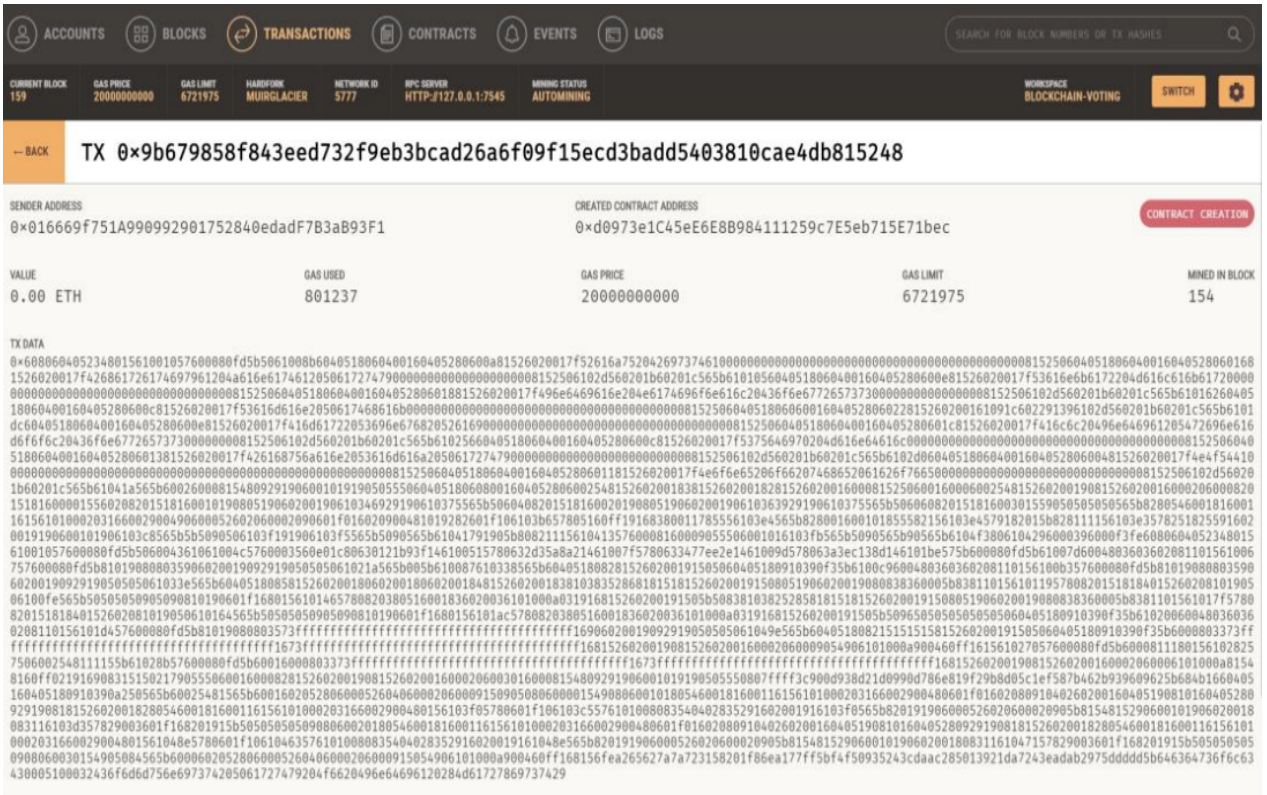


Figure 17: Contract Creation Transaction

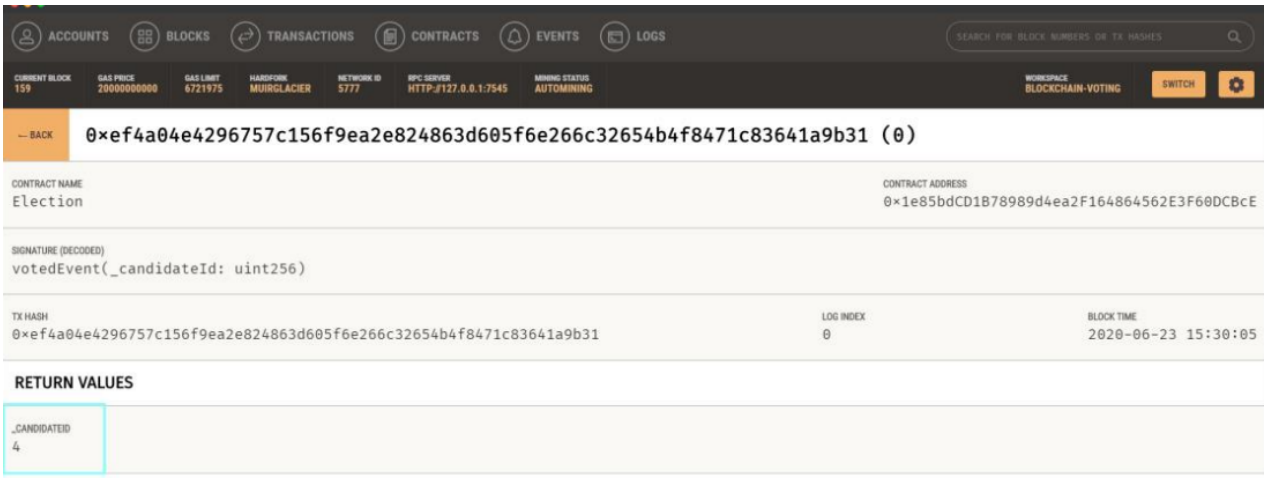
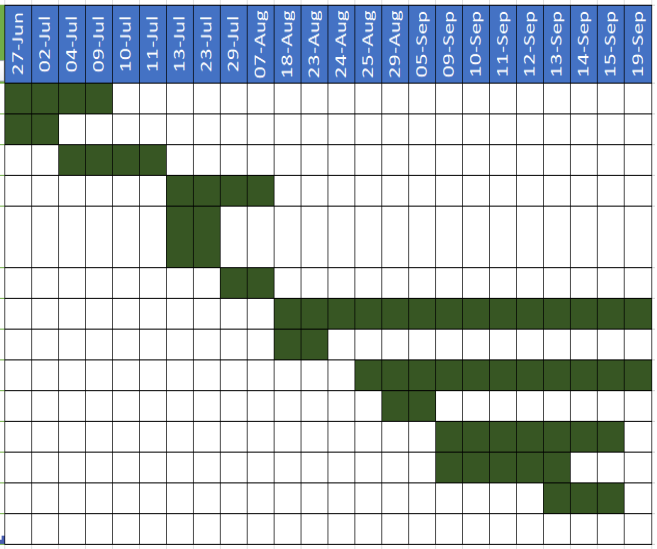


Figure 18:Voted Event Transaction

Gantt Chart

	Project Name	Project Duration(Days)	Project Start Date	Project End Date
	Decentralized E Voting System Using Blockchain	90	Jun-21	Sep-21

Task ID	Task Description	Task Duration (Days)	Start Date	End Date
2	Getting Started	12	27-Jun	Jul-21
3	Analysis	5	27-Jun	Jul-21
4	Requirement Gathering	7	04-Jul	Jul-21
5	System Design and Architecture	25	13-Jul	Aug-21
6	Detailed Analysis and Requirement Gathering	10	13-Jul	Jul-21
7	Prepare Design Documentation	15	29-Jul	Aug-21
8	Implementation	37	18-Aug	Sep-21
9	Setting up the environment	5	18-Aug	Aug-21
10	Development	25	25-Aug	Sep-21
11	Quality Testing	7	29-Aug	Sep-21
12	Deployment	6	09-Sep	Sep-21
13	Setup Production	4	09-Sep	Sep-21
14	Release	2	13-Sep	Sep-21



CHAPTER 4
CONCLUSION

Popular governments rely upon believed decisions and residents should trust the political race framework for a solid majority rules system. Anyway, customary paper-based decisions don't give reliability. Adjusting computerized casting ballot frameworks to make the public appointive interaction less expensive, quicker and simpler, is a convincing one in present-day culture. Making the appointive interaction modest and fast standardizes it according to the citizens, eliminates a specific force obstruction between the elector and the chosen official and comes down on the chosen official. It additionally opens the entryway for a more straightforward type of majority rules system, permitting citizens to communicate their will on individual bills and recommendations.

This undertaking has been formed into a blockchain-based electronic democratic framework that uses savvy agreements to empower secure and cost-proficient races while ensuring citizens' protection. It diagrams the framework’s engineering, the plan, and the security examination of the framework.

In the following form of this application, it has been proposed to make separate customer plans for different jobs, for example, one for the political race commission and one for up-and-comers enrolled to a specific party with the current democratic customer plan. Additionally, the current variants need validation as we don't approach current Aadhar's or Voter SDK to incorporate into our application. Likewise, it is arranged that in the following form notice brief will be allowed upon the arrival of casting a ballot to every one of the citizens to make their choice so the elector turnout is greatest for that political race.

REFERENCES

[1] Fouard, L., Duclos, M., Lafourcade, P.: Survey on electronic voting schemes. supported by the ANR project AVOTE (2007)

[2] Agora Technologies (2018). Agora - bringing our voting systems into the 21st century. https://www.agora.vote/s/ Agora Whitepaper.pdf. Accessed: 2018-07-25

[3] Brightwell, I., Cucurull, J., Galindo, D., Guasch, S.: An overview of the vote 2015 voting system (2015)

[4] Carter, L., Belanger, F.: Internet voting and political participation: an empirical comparison of technological and political factors. DATABASE 43(3) (2012) 26–46

[5] Barnes, Andrew, Christopher Brake, and Thomas Perry. "Digital Voting with the use of Blockchain Technology." Team Plymouth PioneersPlymouth University (2016).

[6] Caiazzo, Francesca, and Ming Chow. "A BlockChain Implemented Voting System." (2016).

[7] Wolchok, Scott, et al. "Security analysis of India's electronic voting machines." Proceedings of the 17th ACM conference on Computer and communications security.

[8]Ohlin, Jens David. "Did Russian cyber interference in the 2016 election violates international law." Tex. L. Rev. 95 (2016): 1579.

[9] Ayed, Ahmed Ben. "A conceptual secure blockchain-based electronic voting system." International Journal of Network Security & Its Applications 9.3 (2017): 01-09.

[10] Yu, Bin, et al. "Platform-independent secure blockchain-based voting system." International Conference on Information Security. Springer, Cham, 2018