INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 07 ISSUE: 05 | MAY - 2023 IMPACT FACTOR: 8.176 ISSN: 2582-3930

Implementation of Online Voting Using Blockchain

[1]Jeevan Raju, [2]Kamal Nayan, [3]Manik Chauhan, [4]Sahil Tagala, [5]Ms.Vijaylaxmi Inamdar
[1]Student, [2]Student, [3]Student, [4]Student, [5]Assistant Professor
[1]Department of Computer Science and Engineering
[1]Dayananda Sagar Academy of Technology and Management, Bengaluru, Karnataka, India

***Abstract*— Voting is the fundamental right of every nation. An Electronic Voting(E-Voting) system is a voting system in which the election process is notated, saved, stored, and reused digitally, which makes the voting operation task better than the traditional paper-grounded system. Blockchain is offering new openings to develop new types of digital services. While the exploration of the content is still arising, it has substantially concentrated on the specialized and legal issues rather than taking advantage of this new Conception and creating advanced digital services. Blockchain enabled-voting (BEV) could reduce name fraud and increase name access. Eligible choosers cast a ballot anonymously using a computer or smartphone. BEV uses a translated key and tamperproof particular IDs. Electronic credibility services have come an integral part of the information space. With the dependable Perpetration of introductory services similar to the electronic hand and electronic authentication, it's possible to make more complex systems that calculate on them, particularly the electronic voting system.**

In this design, the conception of developing an electronic voting system using blockchain technology is enforced. The two-position armature provides a secure voting process without redundancy of being (not grounded on the blockchain) systems. The blockchain-grounded voting design has two modules to make the whole design integrated and work along. One will be the Election Commission which will be responsible for creating choices, adding registered parties and campaigners querying for the election added under the smart contracts. The other end will be the name's module where everyone can cast a vote for their separate Assembly Constituency and the vote will be registered on the blockchain to make it tamper proof.

I. INTRODUCTION

Overview:
Modern democracies are erected upon traditional ballot or electronic voting(e-voting). In recent times, bias which is known as EVMs are monstrously blamed due to irregular reports of the election results. There have been numerous questions regarding the design and internal armature of these bias and how it might be susceptible to attacks. This paper has anatomized different ways of tampering the EVMs. Online voting is pushed as a implicit result to attract youthful citizens and the non-resident of the country. For a robust online election scheme, a number of functional and security conditions are to be met similar as translucency, delicacy, auditability, data sequestration. We've worked on the following ideas by having two different sets of modules election commission and the namer(s). The Election Commission creates choices and adds registered campaigners along with the parties for querying the election. Using an election's REST API hosted on Ethereum's Blockchain, the details are shown at the front- end of the namer for casting the vote. also, while polling the vote is stored on our blockchain frame of which the Election Commission fetches the vote count. The limitation which we've faced due to not using the traditional way of smart contracts is that the blockchain frame which we've enciphered can not run on the main net as it needs to be hosted and a separate web3 provider have to be used for interacting with it and not having a public API of namer ID creates a debit of not having authentication of a namer. The most important factor of this operation is to integrate the blockchain frame with both the modules for flawless voting.

Blockchain:
Blockchain is a decentralized, digital ledger that records transactions on multiple computers. These transactions are secured and validated through complex cryptography, ensuring that they are secure and cannot be altered. One of the key features of blockchain technology is its decentralized nature, which means that it is not controlled by any single entity or organization. Instead, it relies on a network of computers to validate and record transactions, which makes it extremely resistant to tampering and fraud. One of the most well-known applications of blockchain technology is in the creation of cryptocurrencies such as Bitcoin. In this case, the blockchain serves as a public ledger of all Bitcoin transactions, allowing users to transfer funds without the need for a central authority. Other potential uses of blockchain are in creating transparent supply chain systems, facilitating secure online voting systems and can even be used to verify important documents such as deeds and contracts. Pros of using blockchain like greater efficiency, transparency and security. However, this tech is still in its nascent phase, but a lot is possible with continuous innovation and growth in this field.

Objectives:
• To improve the existing online voting system using Blockchain technology.

• To reduce the workload of setting up EVMs and conducting elections in physical form.
• Reduction in vote tampering.

Merkle Tree:
A Merkle tree, also known as a hash tree, is a data structure used to efficiently verify the integrity of a large set of data. It is named after Ralph Merkle, who invented the concept in the 1980s to improve the security and efficiency of communication networks.
A Merkle tree is constructed by dividing a large set of data into smaller chunks, known as "leaves." These leaves are then hashed using a cryptographic hash function, which converts them into a fixed-size string of characters known as a "hash." Each leaf is paired with its corresponding hash, and these pairs are organized into a tree structure, with each parent node representing the hash of its child nodes.
The root of the tree represents the final hash of the entire data set and is known as the "Merkle root." This root can be used to verify the integrity of the data set, as any changes to the data will result in a different root hash.
One of the key benefits of Merkle trees is their ability to allow for the efficient verification of large amounts of data without the need to transmit the entire data set. Instead, a client can simply request the hashes of the relevant leaf nodes and the root hash and use these to verify the integrity of the data.
Merkle trees are widely used in various applications, including blockchain technology, file integrity verification, and secure communication protocols.

II. *EASE OF USE*

Online selection is accessible across a range of devices like smartphones, tablets, and computers, creating the whole selection process as straightforward as a click of a button or a faucet on a screen. This also means that elections are often accessed from anyplace within the world. Gone ar the times you pay a whole meeting attempting to pick a date once everyone seems to be in city for the election. For example, if members ar on vacation, they'll merely log into the software package and vote whereas lolling on the beach. this is often conjointly extremely useful for larger organizations with members living in numerous components of the country. It avoids the burden of having to find and farm out multiple polling sites or buy a large number of mail ballots. All members will forged their ballot from the comfort of their home throughout the selection hours.

1.Election creation: Election directors produce election ballots employing a decentralized app. This decentralized app interacts with Associate in Nursing election creation good contract, during which the administrator defines a listing of candidates and balloting districts.

2.Voter Registration: The registration of the citizen section is conducted by the election directors. The election directors should outline a settled list of eligible voters. This needs a part for a government biometric authentication service to firmly certify and authorize eligible people.

3.Vote Transaction: once a personal votes at a balloting district, the citizen interacts with a ballot good contract with constant balloting district as is outlined for someone citizen. This good contract interacts with the blockchain via the corresponding district node, that appends the vote to the blockchain if accord is reached between the bulk of the corresponding district nodes. Every vote is held as a dealing on the blockchain whereas every individual citizen receives the dealing ID for his or her vote for substantiative functions.

Each dealing on the blockchain holds info concerning whom was voted for, and therefore the location of said vote. Every vote is appended onto the blockchain by its corresponding ballot good contract, if and as long as all corresponding district nodes agree on the verification of the vote knowledge. Once a citizen casts his vote, the load of their billfold is remittent by one, thus not facultative them to vote over once per election.

4.Verifying vote: As was mentioned earlier, every individual citizen receives the dealing ID of his vote. every individual citizen will head to his government official and gift their dealing ID once authenticating himself victimisation his electronic ID and its corresponding PIN. the govt official, utilizing district node access to the blockchain, uses the blockchain mortal to find the dealing with the corresponding dealing ID on the blockchain. The citizen will thus see his vote on the blockchain, substantiative that it had been counted and counted properly

Scope of Project:

1. An election system mustn't change coerced pick.
2. An election system mustn't change the traceability of a vote to a voter's distinguishing credentials.
3. An election system ought to guarantee and proof to a citizen that the voters voting was counted and counted properly.
4. An election system mustn't change management to a 3rd party to tamper with any vote.
5. An election system mustn't change one entity to regulate over tallying votes and crucial an elections result.
6. An election system ought to solely enable eligible people to take AN election

Feasability Study:

i. A single evm machine can count only upto **2000** votes and cost Rs **17000**/- per unit and can scale upto billions for large number of populations.
ii. Every EVM is **air-gapped** meaning they are not connected to the internet similar the blockchain used will also be air-gapped and locally hosted on the system.
iii. It provides similar security and functionality but is scalable while being cost efficient.

iv. The blockchain can run on low powered system and will require less energy as comapred to its counterpart.
v. The blockchain stores data in a distributed network which is hard to tamper by.

Overall, the feasibility of using blockchain in online voting will depend on the specific requirements and constraints of the voting system in question, as well as the level of security and transparency that is desired.

III. *LITERATURE SURVEY*

There has previously been some noteworthy research in this area, which has been referenced in order to have a comprehensive understanding of the subject and grasp a few crucial ideas for this study. We referred to conference paper [2] to gain an overall idea on how the authors tried to highlight an existing problem of using EVMs. Research paper [5] was further referred to get a better view of how Blockchain works and whether it can be included in the study. We also referred to paper [4] to gain an idea from a study which focuses on the same problem. In addition, several other sources, many of which are listed below, are used to understand specific concepts by reading earlier research in the same area. The benefits and limitations of generic voting systems, blockchain security, blockchain structure, several current blockchain networks, and many other subjects were understood through consulting numerous relevant publications. This evaluation aided in the development of our own research and helped shed light on related issues.

IV. *FUNCTIONAL REQUIREMENTS*

Voters may use the system to cast their ballots from any location, and it is validated by EC and given the address and private key of an ETH wallet. The most important foundational elements of our blockchain voting system are security and anonymity. We may outline our presumptions and dependencies for the system's correct operation as follows:

- **Metamask Browser Extension:** Users may maintain their keys and accounts using a number of tools, such as hardware wallets, using Metamask while keeping them separate from the site environment.
- **Ganache:** Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.
- **Truffle:** A world class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier.
- **NodeJS:** It is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Table 2: Software Requirements

Software	Type	Version
Ganache	Ethereum Blockchain Server	2.4.0
Metamask	Ethereum Wallet	7.7.9
Truffle	Development framework for ETH	5.1.31
Node	JavaScript Runtime	12.17.0
Visual Studio Code	Integrated development environment	1.46
Remix	Solidity's IDE	0.10.1
Windows 10	Operating System	1809

METHODOLOGY

Proposed modules of work:
• **Module 1:** We will discuss the front-end module in this phase, where we will create the interactive user interface for both the admin and the user. In parallel, research will be conducted on the use of blockchain in decentralized applications.
• **Module 2:** In this phase we will cover the back-end module, we will implement the Blockchain using Ethereum framework and convert the system into a decentralized application.
• **Module 3:** The connection of two different modules along with the testing of the platform will be completed in this phase.

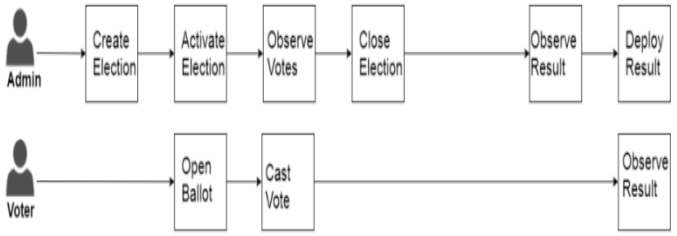

Fig: Voting procedure

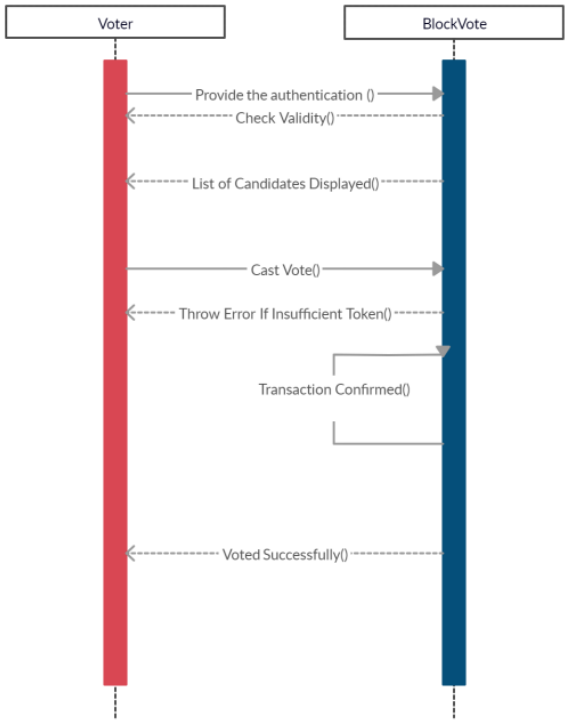

Fig: Sequence Diagram

Proposed Plan of Work:

For our proposed plan of work we are considering two modules that are to be completed in three phases. Two modules are as follows:

1. Front-end for the application

2. Back-end using Solidity to implement Blockchain. Each of these module will be considered as one phase and the remaining one phase will cover the connection and testing of these modules.

- Phase 1:

We aim to develop a front-end application/webpage using Javascript. This will enhance user experience.

It will contain information like: The different contesting parties and the contestants representing the parties.

This will give the voter an overview of available voting options and will ultimately help in casting the vote.

- Phase 2:

We aim to develop the backend part of our project.

1.Ganache:

Local development and testing

Mock blockchain data

Fast and lightweight

2.Truffle:

Smart contract development

Automated testing

Contract deployment

DApp development

When used together, Ganache and Truffle can help streamline the development process for blockchain projects by providing a fast and reliable way to test and deploy smart contracts and dApps. Ganache can be used as a local development environment, while Truffle provides tools for contract development, testing, and deployment.

- Phase 3:

In this phase, will connect the front to the backend modules.

We will use Metamask as a wallet address to initiate the voting process on our locally hosted blockchain.

It is a cryptocurrency wallet used to interact with the Ethereum blockchain.

It allows users to perform transactions on the blockchain by charging gas fees in terms of ether tokens.

Division of Phase One: We have considered 2 main modules which are as follows:

A. Admin- The admin module is divided into 5 components

1. Dashboard-It will contain various charts to display information such as number of parties, number of voters etc.

2. Add Candidate - In this feature of admin, he can add candidates who are standing in the election. After candidate is added it will be displayed on the user side.

3. Create Election- This feature of admin will allow him to create election. A user can cast his vote only after the election is created by admin. A user can cast vote between the start date and end date.

4. Election Details- In this section admin can update election details such as start date, end date etc.

5. Candidate Details- In candidate details all the candidates added by admin will be displayed. Admin can update the candidate details if in case a wrong entry is done.

B. User- The user module is divided into 4 components

1. Dashboard- The user dashboard contains information about parties and their candidates. A user can see all the information about candidate.

2. Voter Register- In this section first user will have to register himself only then he will be able to cast his vote.

3. Voting Area- After user is registered, then only he will be directed to this page and then he can cast his vote.

4. Results- In this component the user will be able to see the results of the election.

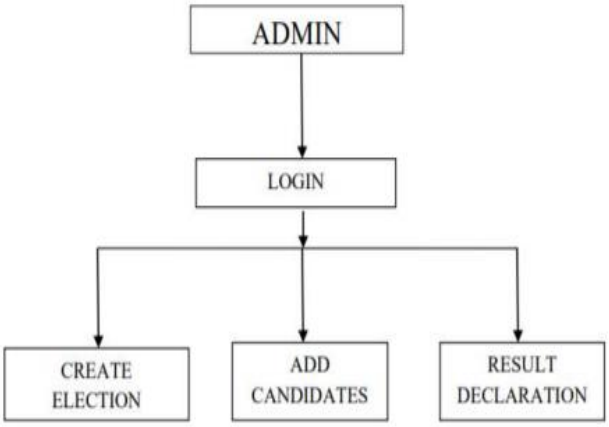

Fig.3.1 Admin flow diagram

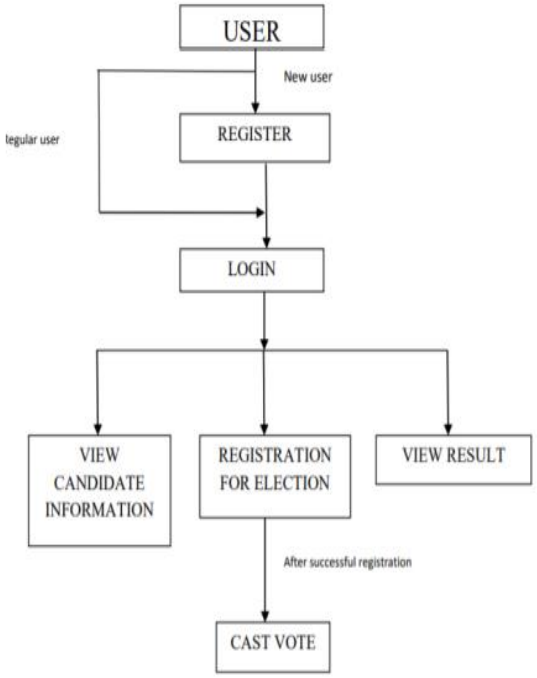

Fig.3.1 User flow diagram

User flow diagram

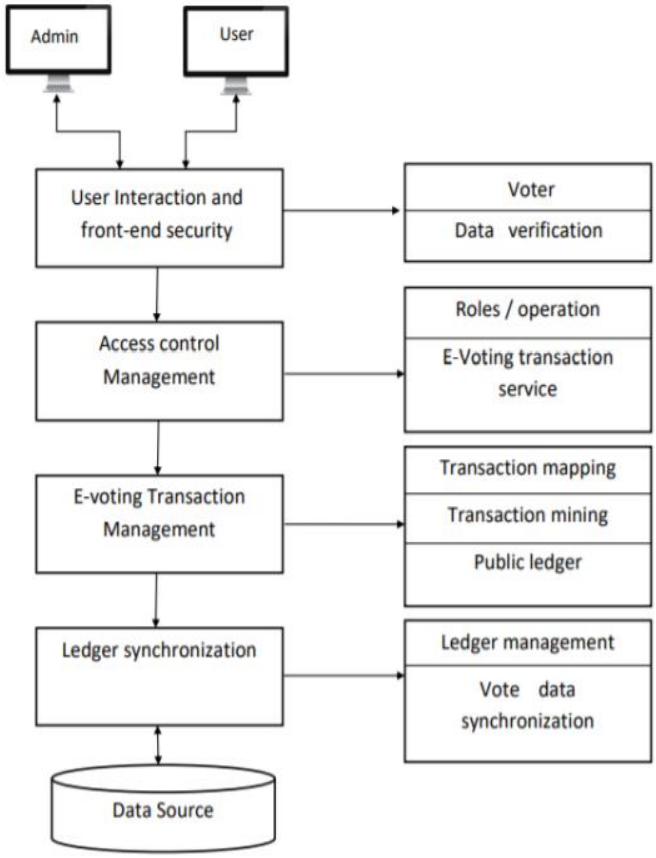

Fig.3.3 Research methodology

Implementation

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

Client: Client is any user or program that wants to perform an operation over the system. Clients interact with the system through a presentation layer.

Presentation Layer: This layer is responsible for the presentation of data at the client side, i.e., it provides an interface for the end-user into the application to cast the votes.

Resource manager: The resource manager deals with the organization (storage, indexing and retrieval) of the data necessary to support the application logic. This resource manager here is the Local Blockchain server maintained by Ganache.

Application logic: The application logic figures out what the system actually does. It takes care implementing the business rules and establishing the business processes. Blockchain voting system is designed and implemented according to the three tier architecture.

Testing

This project uses *Mocha* as the testing framework to unit test and integration test all of our test cases for the application. Following strategies are used:

(i) **Unit Testing:** This is the first and the most important level of testing. Its need begins from the moment a programmer develops a unit of code. Every unit is tested for various scenarios. Detecting and fixing bugs during early stages of the Software Lifecycle helps reduce costly fixes later on. It is much more economical to find and eliminate the bugs during early stages of application building process. Hence, Unit Testing is the most important of all the testing levels. As the software project progresses ahead it becomes more and more costly to find 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:** Integration strategy stands for how individual modules will be combined during Integration testing. The individual modules can be combined in one go, or they can be joined one by one. A decision on how to put the pieces together is called the Integration Strategy.
We have used bottom-up integration approach to integrate test our application.
In Bottom Up Integration, we move from the bottom to top i.e. the components below are first written and these are integrated first. The integration happens from bottom to top. If the calling component is yet to be developed, it is replaced by a specially written component called a Driver.

Candidate Count Unit Test

Double Voting Unit Test

```
//Checks for Invalid Candidates

it("throws an exception for invalid candiates", function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      return electionInstance.vote(99, { from: accounts[1] })
    }).then(assert.fail).catch(function(error) {
      assert(error.message.indexOf('revert') >= 0, "error message must conta
      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, 0, "candidate 2 did not receive any votes");
    });
  });
```

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");
    });
  });
```

Candidate Initialization 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 candi
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");
    })
  });
```

Vote Cast Unit Test

```
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-202069-28610-17j3ypn.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)
```

Test Report

CONCLUSION

Democracies depend on trusted elections and citizens should trust the election system for a strong democracy. However traditional paper-based elections do not provide trustworthiness. The idea of adapting digital voting systems to make the public aware of making the electoral process cheaper, faster and easier, is a compelling one in modern society. Making the electoral process cheap and quick normalizes it in the eyes of the voters, removes a certain power barrier between the voter and the elected official and puts a certain amount of pressure on the elected official. Acts as a direct form of democracy, allows voters to express their opinions on individual bills and propositions. This project has been developed into a blockchain-based electronic voting system that utilizes smart contracts to enable secure and cost-efficient elections while guaranteeing voters privacy. It outlines the systems architecture, the design, and a security analysis of the system.

REFERENCES

[1] Benjamin B., Bederson, Bongshin Lee., Robert M. Sherman., Paul S., Herrnson, Richard G. Niemi.,
[2] Security Analysis of India's Electronic Voting Machines
[3] Chaum D., "Secret-ballot receipts: True voter-verifiable elections", IEEE Security and Privacy, 2(1):38-47, 2004.
[4] Blockchain-Based E-Voting System
[5] Blockchain White Paper
[6] Gritzalis D., [Editor]., "Secure Electronic Voting", Springer-Verlag, Berlin Germany, 2003.
[7] Harris B., "Balck Box Voting: Vote Tampering in the 21st Century", Elon House/Plan Nine, July 2003.
[8] Jones D. W., "The case of the DieboldFTP Site", THE UNIVERSITY OF IOWA Department of Computer Science, July 2003.
[9] "Electronic Voting System Usability Issues", In Proceedings of the SIGCHI conference on Human factors in computing systems, 2003.
[10] Darcy, R., & McAllister, I., "Ballot Position Effects", Electoral Studies, 9(1), pp.5-17, 1990
[11] Dill D.L., Mercuri R., Neumann P.G., and Wallach D.S., "Frequently Asked Questions about DRE Voting Systems", Feb.2003.