21C17

*by* C 17

**Submission date:** 22-Jul-2021 10:20 AM (UTC+0530)

**File Name :** C17.docx (3.67M)

**Word count:** 6517

**Character count:** 25190

**A ­Project report on**

# **AN E-VOTING SYSTEM BASED ON BLOCKCHAIN**

***Submitted by***

**BIKASH RANA MAGAR (17B91A05P5)**

**ANUSHA KAYASTHA (17B91A05P3)**

**GAUTAM PRASAD SAH (17B91A05P6)**

***Under the guidance of***

**Dr. SRIHARI VARMA MANTENA**

*Assistant Professor*

***In partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**S.R.K.R ENGINEERING COLLEGE**

ChinnaAmiram, Bhimavaram, West Godavari Dist., A.P

[2017-2021]

**Appendix 2**

**SRKR ENGINEERING COLLEGE (A)**

**BONAFIDE CERTIFICATE**

This is to certify that the project work entitled **“AN E-VOTING SYSTEM BASED ON BLOCKCHAIN”** is the bonafide work of **“BIKASH RANA MAGAR (17B91A05P5), ANUSHA KAYASTHA (17B91A05P3), GAUTAM PRASAD SAH (17B91A05P6)”** who carried out the project work under my supervision.

**SUPERVISOR**

Dr. SRIHARI VARMA MANTENA

Assistant Professor

**HEAD OF THE DEPARTMENT**

(Dr. V CHANDRA SEKHAR)

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** |  |
| Abstract | i |
| List of Figures | ii |
| List of Tables | iii |
| 1. Introduction | 1 |
| 1. Problem Statement | 4 |
| 1. Literature Survey | 5 |
| 1. Software Requirements and Specifications    1. Purpose    2. Scope    3. Definitions and Acronyms    4. Existing System    5. Proposed System    6. Requirements       1. Functional Requirements       2. Non-Functional Requirements 2. System Design    1. System Architecutre    2. Overview    3. Design Goals    4. Modularization Details    5. Blind Signature    6. UML Design       1. Use Case Diagram       2. Class Diagram       3. ER Diagram       4. Sequence Diagram       5. Activity Diagram       6. Statechart Diagram       7. Collaboration Diagram 3. Methodology    1. Blockchain    2. Ethereum    3. Ethereum Blockchain    4. Mining and Ether    5. Transaction cost and Gas    6. Smart Contracts 4. Implementation    1. Tires       1. Client Layer       2. Presentation Layer       3. Resource Manager       4. Application Logic    2. User Interface Design 5. Testing    1. Unit Testing    2. Integration Testing    3. Testing Designs    4. Test Results 6. Conclusion and Future work 7. References 8. Appendix    1. Source code | 8  8  8  9  9  9  10  10  11  12  12  12  13  16  17  18  19  20  21  22  23  23  24  24  25  25  26  26  26  26  27  31  31  32  35  36  37  38 |

**ABSTRACT**

An electronic voting system ensures the end-to-end verifiability that the voter can uphold whether the vote was correctly counted or not and also anyone who wants to verify the result, they are allowed to verify. We can use several proposals delimiting potential systems, however these all systems are built over the top of protocols mainly called as transactional ledgers .In this paper a voting alternative built over the Ethereum protocol is introduced, which uses smart contracts which impose strict rules acquiring the ballot security. Hence, these ballots are both individually and universally verifiable and maintains the expected quality of the blockchain (immutability). All of these can be achieved without losing voters privacy, anonymity and ballot security. Hence, the blockchain technology has been proposed to become central part of the applications ensuring transparency and security in public scenario.

**List of Figures**

1. System Architecture 11
2. Blind Signature Analogy 14
3. Use Case Diagram 16
4. Class Diagram 17
5. ER Diagram 18
6. Sequence Diagram 19
7. Activity Diagram 20
8. StateChart Diagram 21
9. Collaboration Diagram 22

**List of Tables**

1. Definitions and Acronyms 7
2. Software Requirements 9

**Chapter 1**

**INTRODUCTION**

Existing electronic voting systems are centralized in nature i.e. there is only one party that has the control over the system. They have all accessibility over the database and they can manipulate the outcomes as the need. Due to this, there is lack of an independently verifiable system where once voters have determined their ballot option, they must put their trust to the organization whose vote is registered and counted as intended. Lack of free verifiable output makes it difficult for these centralized systems gaining the credibility that voters expect, which limits the voter participating or casting doubt on the published result of an election.

Although many aspects of modern life have been digitalized, elections are still conducted offline, on paper. Although the electronic Voting machines have been steadily increasing in recent years, Paper ballot are still major tools for voting and are usually marked by humans (Voters) and then counted by machine. To run the system, the voters must rely on physical security and rely on polling station to not manipulate them and handle them properly. Postal votes also use ballot papers to vote and voters must physically be present in one place. These also suffer from similar defects as traditional paper ballots, and increases the chances of attack during their passage through the postage system.

Voting system consist of five main components:

1. Registrations for registering the voters
2. Voter authentication to authorize the voters
3. Voting station where the voters vote
4. Ballot box where the ballot is collected
5. Tallying service to count the votes and make announcement

Current electronic voting (e-voting) takes two forms; using the machine in Polling station instead of ballot paper and pencil or to vote through the Internet. The previous refers to a Direct Recording electronic system. It usually displays the ballot option on the screen that is activated by the voter. Then it records the polling data to be processed into memory components next. However, as with many electronics, there is same flaw, ability to manipulate software and insert the malicious code. It is an issue raised in several recent elections and in a 2015 study, 43 U.S. states are found to use electronic voting machines at least 10 years in the last presidential election.

While voting over the internet, there is always the possibility of issues such as privacy, fraud, and voting under pressure and corruption, it does nothing to improve voter confidence and trust. Voters must assume that the vote they made are recorded and counted with honesty. In order to make the proposed E-voting system to be more efficient and fair challenger with the traditional polling methods, it should be able to perform at least the same level of existing systems, but with more improvements offering significant benefits. There are several standard requirements that voting systems must be compliant, each with equal security, privacy, functionality, usability and accessibility. A "Secure" voting means it cannot be tampered and manipulated at any aspect, ensuring each vote is efficiently and accurately recorded. It must also ensure that there are no extra votes and the votes are not manipulated at any stage of the process. The System functionality is comprehensive but must include; Proper registration and recording of all the votes cast, allowing the voter to vote for any candidate who has the right to vote, allowing only eligible registered voters to vote and allow each voter to vote only once. The right to a secret ballot belongs to the electorate and to cast his vote privately. This is necessary to protect voters being forced or bribed to vote in a certain way, i.e. the system should not provide receipt or another person to make a content of voter’s ballot. On top of this, the system should be simple and easy for voters to operate allowing any kind of voters to be able to vote.

Although there are many functionalities and services provided in the existing voting system, there are many improvements and additions that this system is willing to add and explore. The first smart move is using blockchain to log votes due to its decentralized nature. This means that there is less need to be focused and have trust on organizations where votes are hidden and has only access to them. It also has a property that once the transaction is complete, it is very difficult to change or manipulate once the transaction is verified, attackers must have at least 55% computing power over the network to try fraudulent transactions. Any attempts of manipulating the block, the duplicate block will be easily viewed and ignored by the rest of the network. Transactions in blockchain lead to high level of confidence and transparency in the election. The result is further strengthened by independent verification which can be performed by anyone. Once the transaction (voting process) is completed and block is added to the blockchain, it becomes almost impossible to any person to manipulate the information stored in the blockchain. The total result of a polls are stored indefinitely and can be accessed at any point of time in the future.

The verification features of electronic voting are divided into two categories. "Individual - Verifiable", which includes auditing of voting processes and Vote storage by voter; And "Universal Verifiable", which confirms votes are only from the eligible voters and are safely collected in the ballot box and all the votes are collected and properly matched Exactly matched, anyone can do this . Providing the system, both types of verifiable systems are called "end-to-end verifiable systems".

One of the personal verifiable attributes is cast-as-intended, it focuses on auditing the vote creation process. There is another property i.e. Recorded-as-cast verification intended to audit proper reception and storage of Votes on the remote voting server.

In this paper, the design of the “end-to-end verifiable system” build over the Ethereum blockchain is explained. In this system, first voter goes to the registration process and only those voter who is eligible to vote are given Ethereum address and are added to the list (which is nothing but the list of allowed addresses) over the smart contract. Upon the ballot commencing, the voter can modify the allocation of their vote inside this contract up to the duration of ballot ending. Hence the programmable nature of Ethereum contracts, it ensures that no votes are manipulated or tampered after the ballot end date, only those who have the right over the Ethereum address can change the vote and only those associated Ethereum addresses are allowed to vote. And anyone can go through this transaction by querying the contract and verifying the number of votes and they can be assured that only legit addresses are added to the contract and able vote. Hence the voters can be confident that their votes are secured and not manipulated during the process.

**Chapter 2**

**PROBLEM STATEMEMT**

Since the dawn of democracy, elections have been accused for the lack of transparency, privacy, security and single point of failure. As societies all over the world are rapidly adopting technology across all aspects, digitalized democratic system of voting might just be the next evolutionary step towards a transparent and trusted electoral system ensuring Ballot-Privacy, Coercion-Resistance, individual-Verifiability and Universal-Verifiability. Since, all these features can be acquired with the blockchain technology, an E-Voting system based on blockchain technology has been introduced.

**Chapter 3**

**LITERATURE SURVEY**

Nearly a decade ago, Satoshi Nakamoto, an unknown person / group behind Bitcoin, described how a distributed peer-to-peer linked-structure, such as blockchain technology, managed issues such as ordering transactions and avoiding double -spending can be used for problem (Nakamoto, 2008). Bitcoin orders transactions and bundles them into a limited-volume structure called a single timestamp sharing block. The nodes (miners) of the network are responsible for connecting blocks to each other over time, each block containing the hash of the previous block, which forms the blockchain (Crosby et al., 2016). Therefore, the blockchain structure integrates a robust and auditable registry of all transactions.

In this paper [2], it has highlighted about the major problem in voting security where in the 2016 US Presidential Elections, EVM’s were likely to be intercepted and votes were tampered. The study found that this old voting equipment is not only more prone to failures and crashes but is also notoriously easy to hack and tamper with. In this study [3] by Ayed, Ahmed, et al., it has been proposed an electronic voting system based on the Blockchain technology. The system is decentralized and does not rely on trust. Any registered voter will have the ability to vote using any device connected to the Internet.The Blockchain will be publicly verifiable and distributed in a way that no one will be able to corrupt it. Rifa and Budi has come to a conclusion that if we use of hash values in recording the voting results of each polling station linked to each other makes this recording system more secure and the use of digital signatures makes the system more reliable. The use of the sequence proposed in the blockchain creation process in this system considers that in an electoral system not required for mining as in the Bitcoin system because the voter data and numbers are clear and are not allowed to select more than once, the proposed sequence ensures that all nodes Which is legally connected and can avoid collision in transportation [4]. Bin, Joseph, et al., has come to a conclusion that the current blockchain voting system cannot provide the comprehensive security features, and most of them are platform dependent, we have proposed a blockchain based voting system that the voters’ privacy and voting correctness are guaranteed by homomorphic encryption, linkable ring signature, and PoKs between the voter and blockchain[5]

Recently, blockchain like distributed-ledger technology has been used to achieve e-voting systems mainly due to their advantages in terms of end-to-end validation. With structures like anonymity, privacy protection and ballot privacy, blockchain is a very attractive method in modern voting systems. The research presented in this paper also tries to apply this blockchain structures to achieve an effective voting system. Various research of E-voting has been carried out in recent past years with the blockchain technology:

1. India's Online Voting System Based on AADHAR ID -Himanshu Agarwal, G. N. Pandey in 2013
2. Adhar based biometric voting in india-S Chakraborty, S Mukharjee in 2016
3. Reliable electronic voting Using a modified Blockchain Technology - Basin Shahzad Raju, Jon Crowcroft annually 2019
4. Indian Voting Machine Security - Hari K.Prasad, Arun Kankipati, Sai Krishna Sakhamuri year 2010

**Chapter 4**

**SOFTWARE REQUIREMENTS AND SPECIFICATIONS**

* 1. **PURPOSE**

Elections play very important role in democracy because they are a decisive factor in the future of the country but the major issue is that the public does not trust the election system. The flawed election system is a global problem, the largest democracies like India, United States and Japan. Over time, voting systems have changed and violated security has changed. Big problems that need to be mentioned in the current voting system are vote fraud, EVM hacking, booth capture and election fraud.

Hence, the blockchain technology has been proposed to become central part of the applications ensuring transparency and security in public scenario.

* 1. **SCOPE:**

Electronic voting or E-voting has been used in various ways since the 1970's basically advantages in paper-based programs such as increased efficiency and reduced errors. However, the challenges remain to gain widespread acceptance of such programs especially in terms of improvement their resilience in the face of possible mistakes. Blockchain is a disruptive technology of the present and promises to improve the overall durability of E-voting systems. This paper provides an effort to take advantage of blockchain as the basis for cryptographic and transparency to achieve an effective release system. The proposed system is in line with the basic requirements of E-voting schemes reaching end-to-end verification. This document sets out the details of the proposed voting system by its implementation using the Multichain platform. This paper provides an in-depth review of the program effectively demonstrating its effectiveness in implementing the final voting verification system.

.

* 1. **DEFINITIONS AND ACRONYMS:**

|  |  |
| --- | --- |
| **Abbreviations** | **Definitions** |
| EC | Election Commission |
| ETH | Ethereum |
| API | Application Programming Interface |
| IDE | Integrated Development Environment |
| JSON | JavaScript Object Notation |
| SRS | Software Requirement Specifications |
| SDLC | Software Development Life Cycle |
| STLC | Software Testing Life Cycle |
| PERT | Program Evaluation Review Technique |

* 1. **CURRENT SYSTEM:**

Current electronic voting systems are centralized in nature i.e. there is only one party that has the control over the system. They have all accessibility over the database and they can manipulate the outcomes as the need. Due to this, there is a lack of an independently verifiable system. Once voters have determined their ballot option, they must put their trust to the organization whose vote is registered and counted as intended. Lack of Free verifiable output makes it difficult for these centralized systems gaining the credibility that voters expect, which limits the voter participating or casting doubt on the published result of an election.

* 1. **PROPOSED SYSTEM:**

This proposed system presents an effort to leverage benefits of blockchain such as cryptographic foundations and transparency to achieve an effective scheme for E-voting. The proposed scheme conforms to the fundamental requirements for e-voting schemes and achieves end-to-end verifiability. The paper presents details of the proposed e-voting scheme along with its implementation using Multichain platform. The paper presents in-depth evaluation of the scheme which successfully demonstrates its effectiveness to achieve an end-to-end verifiable e-voting scheme. Hence the blockchain technology has been proposed to become central part of the applications ensuring transparency and security in public scenario.

* 1. **REQUIREMENTS ANALYSIS:**
     1. **FUNCTIONAL REQUIREMENTS**
* Authorized voters should be able to vote.
* The voters should be able to verify whether their vote has been counted or not.
* The vote must be immutable which blocks any kind of third party involvement.
* The result should be the same as the number of voter, voted by authorized voter.
* No other people can refer to their vote than themselves.
* The vote should be fairly counted after the election.
* The voting process must be verified by all the participation in the voting process.
  + - 1. **Software Requirement**

|  |  |  |
| --- | --- | --- |
| **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 |

Table 2: Software Requirement

* + 1. **NON-FUNCTIONAL REQUIREMENTS**
       1. **Performance Requirement**

The system is expected to have a reasonable short time of response. The voter should be able to import his/her wallet provided by the Election Commission within few seconds keeping in the mind the condition of network stability. The system’s performance is different according to its modes:

1. **Election Mode:** In this phase, the expected time to deploy the smart contracts totally depends upon the miners connected to the blockchain and the amount of GAS we decide to sign off the transaction to marked as validated one but as we are working locally, it is just a matter of half a minute or so.

1. **Voting Mode:** In this phase, the system will be responding within seconds as we don’t have to sign off transaction just to fetch the list of candidates for the elections but depending on the network stability and web3 connection the above performance might be delayed. Next, after casting the vote it might take a minute or two to sign off the transaction depending **upon the miners and GAS limit.**

**4.6.2.2 Security Requirements**

* The data transaction between client and the blockchain server must be done over https to avoid mixed content attack.
* The reentrancy on a single function has to be minimized while deploying the smart contract.
* To address the integer overflow error, the idea of counting the votes have been done within a specific event responsible for it.

**4.6.2.3 Reliability**

* **In Election Mode:** The system needs to be maintained time to time as if the smart contract which is to be deployed encounters any bugs, it needs to be fixed to prevent votes miscalculation and transaction error handling.
* **In Voting Mode:** As the maintaining part is in the Election Mode, if there’s any error in web3 connection the interoperability status might change otherwise the system will work flawlessly all the time.

**4.6.2.4 Usability**

* The system will have a minimal and simple User Interface.
* To guide the users for the first time using it, there will be a guidance related to the usage of the system.
* **HARDWARE REQUIREMENTS:**

1. RAM : 4GB
2. Processor : Intel core i3
3. Hard Disk : 500GB
4. OS : Windows 10, Linux, Mac
5. High speed internet

**Chapter 5**

**SYSTEM DESIGN**

* 1. **System Architecture**

It involves various steps that are done throughout the system design.

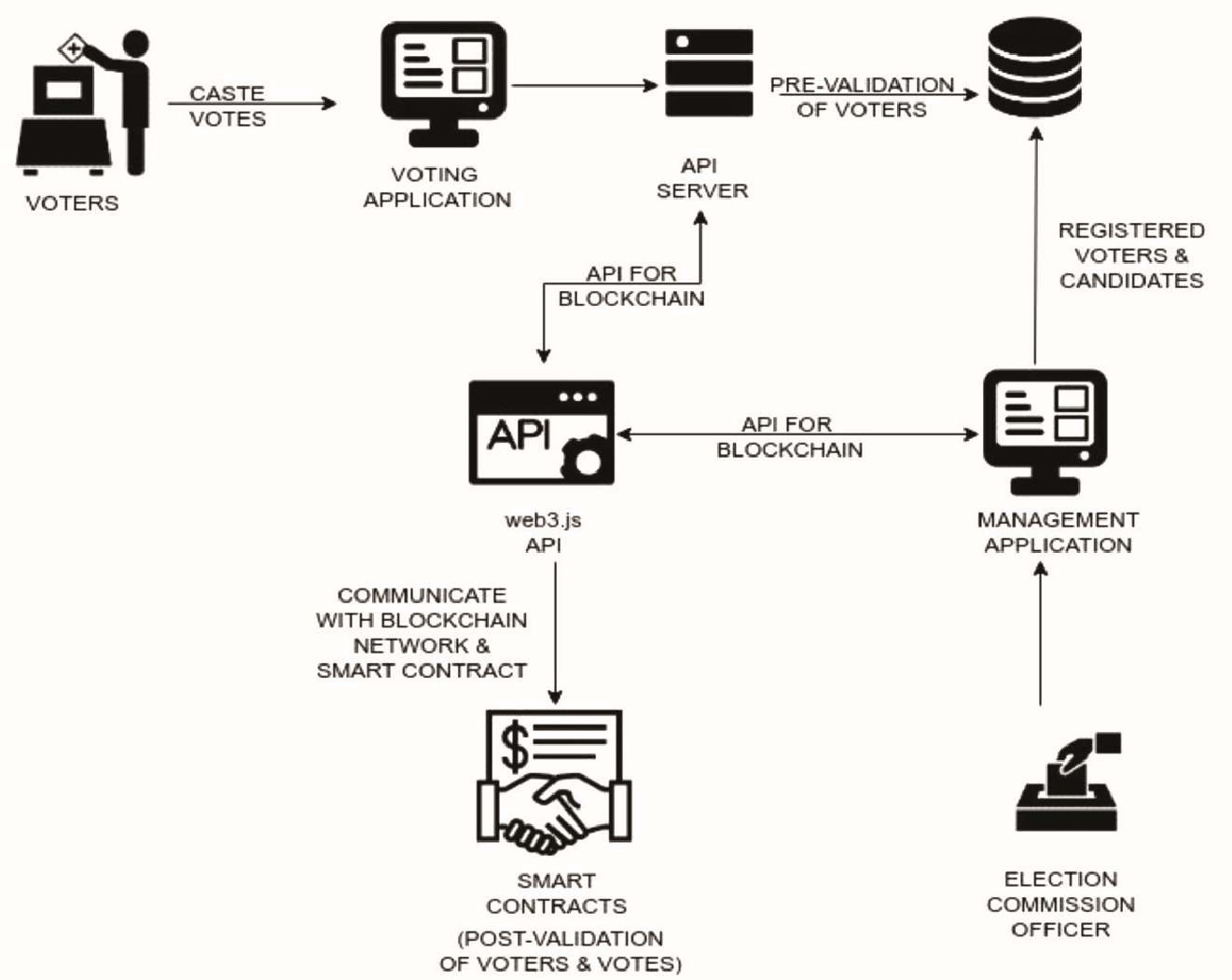


Figure 1: System Architecture

* 1. **Overview**

Although this system has been designed and developed for general election, this idea could also be applied to smaller scale ballots which wish to provide transparency in their audit. Although we want to avoid central authority, due to the nature of elections (where there must be some degree for voter eligibility verification), (where there needs to be some degree of voter eligibility verification), we will not be able to make out system fully decentralized. While we need to reassure someone, we still need to make sure that their votes are anonymous, especially if you are given a public action that supports the concept of blockchain while empowering a person to ensure that their vote was properly calculated. The designed schema for this protocol is the following:

1. **Ballot Creation**

* A smart contract is created and pushed onto the blockchain for all the ballots.

1. **Pre-election voter verification**

* Voter registers to the external voter registrar with a valid ID.
* This external registrar generates a user id and nonce which is used by the Voter to log in to the system.

1. **Voter registration:**

* The Voter logs into the system with the user id and nonce.
* The voter registers to vote.
* Unique Ethereum address is then created and validated.
* This address is added to the smart contract with which voter votes.
* The address is funded to the ether as a gas fees to complete transaction.

1. **Voting:**

• During the voting process, Voters can vote with the interface mirroring the options in the ballot smart contract.

• With the voter selecting the options, the contract is funded with the voters selected options.

• The Voters choice is immutably entered into the blockchain and the tally is verifiable by all.

1. **Election result:**

• Once the voting process is over due to the blockchain property no extra vote can be added.

• The transaction can be verified by everyone who are involved in the voting.

* 1. **Design Goals**

Design goals are important properties of the system to be optimized, and which may affect the overall design of the system. There is a fine line between system design and requirements. Requirements include specific values that must be met in order for the product to be acceptable to the client, whereas design goals are properties that the designers strive to make "as good as possible", without specific criteria for acceptability.

* 1. **Modularization Details**

The project has been divided into many modules in which for every functionality we have designated modules. Any software comprises of many systems which contains several sub-systems and those sub-systems further contains their sub-systems. So, designing a complete system in one go comprising of each and every required functionality is a hectic work and the process can have many errors because of its vast size.

Effective modular design can be achieved if the partitioned modules are separately solvable, modifiable as well as compliable. Following are the project modules:

1. **Election Commission:** In this module, an entity named Election Commission will be responsible to setup the smart contract and register candidates, parties and start off an election.

1. **Election Test:** This is the module to test our smart contract where we use Mocha Framework to perform unit test on our application.

1. **Voter Module:** In this module, voters who have been provided with the personal ETH wallet will import onto the voting portal using the Metamask extension and cast their vote.
   1. **Blind Signatures**

"Blind Signature" is a signature system in which the signature may not contain the content of the message you are signing but the resulting signature can be verified with the first original unblinded message as a standard digital signature. This is then handed to a trusted third part, Bob, who (without opening the letter) signs the outside of the document and hands it back to Alice. D The carbon paper inside the envelope, Bobs' signature is also transferred to the letter inside. Alice could then publish a letter now with Bobs' signature even though she had never seen the contents of the letter.

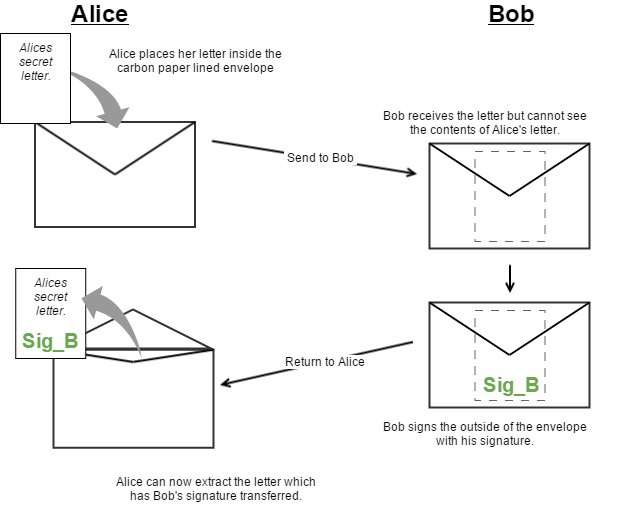


Figure 2: Blind signature analogy showing how Bob never sees the contents of Alice’s message despite being able to sign it.

Now we can try to translate this into cryptography. Suppose Alice has a message that she wants to sign by Bob, and she does not want to know bob about m. Let (a, b) be the Bob's public key and (a, c) be his private key. Alice generates a random value l such that gcd(l, a) = 1 and sends k = (l bm) mod n to Bob. The value k is “blinded” by the random value l ; hence Bob can derive no useful information from it. Bob returns the signed value t = k c mod a to Alice. Since k d ≡ (l bm) c ≡ rmd mod a, Alice can obtain the true signature s of m by computing s = l −1 t mod a. Now Alice’s message has a signature she did not obtained on her own.

* 1. **UML DESIGN:**

The Unified Modeling Language is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system.

Each Unified Modeling Language diagram is designed to let developers and customers view software system from a different perspective and in varying degrees of abstraction.

UML diagrams commonly created in visual modeling tools include:

1. Use Case Diagram
2. Class Diagram
3. ER Diagram
4. Sequence Diagram
5. Activity Diagram
6. Statechart Diagram
7. Collaboration Diagram
   * 1. **USE CASE DIAGRAM:**

Use Case diagram summarizes about a number of relationships between use cases, actors and systems.

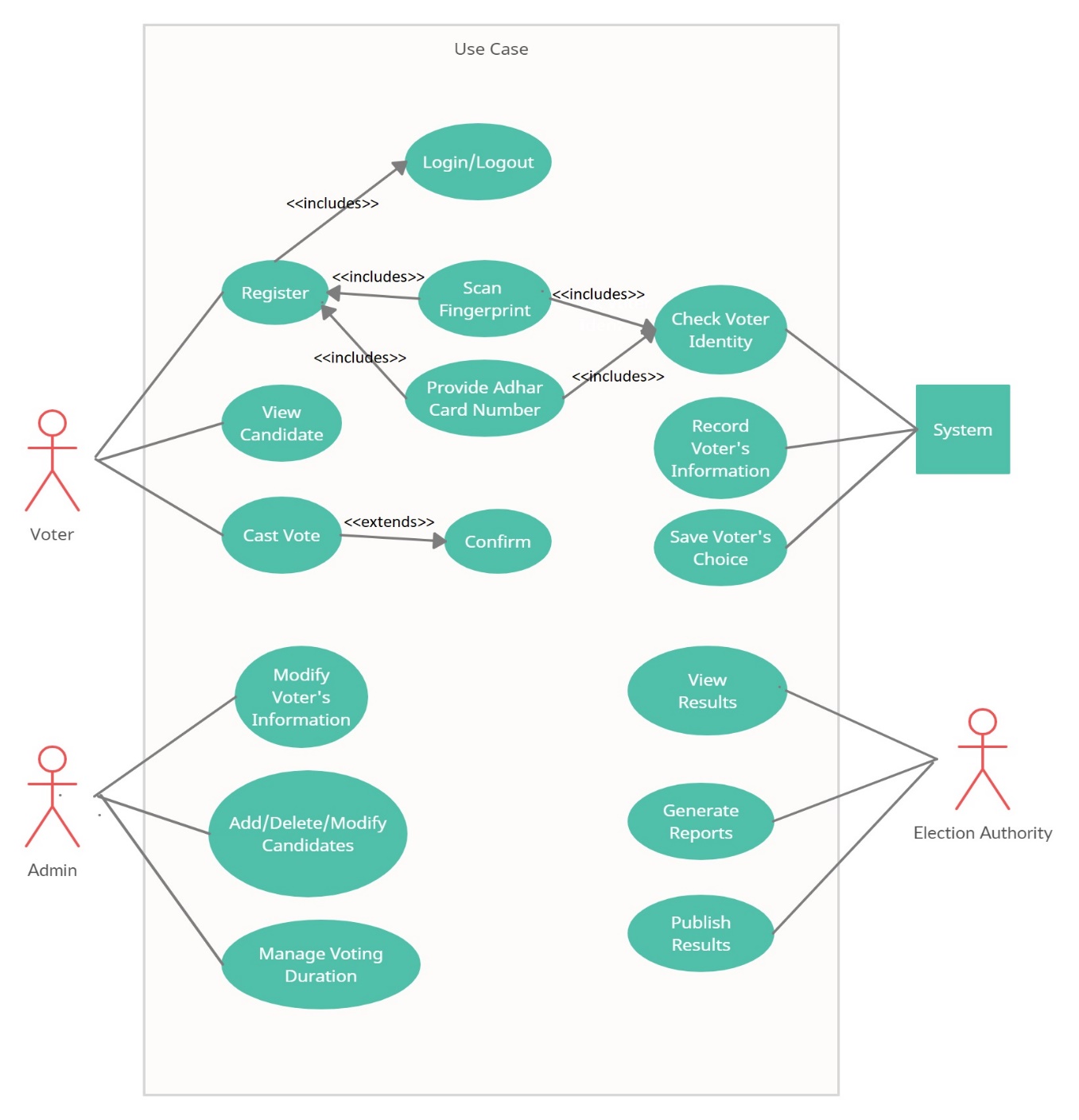
****

Figure 3: Use Case

* + 1. **CLASS DIAGRAM:**

The class diagram is the diagram that which classifies the actor that which is defined in the use case diagram into a set of inters related classes.

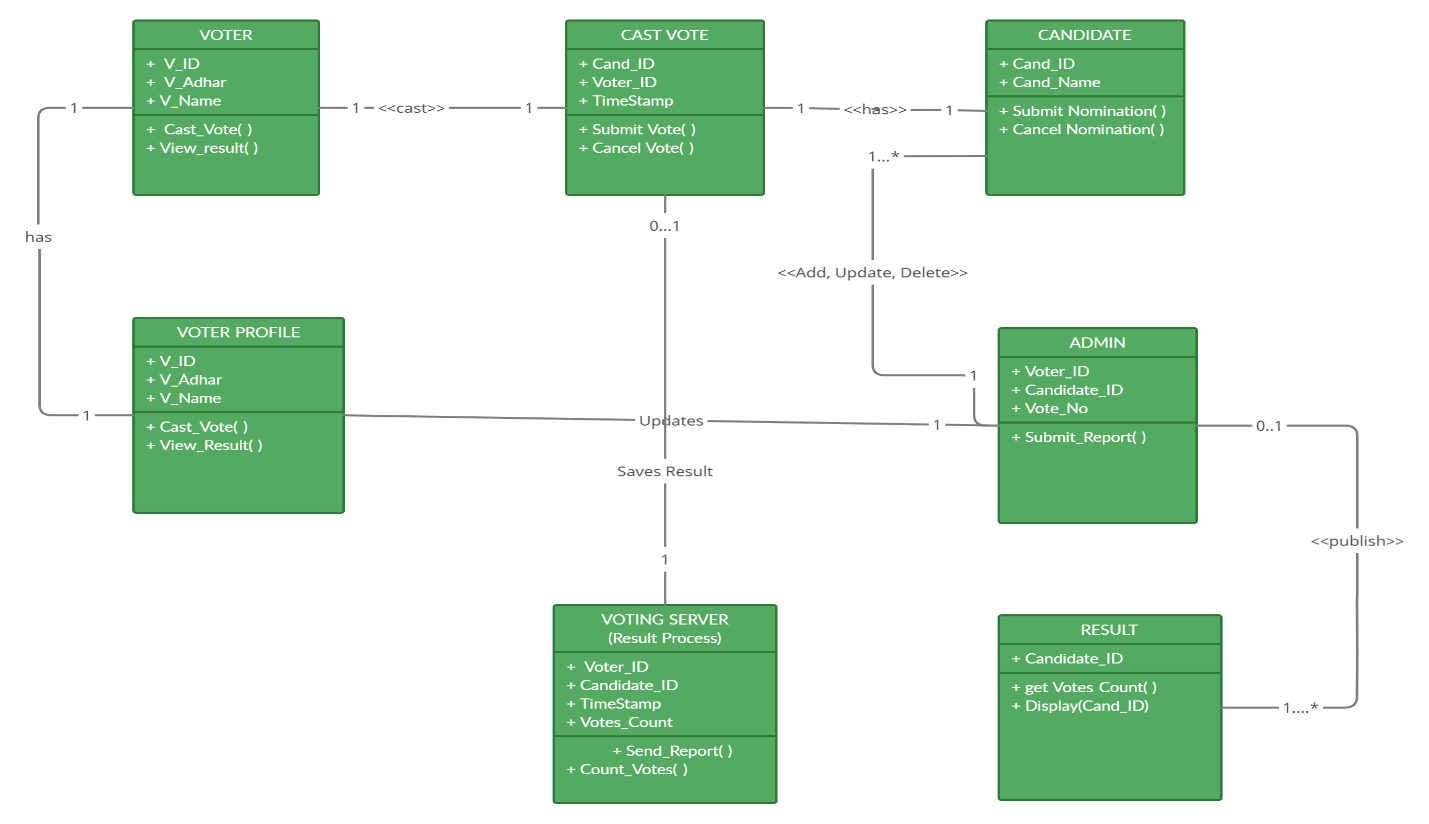


Figure 4: Class Diagram

* + 1. **ER DIAGRAM**

An ER diagram shows the relationship among entity sets. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

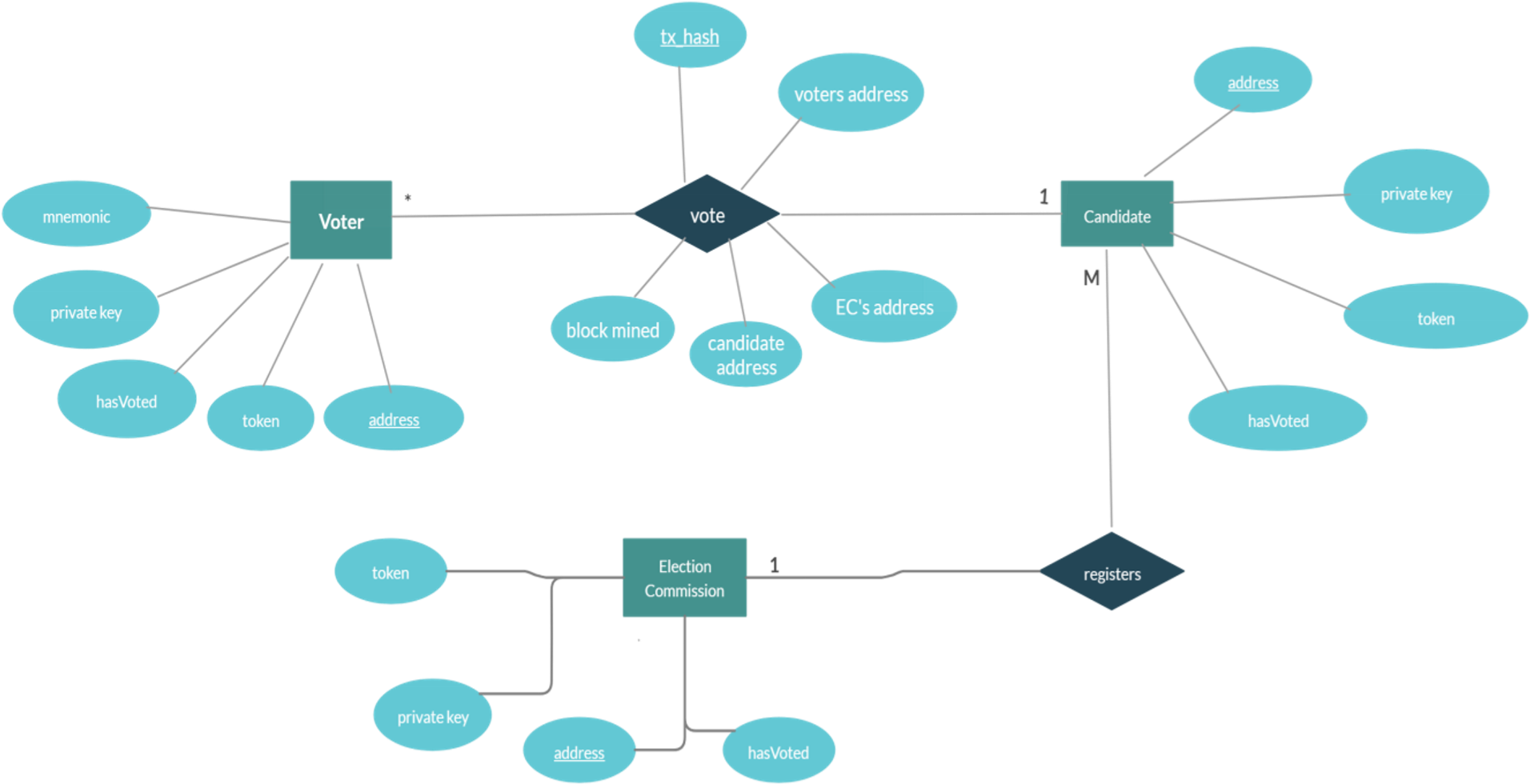


Figure 5: ER Diagram

* + 1. **SEQUENCE DIAGRAM**

Sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

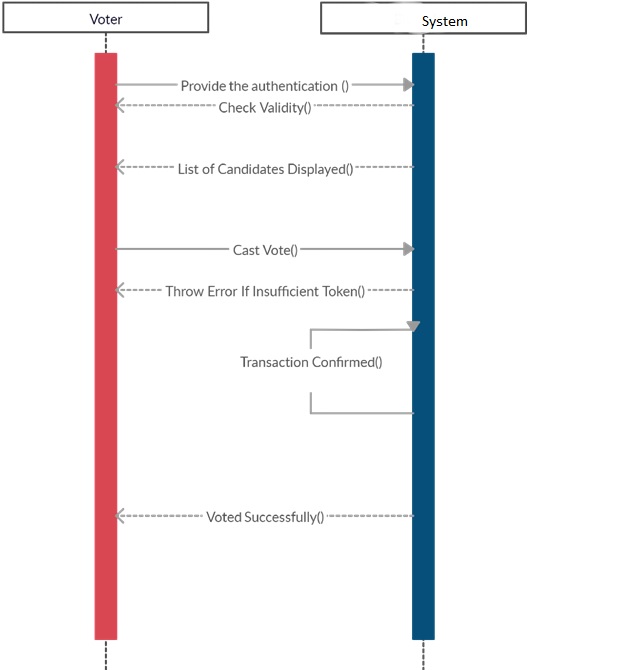
****

Figure 6: Sequence Diagram

* + 1. **ACTIVITY DIAGRAM**

Activity Diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

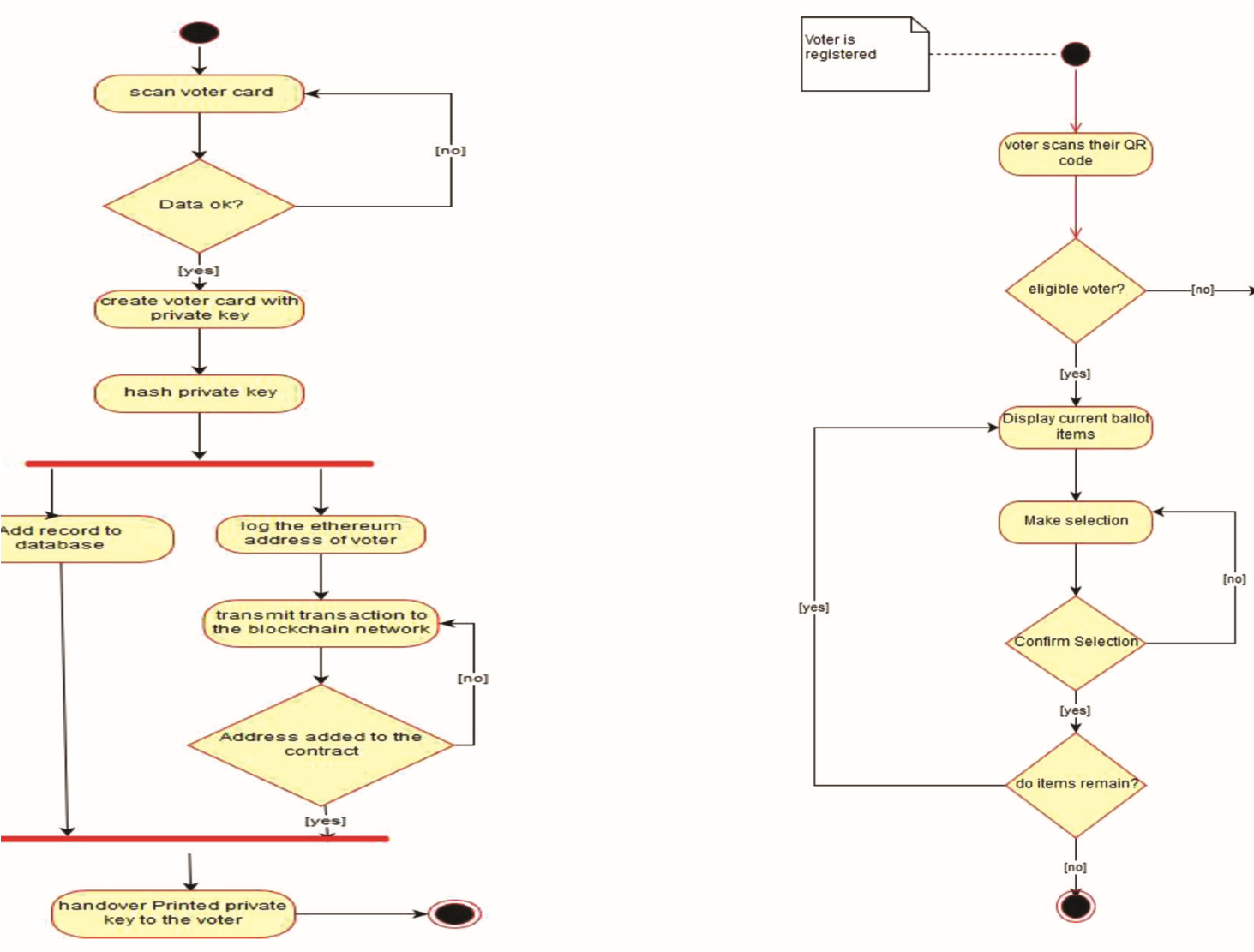


Figure 7: Activity Diagram for Voting process

* + 1. **STATECHART DIAGRAM**

Statechart diagrams define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems.

It describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

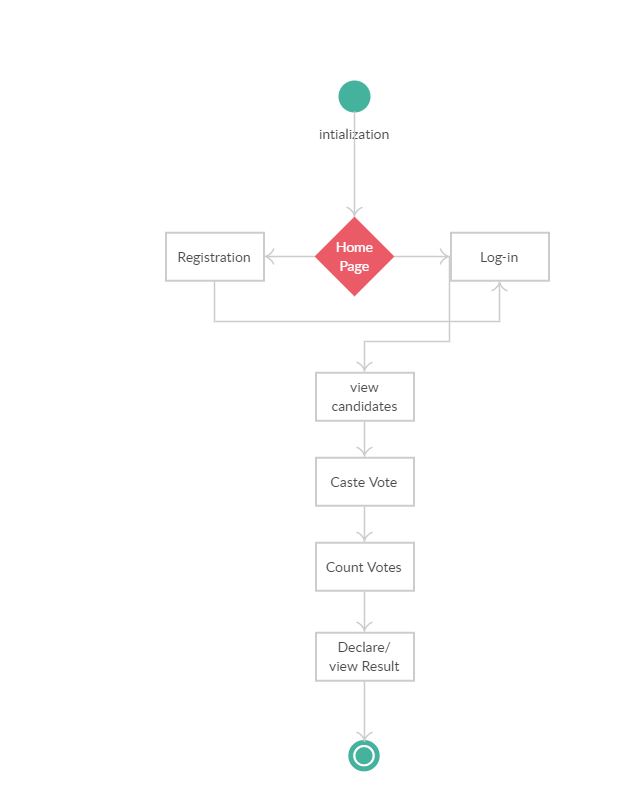


Figure 8: Statechart Diagram

* + 1. **COLLABORATION DIAGRAM**

Collaboration Diagram shows the object organization as seen in the following diagram. In the collaboration diagram, the method call sequence is indicated by some numbering technique. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.

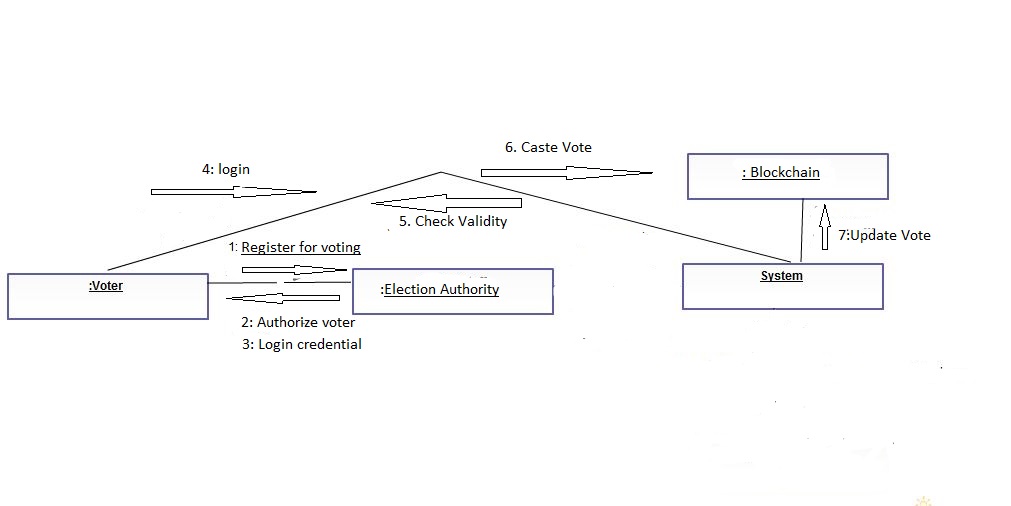


Figure 9: Collaboration Diagram

**Chapter 6**

**METHODOLOGY**

**6.1 Blockchain**

Blockchain is a distributed, decentralized, public ledger .Blockchain is of three different types, that is, public, private and the consortium blockchain. Ethereum and Bitcoin are public blockchain .This is proofed by the complexity mathematical operations. This study uses a social blockchain (Ethereum). Blockchain basically contains a series of blocks, where these blocks are linked with the chain called blockchain. The block consist of the head and body, the body of the block contains network-enabled activity. The block header contains detailed block information i.e. previous hash, nonce value and difficulty, block timestamp and transactions. Each block also contains information about person who participates in a transaction. Block length may vary, and is estimated to be about 1 and 8 MB in size.The block header specifies the block that should be put.

**Blockchain functionality**: Blockchain is the actual system built around a transparent peer-to-peer system among users to generate a transaction record namely unrepentant. To generate a block, a transaction requires to happen, after which the validity of the transaction needs to be confirmed. Actions will be stored in the block as well. The hash value should be assigned to the marking block. Therefore the block is being built and closed.

**6.2 Ethereum**

Ethereum is an open-source blockchain technology-based software platform that enables developers to build and deploy decentralized applications. A block chain is a network of separate computers, which is basic at the level, and all controls the ledger in consensus with each other .One block added to chain at a time, each block contains mathematical evidence confirming that it is added to chain and internal components are protected by strong cryptography.

Ethereum empowers developers to build and deploy distributed applications. A Decentralized application or 'Dapp' allows some specific functionality to the user, for example Bitcoin, is a Dapp that offers its users a peer to peer payment system. Because decentralized applications are coded operating on a blockchain network, not controlled by anyone or medium business. Launching these Dapps in a shared platform, blockchain, means that they benefit from all of its properties

* Immutable, no one can manipulate the data
* No single point of failure due to its decentralized property
* Secured using cryptography technique
* Zero downtime, Dapps never go down and can never be switched off.

**6.3 Ethereum Blockchain**

Ethereum Blockchain is not the only effective distributed peer-to-peer ledger exists, but has a few years of experimentally proven method.It benefits distributed consensus and this is done through a ‘proof of work’ process. This is how new information can be added to the blockchain, by nodes in a network that uses a different 'mining' alternative to the Ethereum software namely uses multiple computer resources to win the right to add another block to Blockchain, which comes with a winning user reward. The Blocks are grouped together which makes it difficult to convert transactions placed on any single block without changing all of the following blocks; as a result, the cost of converting the expansion of each block with a new block added to a file blockchain, enhances the effect of proof of work. That is why, although the transaction is considered explicit in its blockchain to Blockchain, advanced methods mean that the user is looking at a guaranteed transaction after its inclusion in the block and the addition of the following five blocks to the Blockchain

The difficulty of proof of work mining needs to be controlled so that mining time is maintained at an average of 12 seconds per block. This time Is somewhat arbitrary, but tries to find a balance between accepting Speed ​​up transactions and reduce network instability and waste, e.g., While the new block is being distributed, other miners may be working on obsolete one problem. As more miners join the network, there will be a block creation rate increase due to greater collective power. Therefore, each 2016 block re-calculates the difficulty of the math challenge Average mining time returns to normal.

**6.4 Mining and Ether**

Ether is the fuel of the Ethereum system. It is the currency of Ethereum network that received the calculated payment as Ethereum. All blockchain technologies use an incentive-based security model. The transaction is based on the "proof of work" criterion given by consensus.

Ethereum is known as the blockhain that runs certain environments as Ethereum Virtual Machine (EVM). In each node of the participant network runs EVMs and is called a proof-of-work algorithm called Ethash. Ethash is the result of finding the nonce input in an algorithm is below a certain threshold (depending on the difficulty). Nothing could be further from the truth i.e Strategy to find such nonce instead of computing potentials during validation of solution is trivial and cheap. If the output has a uniform distribution, then We can guarantee that the average time required to find nonce will depend on difficulty threshold, it is possible to control the time to find new block by changing the difficulty.

This process is how transactions are verified, new transactions are placed around the network and placed in the pool of unconfirmed transactions. These are not yet 'approved', but available for viewing by almost everyone. Miners are drawn from this pool to create candidates the transaction must be formally accepted, which will be the next block. The text of this candidate transactions along with the previous hash and nonce are input to the hash function (Ethash) and Minor Try different values ​​for nonce until the resulting hash is less than the specified one value. Since it is a cryptographic hash, there is no way to find nonce. In addition to trying to outsource guess while satisfying the output hash situation, all the miners are competing to find the first hash, each one is different set of transactions to be verified. Once minor is successful they announce their solution to the rest of the network, which becomes their block. The next block in the blockchain, and the transaction within it is confirmed. This strategy means that the minor selects the next set of verified transactions, but the hash function effectively randomizes the minor. All

Other miners verify this new block and the transactions made within and you can accept this and start working on the next block. Like the new block contains the hash of the previous block, which creates a chain of verified blocks preserving the order of transactions conducted within.

**6.5 Transaction Costs and Gas**

Ethereum has a small transaction fee, such as Bitcoin, where users has to pay small amount of gas fee (Ether) at every stage of the transaction. The sender must pay a one-time fee for each program used, including memory, storage and calculation. The size of the fee is equal to transaction difficulties, i.e. the more complex the code you wish to run the more gas (the Ether) you have to pay. For example if "Alice" wants to send a unit of "Bob" 1 Ether, there will be a total cost of 1.00001 Ether paid by Alice. However if A wants to enter into a contract or do contract work, there will be more lines of valid code, so more power consumption takes place on a distributed Ether network and will have to pay more than 1 Gas made on the first transaction. Some technical computation steps are more expensive than others, either because they are more expensive for the computer or because they increase the amount of data to be stored.

**6.6 Smart Contracts**

Smart contracts are the key element of Ethereum. In them, any algorithm must be encoded, they can implement arbitary state and perform any arbitary computations even being to call other smart contracts. This provides scripting capacity of Ethereum huge flexibility. When you make a smart contract it becomes like an automated computer program that execute automatically when certain conditions are met and because it works in the blockchain. They run as planned without restriction, downtime and any third party fraud or interference. While all blockchains have the ability to process code, most are very limited. Ethereum is different from this as it is instead of offering a limited set of functions, Ethereum allows developers to create any functionality they want to allow developers to build thousands of variety of applications that far surpasses anything ever seen before

An important part of how smart contracts work in Ethereum is that they have them their unique address on the blockchain. In other words, the contract code may not carried out inside each function you take. Instead contracts may “delayed” on a blockchain at a special price that gives an address to agreement. This function can also be coded at the time of creation. After this first transaction, the contract becomes permanent part of the blockchain as well as its address does not change. Whenever a node wants to call any of these methods defined by contract, can send a message to the contract address, specifies data as input and method to be called. The contract will do then work as part of the construction of new blocks (depending on the gas limit or completion) it may return the total value or store data.

**Chapter 7**

**IMPLEMENTATION**

**7.1 Tires**

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

**7.1.1 Client layer**

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

**7.1.2 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.

**7.1.3 Resource Manager**

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

**7.1.4 Application Logic**

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

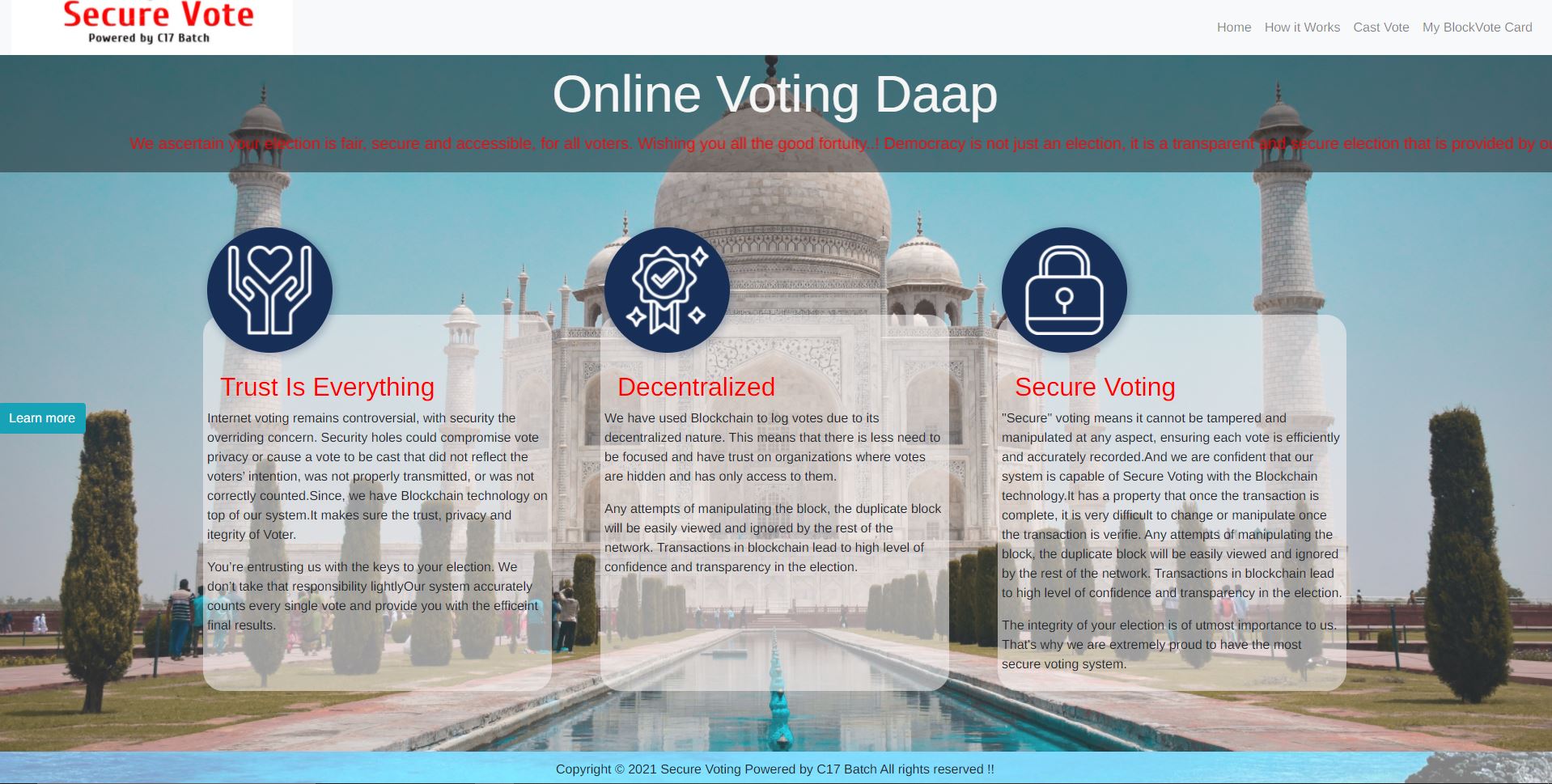
**6.2 User Interface Design** ****

Figure 10: Homepage

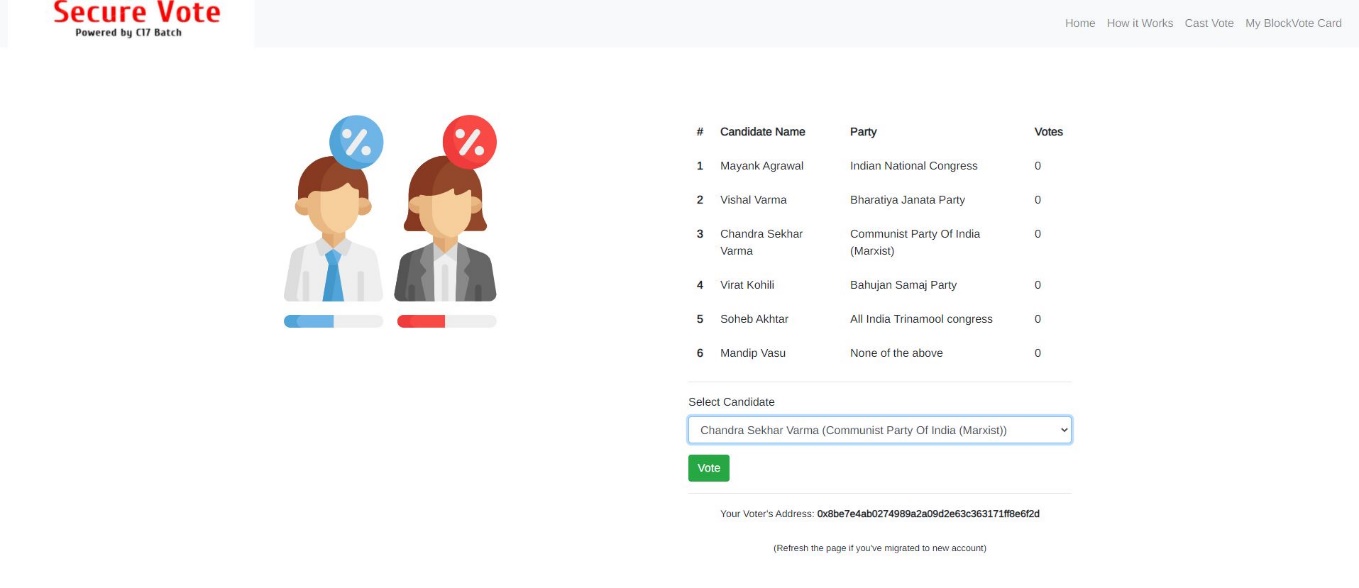


Figure 11: Casting the vote

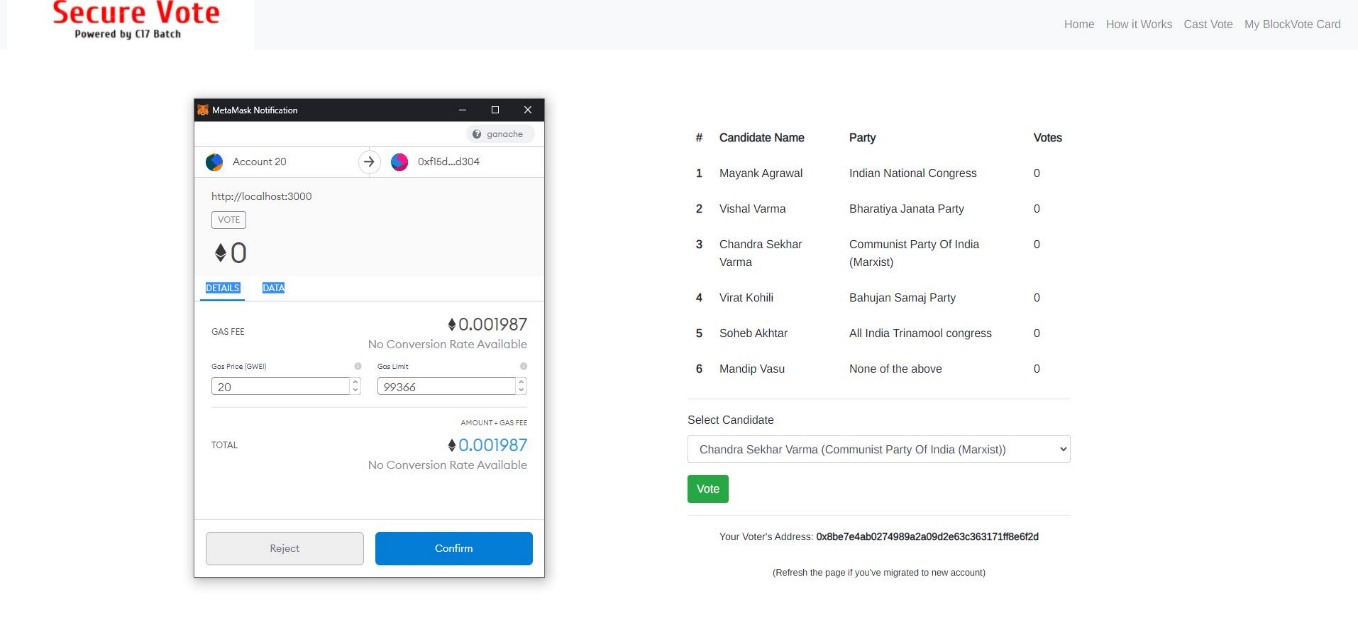
****

Figure 12: Confirming the transaction to cast vote

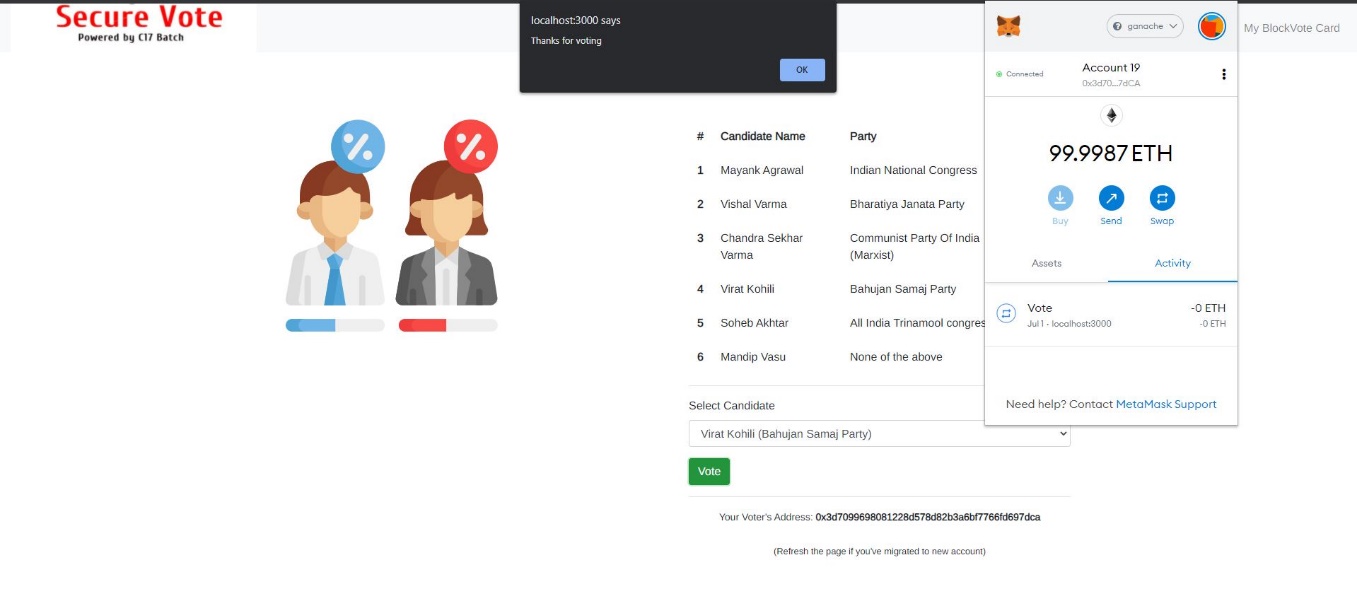


Figure 13: Transaction confirmed by miner

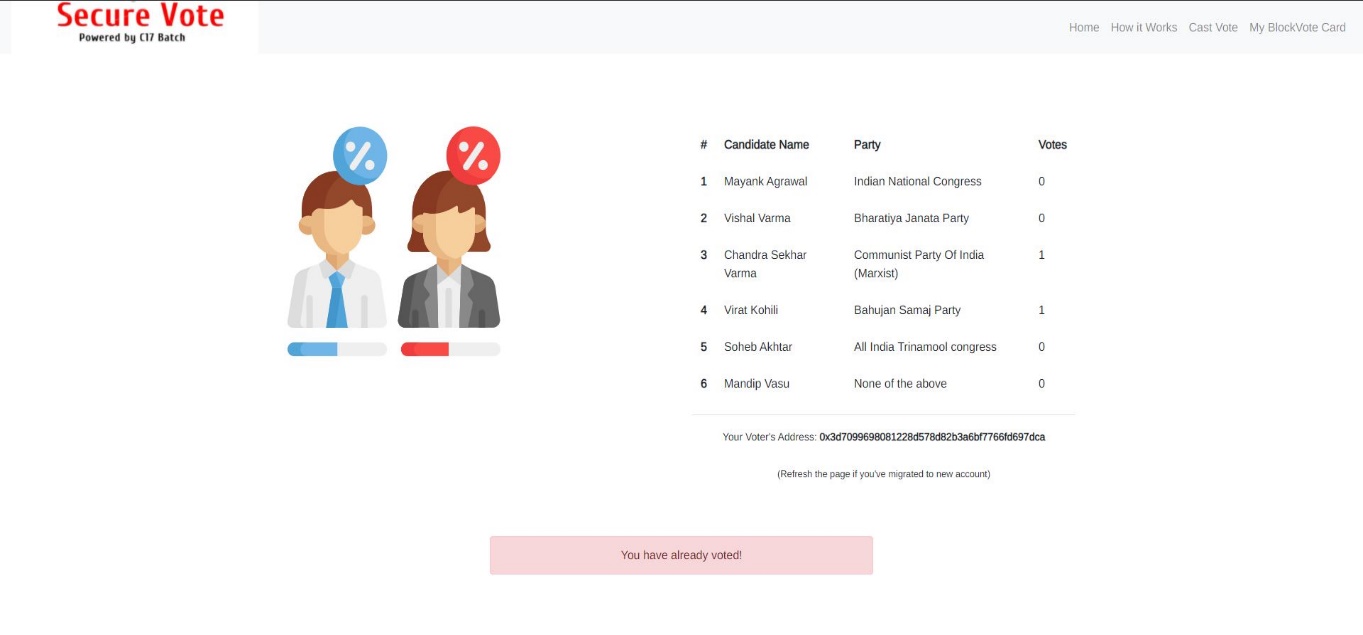


Figure 14: Already Voted Prompt

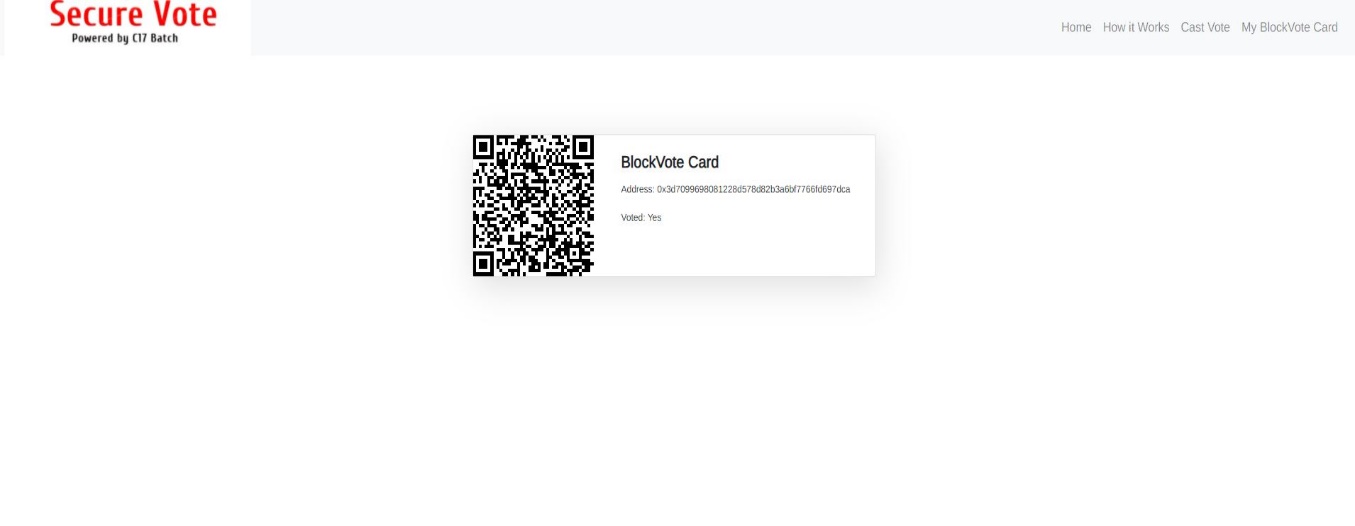


Figure 15 Customized Blockvote Card

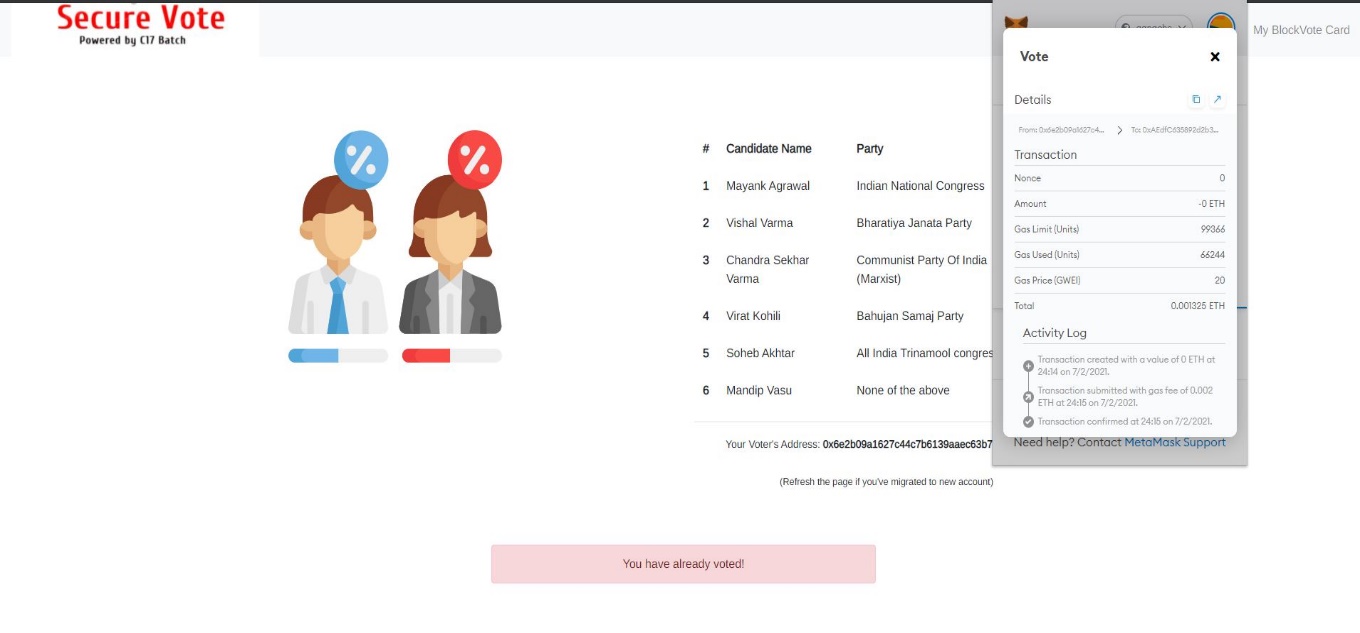


Figure 16: Transaction Confirmed Log

**Chapter 8**

**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:

**8.1 Unit Testing**

This is the first and important level of testing. It's need starts from the moment of the development of code. Every unit is tested for various scenarios. Detecting and fixing bugs earlier will help to reduce cost and time later on. It is most important to find and remove the bugs during early stage of application building process. Hence, Unit Testing is the most important of the testing levels. As the development process goes on it becomes more costly and time consuming to fix the bugs.

Steps for Unit Testing are:-

Step 1: Creation of a Test Plan.

Step 2: Creation of Test Cases and Test Data.

Step 3: Development of scripts to conduct test cases where applicable.

Step 4: Execution of the test cases, whenever the code is read.

Step 5: Fixing of the bugs if present and re-testing the code.

Step 6: Repetition of the cycle until the unit is free from the bugs.

**8.2 Integration Testing:**

Integration strategy stands for how individual modules will be combined during Integration testing. The individual modules can be combined once or they can be combined one by one. A process on how the individual module integrated together for testing is Integration Strategy. We have used bottom-up integration approach to integrate test our application. We move from bottom to top to integrate i.e. the components below are first taken and integrated first. The integration happens from bottom to top. If the calling component is yet to be developed, it is replaced by written component called a Driver.

**8.3 Testing Designs**

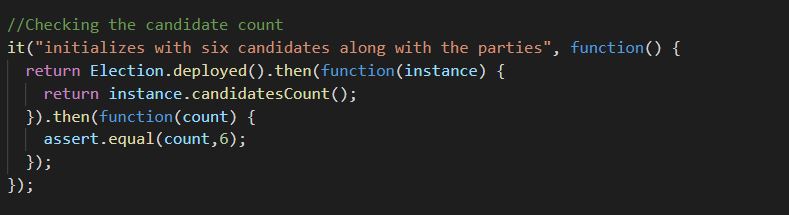
****

Figure 17: Candidate count Unit Test

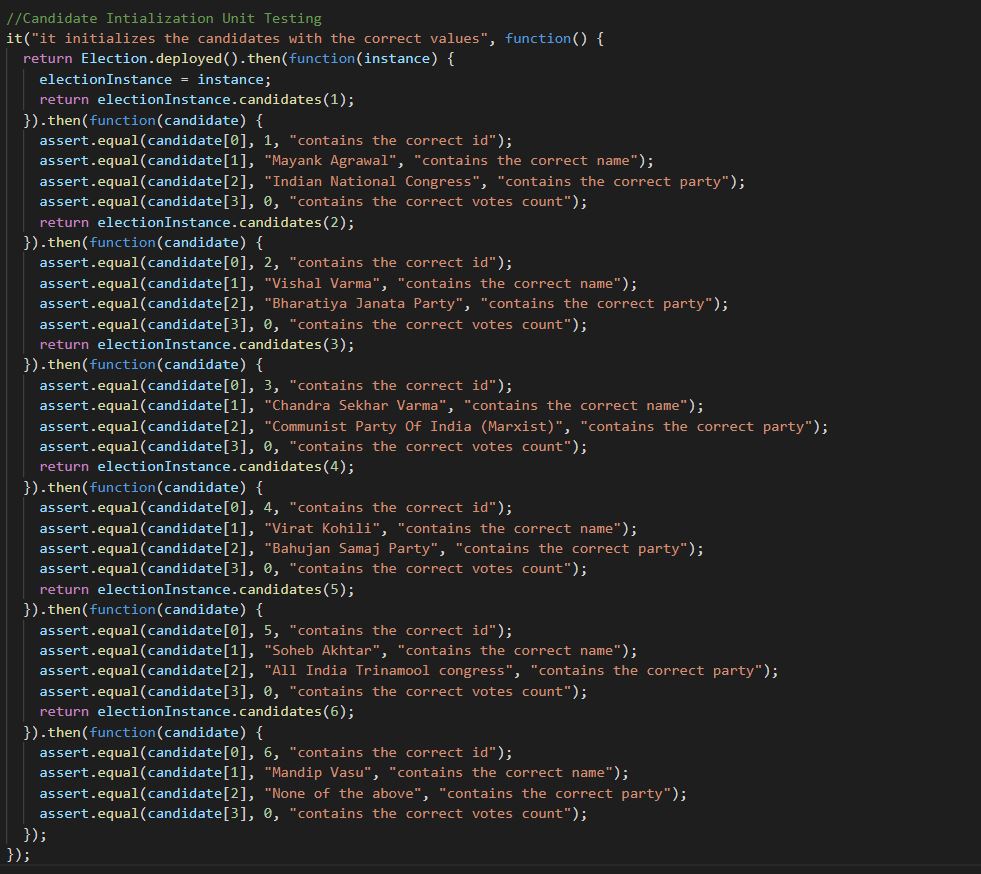


Figure 18: Candidate Initialization Unit Test

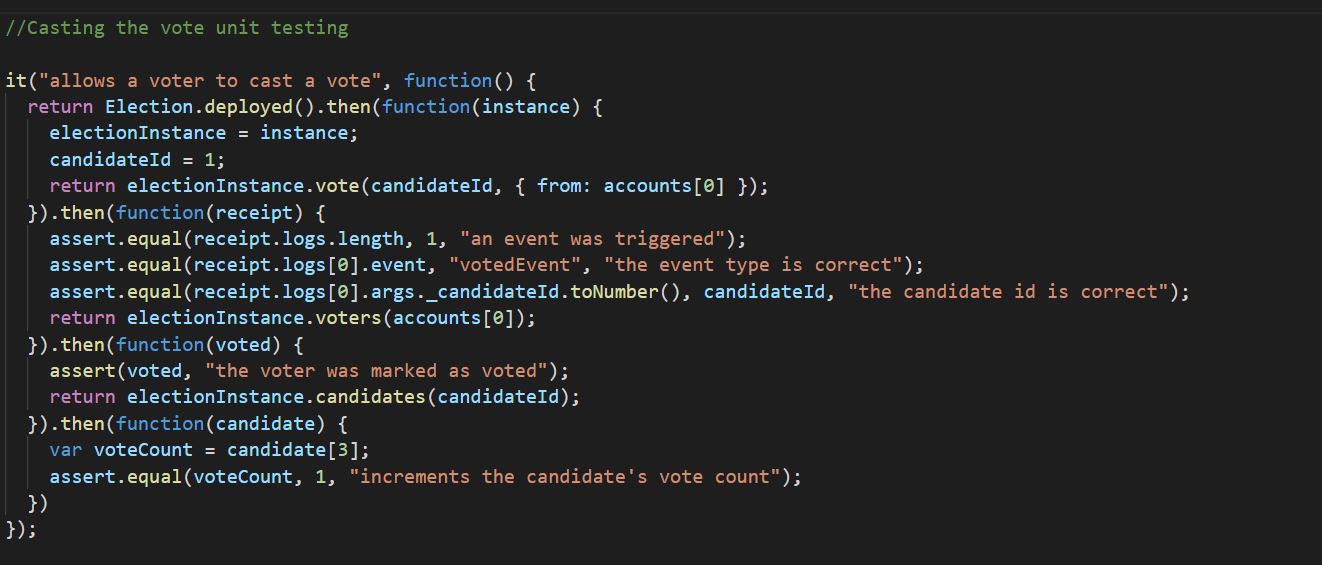


Figure 19: Casting the vote Unit Test

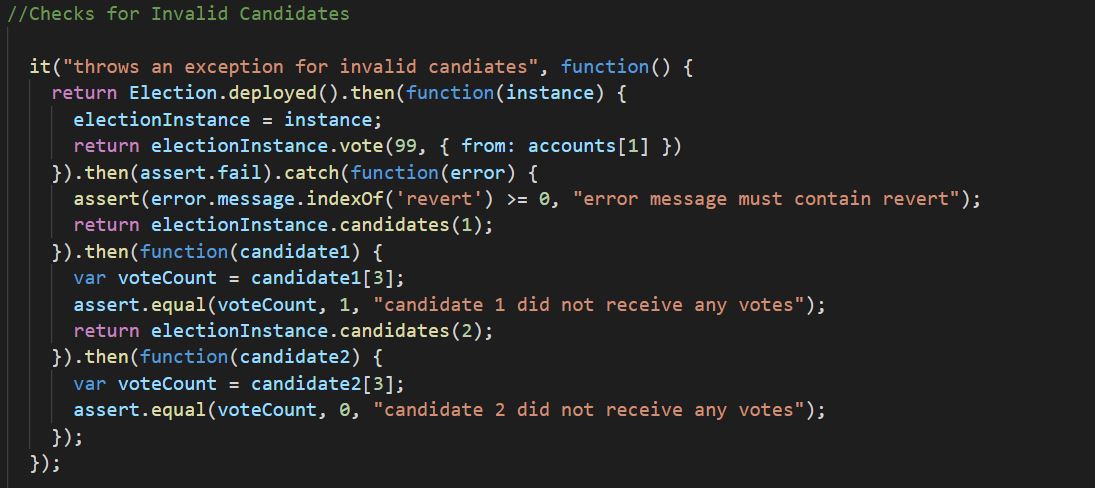


Figure 20: Invalid Candidate Unit Test

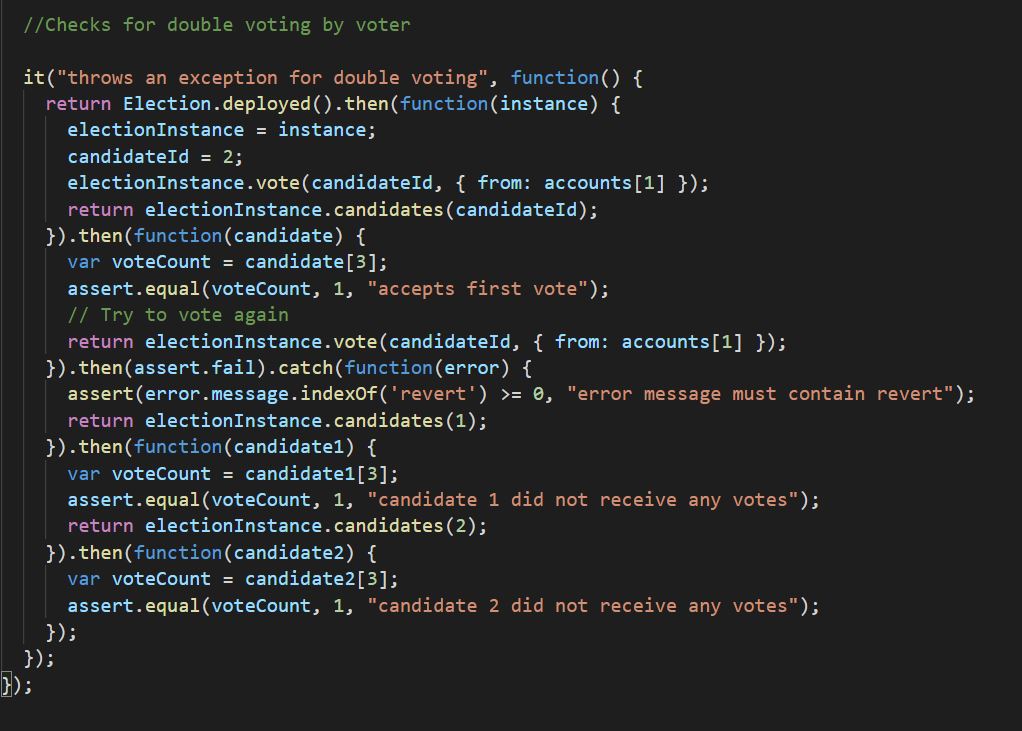


Figure 21: Double vote Unit Test

**8.4 Test Result**

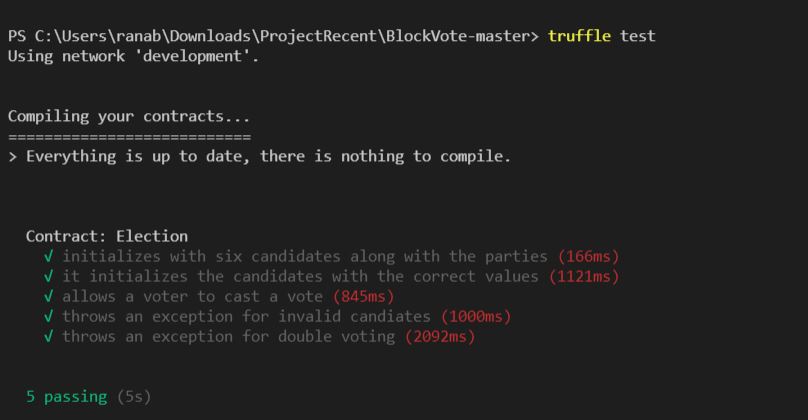
****

Figure 22: Test Result

**Chapter 9**

**CONCLUSION AND FUTURE WORK**

The idea of adapting digital voting systems to make the public electoral process cheaper, faster and easier, is a compelling one in modern society. This project has been developed to a blockchain-based electronic voting system that utilizes smart contracts to enable secure and cost-efficient election while guaranteeing voters privacy. It outlines the systems architecture, the design, and a security analysis of the system.

In the next build of this application, it has been proposed to create separate client designs for various roles such as one for election commission and one for candidates registered to a certain party with the existing voting client design. Also, the current versions lack authentication as we don’t have access to current Aadhar’s or Voter SDK to integrate in our application. Also, it is planned that in the next build notification prompt will be given on the day of voting to all the voters to cast their vote so that the voter turnout is maximum for that election.

**REFERENCES**

1. Wolchok, Scott, et al. "Security analysis of India's electronic voting machines." Proceedings of the 17th ACM conference on Computer and communications security. ACM, 2010.
2. Ohlin, Jens David. "Did Russian cyber interference in the 2016 election violate international law." Tex. L. Rev. 95 (2016): 1579.
3. Ayed, Ahmed Ben. "A conceptual secure blockchain-based electronic voting system." International Journal of Network Security & Its Applications 9.3 (2017): 01-09.
4. Hanifatunnisa, Rifa, and Budi Rahardjo. "Blockchain based e-voting recording system design." 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA). IEEE, 2017.
5. Yu, Bin, et al. "Platform-independent secure blockchain-based voting system." International Conference on Information Security. Springer, Cham, 2018

**Appendix**

**Source Code:**

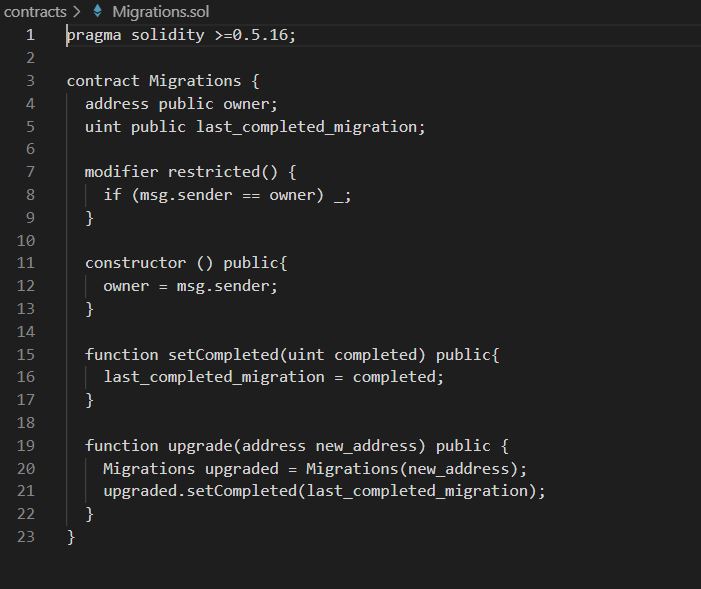
****

Figure 23: Migration Smart Contract

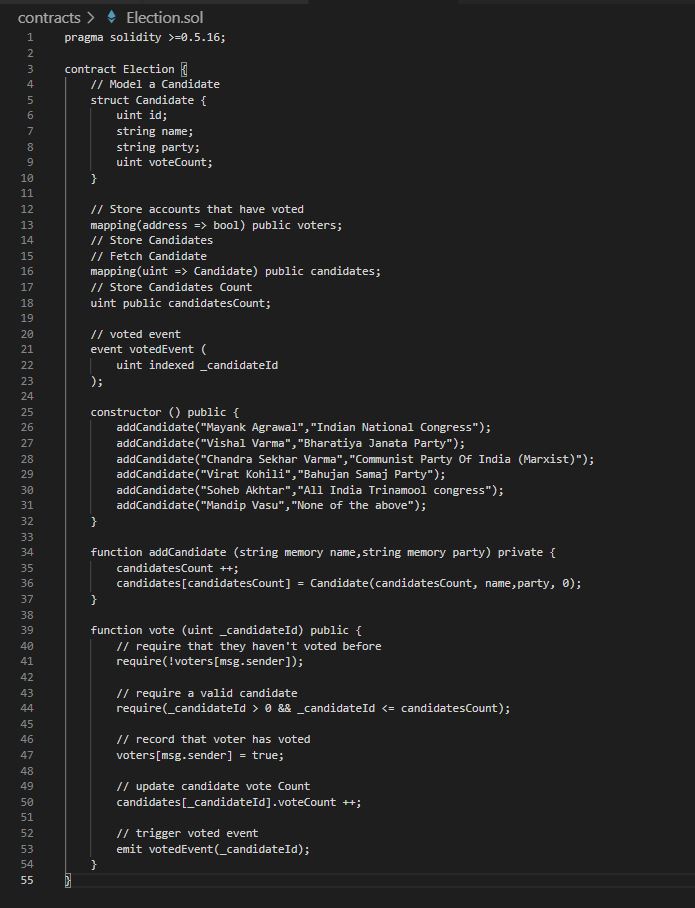


Figure 24: Election Smart Contract

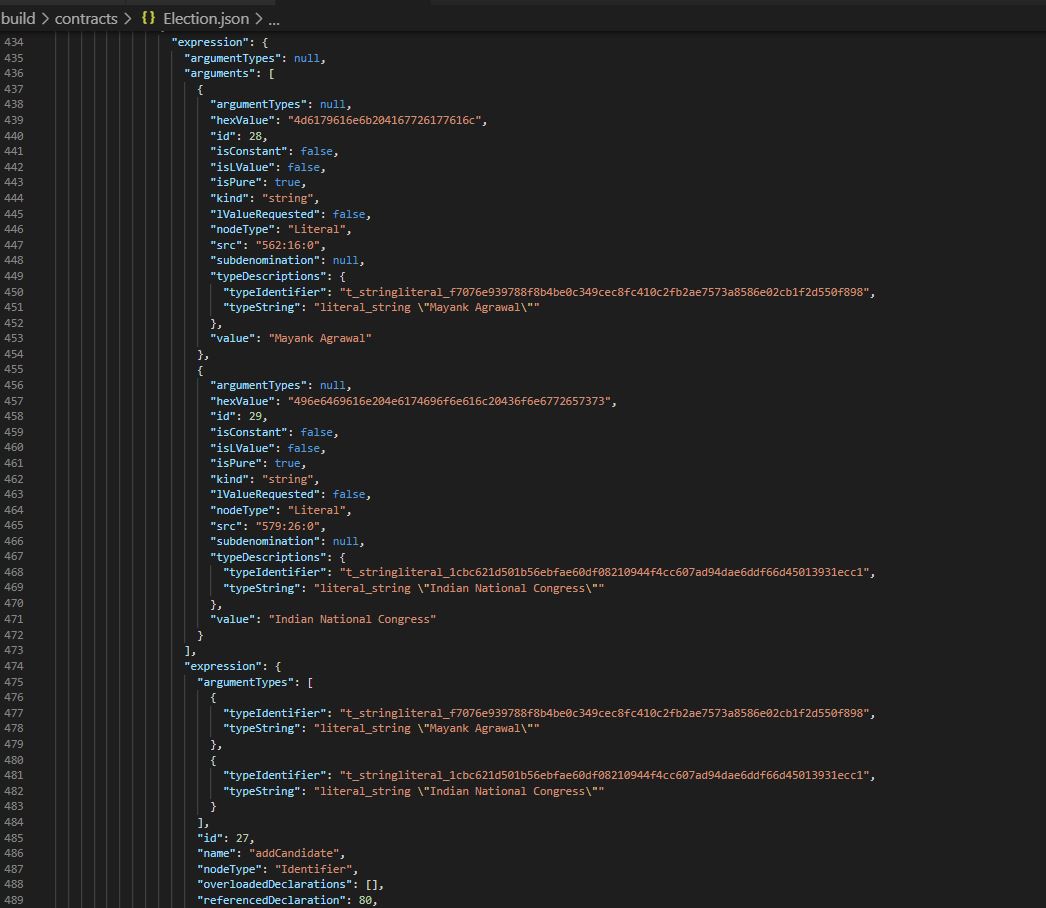


Figure 25: Election API



Figure 26: Initializing the web3 connection on Front-End



Figure 27:Initializing the Smart Contract

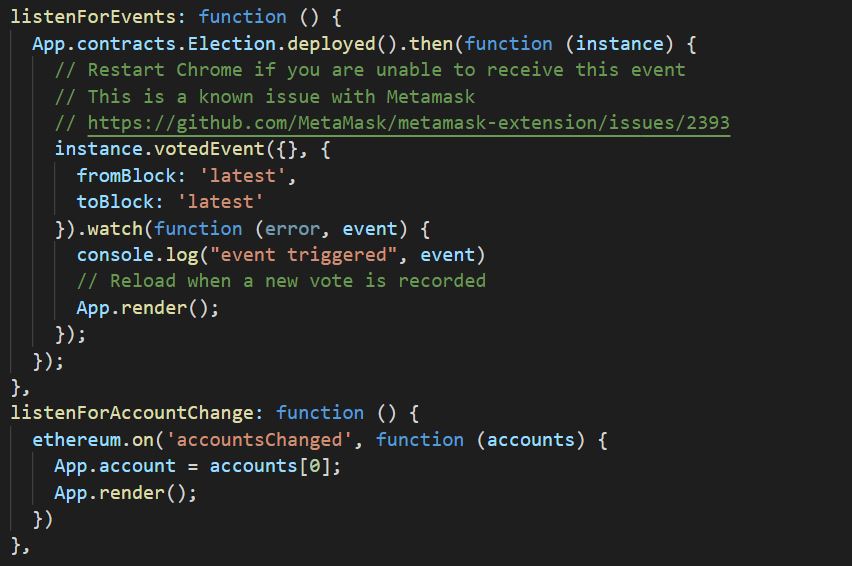


Figure 28: Trigger Voted Events



Figure 29 : Front End Integration for the Election



Figure 30: Cast Vote Function to Vote