E-VOTING BASED ON BLOCKCHAIN

A PROJECT REPORT

Submitted by

Prateek Sharma (20BCS9383)

Anant Dev Pandey (20BCS9366)

Aman Singh Negi (20BCS9380)

Kunwar Vabhav Mishra (20BCS9361)

Rohit Garg (20BCS9368)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh university

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report **“ E-Voting based on Block-Chain ”** is the bonafide work of “**Prateek Sharma , Aman Singh Negi , Anant Dev Pandey, Rohit Garg and Kunwar Vabhav Mishra”** who carried out the project work under my/our supervision.

SIGNATURE	**SIGNATURE**
Abha Gupta	Dr. Navpreet Kaur Walia
SUPERVISOR	**HEAD OF THE DEPARTMENT**

Submitted for the project viva-voce examination held on____________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our respected and learned guides, **Abha Gupta** for their valuable help and guidance, we are thankful to them for the encouragement they have given us in completing the project.

We are also grateful to respect our HOD **Dr. Navpreet Kaur Walia** and all other teachers for providing us assistance in various stages during our project.

Signature of all the group members

1. Prateek Sharma
2. Aman Singh Negi
3. Kunwar Vabhav Mishra
4. Anant Dev Pandey
5. Rohit Garg

CONTENT

List of Figures

List of Tables

ABSTRACT

Voting is the 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. While research on the topic is still emerging, it has mostly focused on the technical and legal issues instead of taking advantage of this novel concept and creating advanced digital services. Blockchain-enabled e- voting (BEV) could reduce voter fraud and increase voter access. Eligible voters cast a ballot anonymously using a computer or smartphone. BEV uses an encrypted key and tamper-proof personal IDs. Electronic credibility services have become an integral part of the information space. With the reliable implementation of basic services as an electronic signature and electronic authentication, it is possible to build more complex systems that rely on them, particularly the electronic voting system.

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 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.

CHAPTER 1. INTRODUCTION

1.1 Overview

Modern democracies are built upon traditional ballot or electronic voting (e- voting). In these recent years, devices which is known as EVMs are hugely criticized due to irregular reports of the election results. There have been many questions regarding the design and internal architecture of these devices and how it might be susceptible to attacks. This [1] paper has analyzed different techniques of tampering the EVMs. Online-voting is pushed as a potential solution to attract the young citizens and the non-resident of the country. For a robust online election scheme, a number of functional and security requirements are to be met such as transparency, accuracy, auditability, data privacy, etc.

We have worked the following ideas by having the two different set of modules: election commission and the voter(s). Election Commission creates elections and adds registered candidates along with the parties for contesting the election. Using an election's REST API hosted on Ethereum's Blockchain, the details are shown at the front-end of the voter for casting the vote. Then, while polling the vote is stored on our blockchain framework of which the Election Commission fetches the vote count. The limitation which we have faced due to not using the traditional way of smart contracts is that the blockchain framework which we have coded cannot 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 voter ID creates a drawback of not having authentication of a voter.

The most important factor of this application is to integrate the blockchain framework with both the modules for seamless voting.

1.2 Objectives

The objectives for developing the project are as follows:

- To improve the existing online voting system using Blockchain technology.

- To reduce the workload of setting up an election booth and conducting elections in physical form.
- Non-Resident Indian can cast their votes as it is totally online.
- We are supposed to learn the concept of Blockchain and how it can be utilized to work on different sectors

1.3 Literature Review

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 Syed, 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].

CHAPTER 2. SYSTEM DESIGN

2.1 Identification of Need

Identification of need is a process of determining what and how an end-user would expect a product to perform after the deployment at production level. There’s also non- technical needs of an end-user or a business client which reflects the users’ perception of the product and not the actual technical workaround, but they are closely related to the technical need at times. By implementing a needs identification system, the organization helps to ensure the proper allocation of assets to different project within the organization.

Identifying Problems

Identifying potential problems before the start of a project can save the organization significant amounts of time and money. Problem analysis is one of the most critical stages of project planning because this stage helps to guide all subsequent analysis and decision-making. If the project does not advance past this stage with solutions that the organization can implement, the project should not go forward in its current form.

Observations

The needs for a project are identified after the organization makes observations about the project. Observations are often subjective and therefore someone with expertise about the proposed project should help to make observations. A good observer can identify the needs of the project by answering key questions about the project. If the observations take into consideration the project itself and the outcome of the project, the observations should meet all of the needs of the project.

Gathering Information

Observation and gathering information represent two processes. Observations highlight what is needed. On the other hand, gathering information highlights the processes needed to execute the proposed project. Both observations and the actual gathering of information should include comments from the group that ultimately will benefit from the completed project.

Objectives and Opportunities

Once the organization has analyzed the needs and identified the objectives, the organization needs to allocate funds to capitalize the project. By successfully identifying the needs, an organization can begin to allocate resources to pay for the project. Additionally, a business needs to consider the potential future cash flow of the project. This allows the business to analyze potential cost savings to minimize costs and maximize the efficiency of the project.

2.1.1 Existing System

In India, before 2004 there was a paper-based voting system. This is called as ballot Paper system. Voters had to go to polling booth and cast their vote by marking on seal in front of the symbol of a candidate for which they wanted to cast their votes on ballot paper. Results were announced by counting the votes. The maximum vote gainer was declared as winner. India has population more than 120 crores the ballot paper voting is not much reliable, time consuming and very difficult to count the vote and there are also problems like replacement of ballot paper boxes with duplicate, damage of ballot paper, marking stamp seal for more than one candidate hence there is a strong need to overcome these problems. In order to overcome these problems Electronic Voting Machines Were introduced. Electronic Voting Machine (EVM's) mainly consists of two components:

1. ***Control Unit***: It stores and assembles votes, used by poll workers.
2. ***Ballot Unit***: It is placed in the election booth and is used the voters.

Both the units are connected via 5m cable and one end of the cable is permanently fixed to ballot unit. The control unit has a battery pack inside, which motorizes the system. The ballot unit has 16 candidate button and the unused buttons are covered with a plastic masking tab inside the unit. An additional ballot unit can be connected when there are more than 16 candidates. The additional ballot unit can be connected to a port on the underside of the first ballot unit. EVM's are internationally known as DRE's (Direct recording Electronic). EVM's are universally used in India since the general elections of 2004, when ballots were completely out of trend. They have been used in all the assembly polls and general elections of 2009. By using EVM's, Votes are correctly recorded and there is no problem in counting, scalability, Accuracy, fast declaration of results and robustness of system. Main Problem lies in authentication, the person who is voting may not be the legitimate person. Other problems like capturing of booth by political parties, casting of votes by underage people and fraud voting may occur. A person is provided with the voter id card as a proof of identity, issued by Indian government. Lot of problems are seen in voter id cards like name misprinting, missing of name, no clear photo on photo id card, etc.

2.1.2 Proposed System

Several studies have been done on using computer technologies to improve elections. These studies tell about the risks of adopting electronic voting system, because of the software challenges, insider threats, network vulnerabilities, and the challenges of auditing.

We've proposed to design the existing online voting system which is integrated with the Blockchain technology. The proposed system has the following advantages as compared to the existing system:

- Users' can vote from anywhere in the world until he possess a citizenship of the country.
- The voting is stored in the Blockchain which makes it tamper proof.
- As there's no standing in queue for casting vote it will save a lot of time and reduce the workload. Users' can vote from anywhere in the world until he possess a citizenship of the country.

2.2 Preliminary Investigation

The main aim of preliminary investigation is to identify the problem. First, need for the new or the enhanced system is established. Only after the recognition of need, then the proposed system is compared and then further analysis is possible. At this stage, we had to perceive the problem and opportunities, the existing system is studied and found out that there were few areas where we can integrate with other technology to make the system better than the existing system. It was analyzed that such proposed system would be possible to develop with given and it might turn out to be the feasible solution.

In this project, the biggest challenge was to integrate the existing online voting system with the designed blockchain framework and on further development levels we encountered various unit level problems such as the model for the Election Commission to create votes and store the necessary details of candidates along with the election details. On the later part of this document, we have come up the features which can be added to our software to make it better than the initial deployment.

2.3 Feasibility Study

A feasibility study is a high-level capsule version of the entire system analysis and design process. The study begins by classifying the problem definition. The purpose of feasibility study is not to solve the problem, but to determine whether the problem is worth solving. It is a preliminary study which is conducted before the real development of the project commences not keeping the factor of project’s success. It creates a roadmap of what are the possible solutions if we choose a certain path. The feasibility study concentrates on the following areas:

2.3.1 Technical Feasibility

Evaluating the technical feasibility study is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed designs of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Overall, this study needs to demonstrate that the proposed system which is need to be developed is technically feasible.

This requires:

- An outline of the requirements,
- A possible system design,
- Possible choices of software to be used or developed,
- Estimates on number of users, data, etc.

2.3.2 Economic Feasibility

The economic feasibility study evaluates the cost of the software development against the ultimate income or benefits gets from the developed system. There must be scopes for profit after the successful Completion of the project. The life cycle of an engineering project or product contains of several stages, namely: (i) Planning and design; (ii) Development; (iii) Operation and maintenance. It should be performed to identify the financial risk associated with the project.

Various techniques like net present value (NPV), payback period, return on investment (ROI) are employed. Techno-Economic Assessment (TEA) is a cost-benefit comparison using different methods. These assessments are used for tasks such as:

- Evaluate the economic feasibility of a project.
- Investigate cash flows over the lifetime of the project.
- Evaluate the likelihood of different technology scales and applications.
- Compare the economic quality of different technology application providing the same service.

2.3.3 Operational Feasibility

The operational feasibility study focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture, and existing business processes. It is also the measure of how well the solution will work in the organization after it is deployed. As we are dealing with blockchain voting system, which indirectly targets the country’s or state’s election process protocol, so there will be a detailed comparison between these two to check which one dominates the other. It is also the measure how people will feel about the project as in will people be accustomed to use this in a proper way or it will be too complex to deal with.

There are two aspects of operational feasibility to be considered:

- Is the problem worth solving?
- How do the end user (voters in this case) and management (Election Commission) feel in this case?

2.3.4 Schedule Feasibility

It means that the project can be implemented in an acceptable time frame. When assessing schedule feasibility, a systems analyst must consider the interaction between time and costs. For example, speeding up a project schedule might make a project feasible, but much more expensive.

Other issues that relate to schedule feasibility include the following:

- Can the company control the factors that affect schedule feasibility?
- Has management established a firm timetable for the project?
- What conditions must be satisfied during the development of the system?
- Will an accelerated schedule pose any risks? If so, are the risks acceptable?
- Will project management techniques be available to coordinate and control the project?
- Will a project manager be appointed?

It is also the likelihood that timeframes can be met and that this is adequate to meet organization’s needs.

2.3.5 Legal Feasibility

It determines whether the proposed system conflicts with the legal requirements, in this case as we didn’t try to execute anything on the public domain, hence this project is legal feasible.

It is important that the project is following the requirements needed to start a project including certificates, copyrights, business insurance, tax number, health and safety measures and many more. There are some things to consider in legal feasibility study including ethical issues and some social issues. These issues are the privacy and accountability. In this project, everything is designed keeping in mind all the legal terms and no real-world data or privacy has been breached of any person of this country to use it as a sample voter to implement this application.

2.4 Project Planning

Project Planning is the most essential thing in developing a project. It sets out the phases, activities and task needed to deliver a project. The timeframes required to deliver the project, along with the resources and milestones are also shown on the project plan.

Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. Project planning is often used to organize different areas of a project, including project plans, workloads and the management of teams and individuals. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path. Project planning is inherently uncertain as it must be done before the project is actually started. Therefore, the duration of the tasks is often estimated through a weighted average of optimistic, normal, and pessimistic cases. The critical chain method adds "buffers"; in the planning to anticipate potential delays in project execution. Float or slack time in the schedule can be calculated using project management software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project schedule may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the project schedule becomes what is known as the baseline schedule. Progress will be measured against the baseline schedule throughout the life of the project. Analyzing progress compared to the baseline schedule is known as earned value management.

A project plan is a model of the process that the project team intends to follow to realize the project objectives. It brings together a number of important aspects of this process including its scope, timing and associated risks. The project plan can be viewed as a type of "contract" between the project team members and the reviewers. It defines the process by which objectives will be achieved, and the responsibilities in carrying out this process. It also underpins a number of other key project management functions including estimating and forecasting, options analysis and decision-making, and performance monitoring and control.

The essential elements of a project plan are:

- Scope statement
- Schedule
- Requirements
- Quality criteria
- Project resources

- Communications Plan

Scope statement

It is a statement of what work is included within the project, and what is not. A good scope statement significantly reduces risk of project overruns and unexpected turbulence. In this project, the scope statement is as follows:

“This project is for the creation of an online election system using Blockchain technology. There will be a website for Election Commission and for the voters. The user interface will be designed as part of the project which will contain necessary details at both the end”.

Schedule

The project schedule communicates to all stakeholders what the expected arrival time will be, and serves to keep the project manager’s hands on the throttle throughout the project. Since projects are by definition temporary endeavors with a defined beginning and end, the exact location of that end date is a primary consideration for most projects.

The details of this project’s schedule will be discussed later under *Project Scheduling.*

Requirements

All projects have requirements which are drafted at the beginning as per client’s needs. In this project, the requirements are such as the module of creating elections, adding candidates contesting the elections. Detailed discussion of the requirements is discussed under ***Requirement Specifications.***

Quality Criteria

It is one of the essential elements of project planning as if a software is not inspected properly and then deployed to the market it might cause few problems which will then create pressure among the maintenance. The quality criteria should be identified in the project plan, including pass/fail requirements, as well as the methods used to ensure the quality criteria will be met.

Project Resources

Resources often require the most planning and coordination throughout the project’s execution. That’s because they arrive late, require unexpected maintenance, don’t meet specifications, or any other host of issues that can trip up a project. Resources are the technology stack which will be used in developing the software. Details about this is discussed at the later part of this documentation.

2.5 Project Scheduling

It requires us to follow some carefully laid-out steps, in order, for the schedule to take shape. It is an organized method of presenting in formation on when activities need to be started, how long activities are planned to be completed.

There are basic principles for project scheduling, such as follows:

- **Defined responsibilities**
  - Every task that is scheduled is assigned to a specific team member.
- **Defined outcomes**
  - Every task that is scheduled should have a defined outcome for software projects such as a work product.
- **Define milestones**
  - Every task or group of tasks should be associated with a project milestone.
  - A milestone is accomplished when one or more work products has been reviewed and then approved by the team leader.

PERT and GANTT Chart were developed to represent the project schedule and track the different tasks.

2.6 Software Requirement Specification

2.6.1 Introduction

This document describes the structural properties and software requirements of the Online Election System using Blockchain Technology.

2.6.1.1 Problem Definition

Manual voting system has been deployed for many years in our country. However, in many parts of our country people cannot attend the voting because of several reasons. To illustrate, sometimes people may not be in their own registration region and due to this fact, they cannot fulfill their voting duties. In order to solve these problems, there is a need of online election voting system with this keeping in mind that EVM votes tampering issues are also encountered, so this online election system will be integrated with Blockchain Technology to make it tamper proof.

2.6.1.2 Purpose

The purpose of this document is to make the functional and non-functional requirements of the Online Election System using Blockchain Technology easy to comprehend. It also serves the purpose of making the functionality clear to end users.

2.6.1.3 Scope

This SRS document applies to the initial version (release 1.0) of the "Online Election System using Blockchain Technology" software package. This document describes the modeling and the requirement analysis of the system. The main aim of the system is to provide a set of protocols that allow voters to cast votes while the election commission is responsible for creating elections and adding candidates.

2.6.1.4 Definitions and Abbreviations

The following is a list of terms, acronyms and abbreviations used by the Online Election System using Blockchain software package and related documentation.

Table 1: Definitions and Abbreviations

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

2.6.1.5 Overview

The remainder of this document identifies the actors, use-cases, use-case scenarios, activity diagrams, assumptions and dependencies needed for the analysis and design of the Online Election software package. The rest of the document contains the overall description of the system, requirements, data model and behavioral description of the system and project planning.

2.6.2 Overall Description

The Online Election System is a web-based system so fundamental features related with web-based technologies such as client-server and database properties determine the software requirements of that project along with the addition of blockchain framework.

2.6.2.1 Product Perspective

The software product is a standalone system and not a part of a larger system. The system will be made up of two parts. One will be used for general purposes by the EC, such as viewing candidates. On the election day, the voter needs to import his/her Ethereum's wallet and get authenticated accordingly. The voters cast their votes using the interface that is provided. These votes are accepted by the blockchain and then thrown into the server. The EC configures the whole system according to its needs on the server.

2.6.2.2 Product Features

1. *Eligibility*: This property states that only eligible users can vote. Those who are provided with authentication by the Election Commission.

2. *Privacy*: Privacy is one of the most important aspects of democratic voting. Voters privacy should be maintained. No one should be able to know how a particular person voted or to whom the particular voter voted.

3. *Coercion resistance*: No one should be able to force the voter and should not have the ability to distinguish between whether the voter voted the same way he/she was instructed to vote.

4. *Verifiability*: This property states that everyone involved in the voting

process should be able to verify the results. This brings transparency in the election. Also, an individual voter should be able to verify whether his/her vote is counted or not.

5. *Immutability*: The voter's vote should be immutable. No one should be able to change the vote of any voter without proper concern of the voter. All the records should be immutable.

2.6.2.3 Constraints, Assumptions and Dependencies

The system enables voters to cast their vote from anywhere and is authenticated by EC and provided with the ETH wallet address and private key. Security and anonymity are the most crucial fundamentals of this blockchain voting system.

For the proper working of the system, we can list our assumptions and dependencies as follows:

- Metamask Browser Extension: Metamask allows users to manage accounts and their keys in a variety of ways, including hardware wallets, while isolating them from the site context.
- Ganache: It is a personal blockchain for rapid Ethereum and Corda distributed application development.
- 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.

2.6.3 Functional Requirements

2.6.3.1 Software Requirements

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

2.6.4 Non-functional Requirements

2.6.4.1 Performance Requirements

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:

(i) **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.

(ii) **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.

2.6.4.2 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.

2.6.4.3 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.

2.6.4.4 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.

2.7 Software Engineering Paradigm

This project uses an iterative model approach using Agile methodologies. Let's discuss this in details. Agile methods of software development are most commonly described as iterative and incremental development. The iterative strategy is the cornerstone of Agile practices, most prominent of which are SCRUM, DSDM, and FDD. The general idea is to split the development of the software into sequences of repeated cycles (iterations). Each iteration is issued a fixed-length of time known as a timebox. A single timebox typically lasts 2-4 weeks.

The ADCT (Analysis, Design, Code, Test) wheel is more technically referred to as the PDCA (Plan, Design, Check, Adjust) cycle. The team implements the PDCA cycle on each iteration separately in the following manner:

- **P (Plan) – Iteration Planning**

In this event, the team collaborates to discuss the objectives for the next iteration. It also summarizes the work done and determines the team backlog required for the next iteration.

- **D (Design) – Iteration Execution**

This is the ‘do’ step where the development of the software, its design and coding takes place. If it’s a second or third iteration, then functionality testing is also conducted. The team collects user stories and prepares for the next step, that is the Iteration Review.

- **C (Check) – Iteration Review**

Also known as the ‘check’ step, Iteration Review is carried out with the Product Owner. The team shows the tested deliverable to the Product Owner, who then reviews the completed work and ascertains whether all criteria have been met.

- **A (Adjust) – Iteration Retrospect**

In this event, the team evaluates the entire process of the iteration from the first step. It essentially works on any improvements that are gathered in previous iterations. New problems are identified along with their causes. Before the team starts the next cycle again, team backlog is refined for future reference. The iterations are repeated for optimizations and improvisations and, the lessons learned from previous cycles are applied in the next cycle. Until a fully functional software is ready to hit the market.

Agile methodologies have the following advantages over other methods:

Customer Involvement – Agile Iterative development encourages user contribution. After each iterative cycle, customer feedback is obtained, and the product is then subjected to necessary changes based on that feedback. This aspect brings adaptability into the project’s framework.

Favors Evolution – The planning in the Agile Iterative development process is a continuous feat, that allows space for evolving ideas, instead of extensive planning that only precedes execution and testing in Waterfall.

Risk Assessment – Agile iteration allows risk identification and mitigation early on in the development to avoid speed bumps later down the timeline.

Rapid Delivery – The work is divided into small cycles, allowing team members to dedicate their focus and deliver on time. Moreover, testing is conducted simultaneously in coding and design in every iteration, which greatly reduces the time needed to achieve completion.

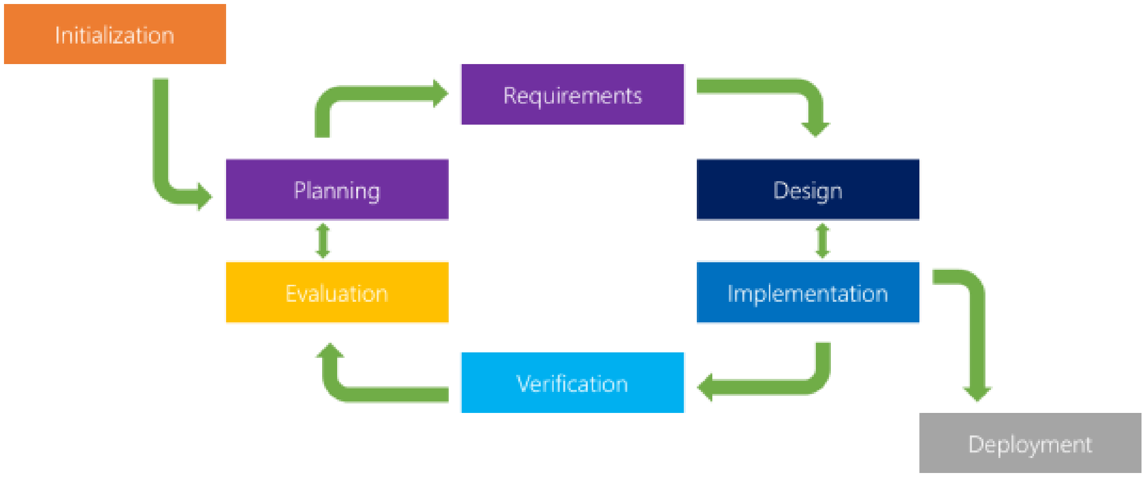


Figure 1: Iterative Model of Software Development

2.7.1 Process

- **Planning & Requirements**: As with most any development project, the first step is to go through an initial planning stage to map out the specification documents, establish software or hardware requirements, and generally prepare for the upcoming stages of the cycle.
- **Analysis & Design**: Once planning is complete, an analysis is performed to nail down the appropriate business logic, database models, and the like that will be required at this stage in the project. The design stage also occurs here, establishing any technical requirements (languages, data layers, services, etc.) that will be utilized in order to meet the needs of the analysis stage.
- **Implementation**: With the planning and analysis out of the way, the actual implementation and coding process can now begin. All planning, specification, and design docs up to this point are coded and implemented into this initial iteration of the project.
- **Testing**: Once this current build iteration has been coded and

implemented, the next step is to go through a series of testing procedures to identify and locate any potential bugs or issues that have cropped up.

- **Evaluation**: Once all prior stages have been completed, it is time for a thorough evaluation of development up to this stage. This allows the entire team, as well as clients or other outside parties, to examine where the project is at, where it needs to be, what can or should change, and so on.

2.7.2 Advantages

- **Inherent Versioning**: It is rather obvious that most software development life cycles will include some form of versioning, indicating the release stage of the software at any particular stage. However, the iterative model makes this even easier by ensuring that newer iterations are incrementally improved versions of previous iterations. Moreover, in the event that a new iteration fundamentally breaks a system in a catastrophic manner, a previous iteration can quickly and easily be implemented or "rolled back," with minimal losses; a particular boon for post-release maintenance or web applications.

- **Rapid Turnaround**: While it may seem like each stage of the iterative process isn't all that different from the stages of a more traditional model like the waterfall method — and thus the process will take a great deal of time — the beauty of the iterative process is that each stage can effectively be slimmed down into smaller and smaller time frames; whatever is necessary to suit the needs of the project or organization. While the initial run through of all stages may take some time, each subsequent iteration will be faster and faster, lending itself to that agile moniker so very well, and allowing the life cycle of each new iteration to be trimmed down to a matter of days or even hours in some cases.

2.8 Data Models and Descriptions

2.8.1 Sequence Diagram

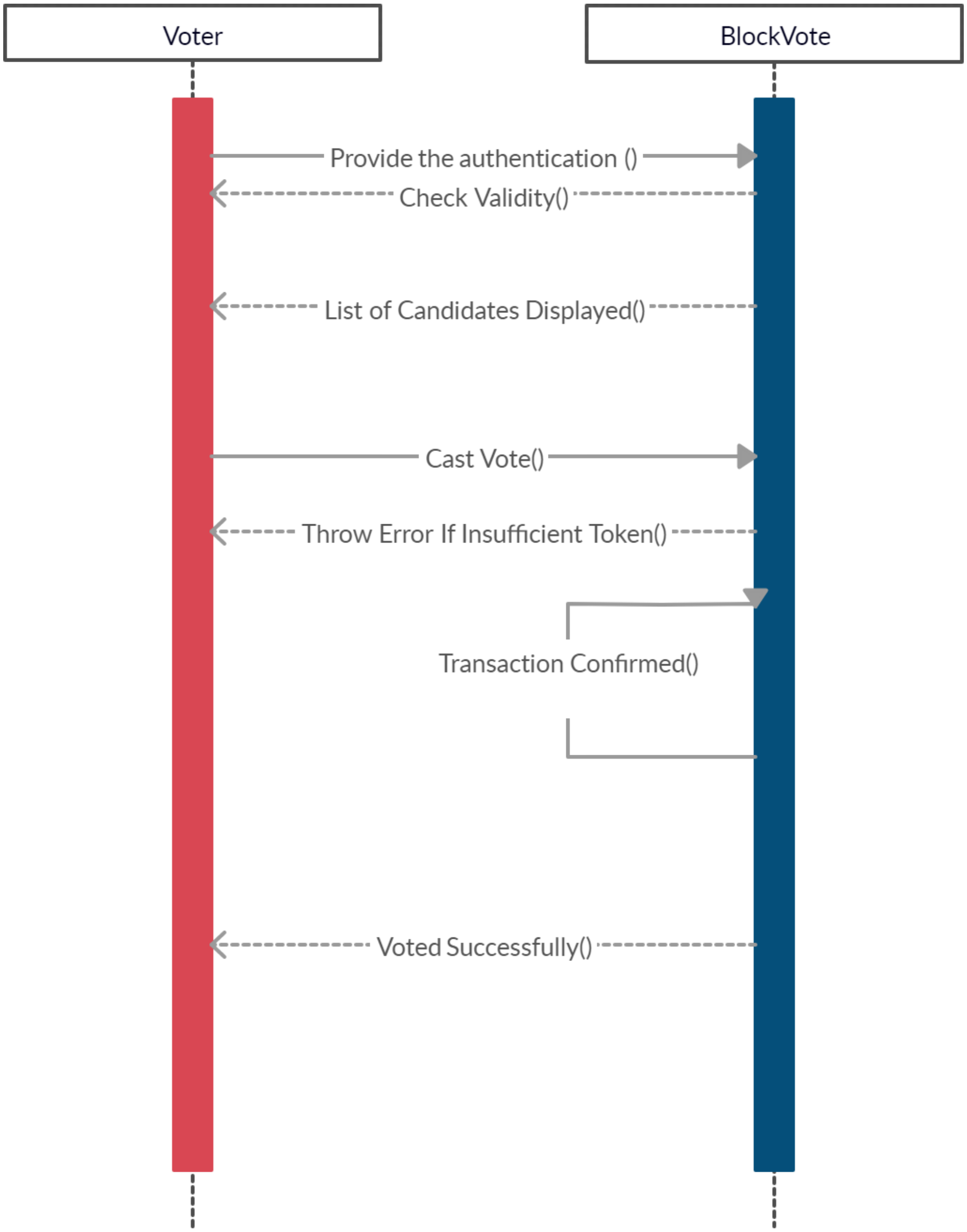

Voter
BlockVote
Provide the authentication ()
Check Validity()
List of Candidates Displayed()
Cast Vote()
Throw Error If Insufficient Token()
Transaction Confirmed()
Voted Successfully()

Figure 2: Sequence Diagram

2.8.2 Entity Relationship Diagram

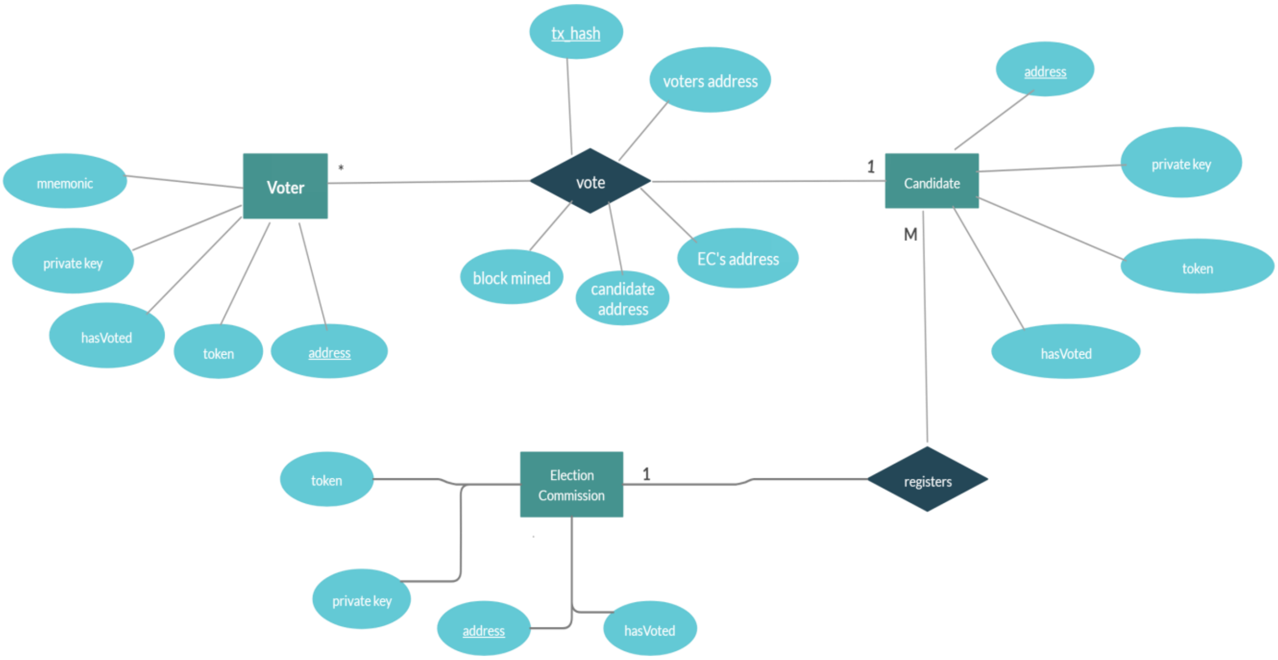


Figure 3: ER Diagram

2.8.3 Flowchart

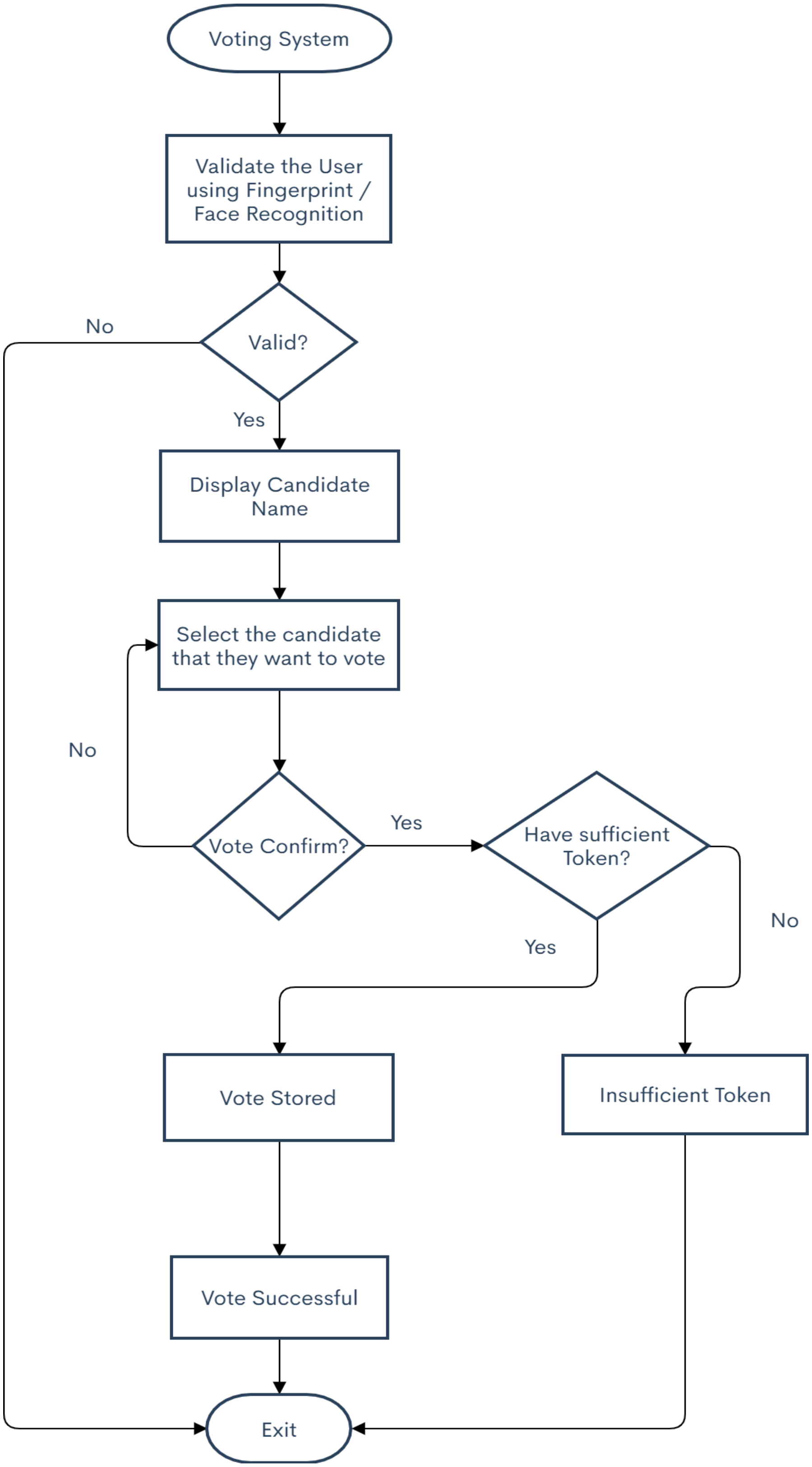


Figure 4: Flowchart

2.9 System Design

2.9.1 Design Goal

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.

2.9.2 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:

(i) **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.

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

(iii) **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.

2.9.3 Implementations

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.

2.9.4 User Interface Design

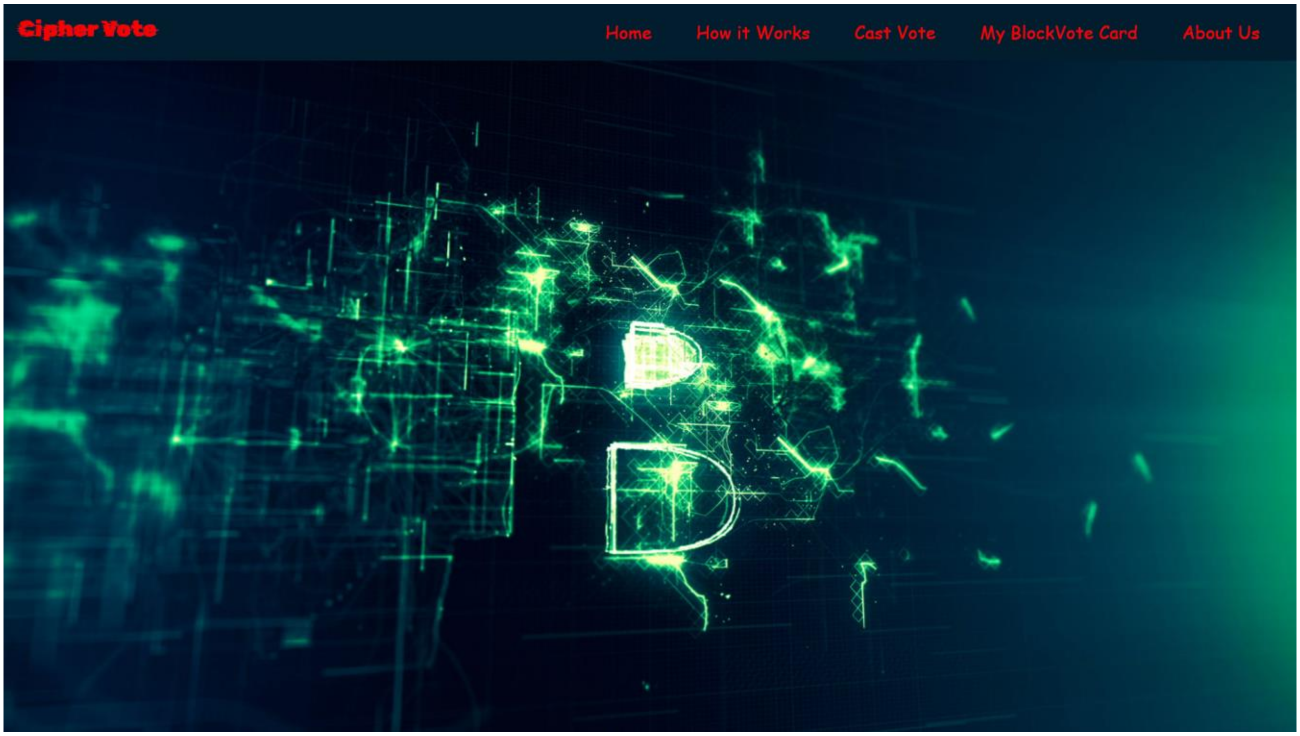
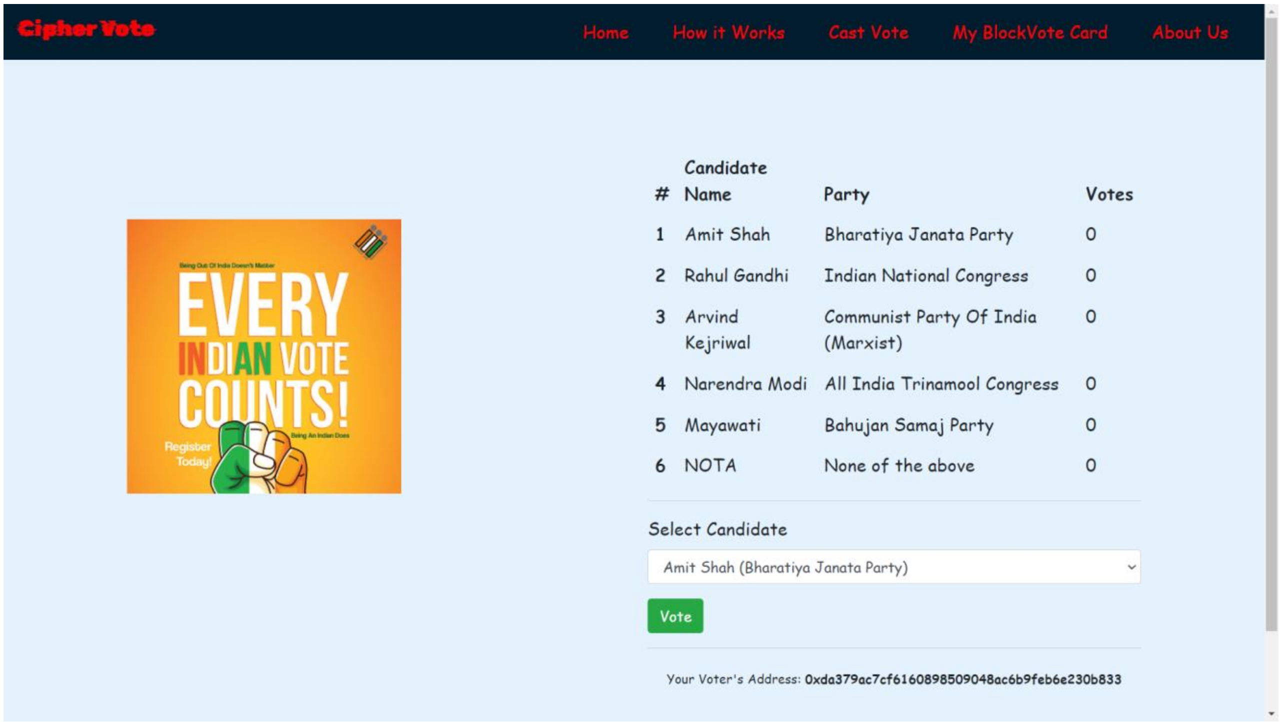


Figure 5: Homepage



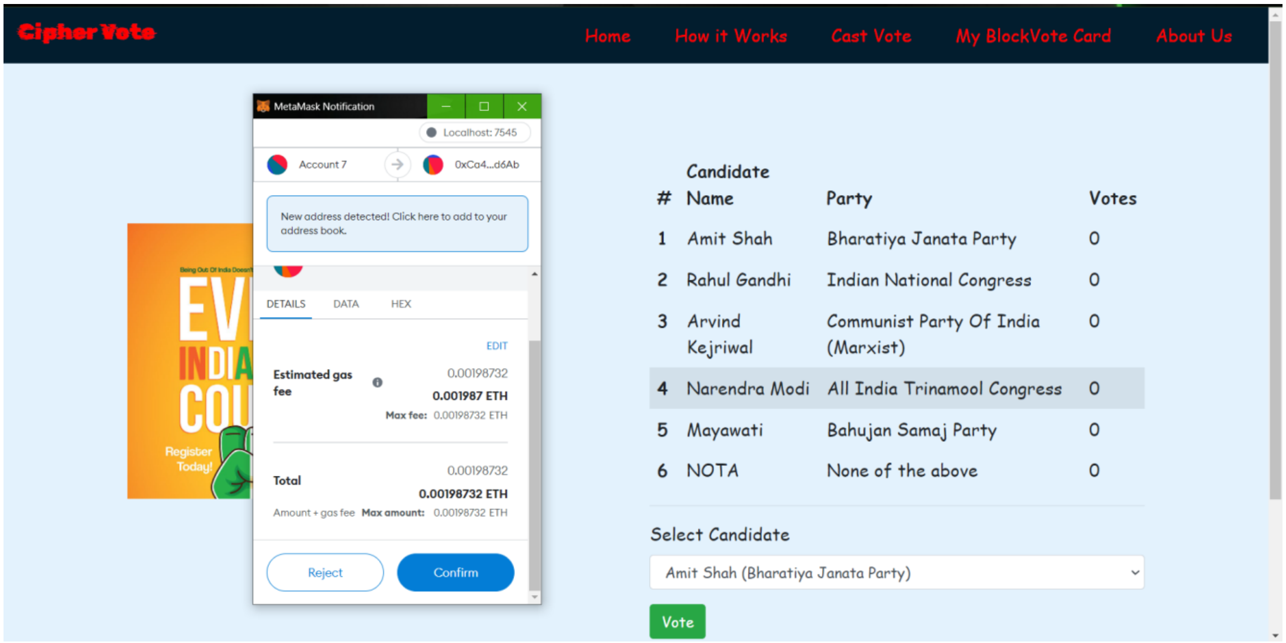
#	Candidate Name	Party	Votes
1	Amit Shah	Bharatiya Janata Party	0
2	Rahul Gandhi	Indian National Congress	0
3	Arvind Kejriwal	Communist Party Of India (Marxist)	0
4	Narendra Modi	All India Trinamool Congress	0
5	Mayawati	Bahujan Samaj Party	0
6	NOTA	None of the above	0

Select Candidate: Amit Shah (Bharatiya Janata Party)

Vote

Your Voter's Address: 0xda379ac7cf6160898509048ac6b9feb6e230b833

Figure 6: Casting the Vote



New address detected! Click here to add to your address book.

Estimated gas fee: 0.00198732
0.001987 ETH
Max fee: 0.00198732 ETH

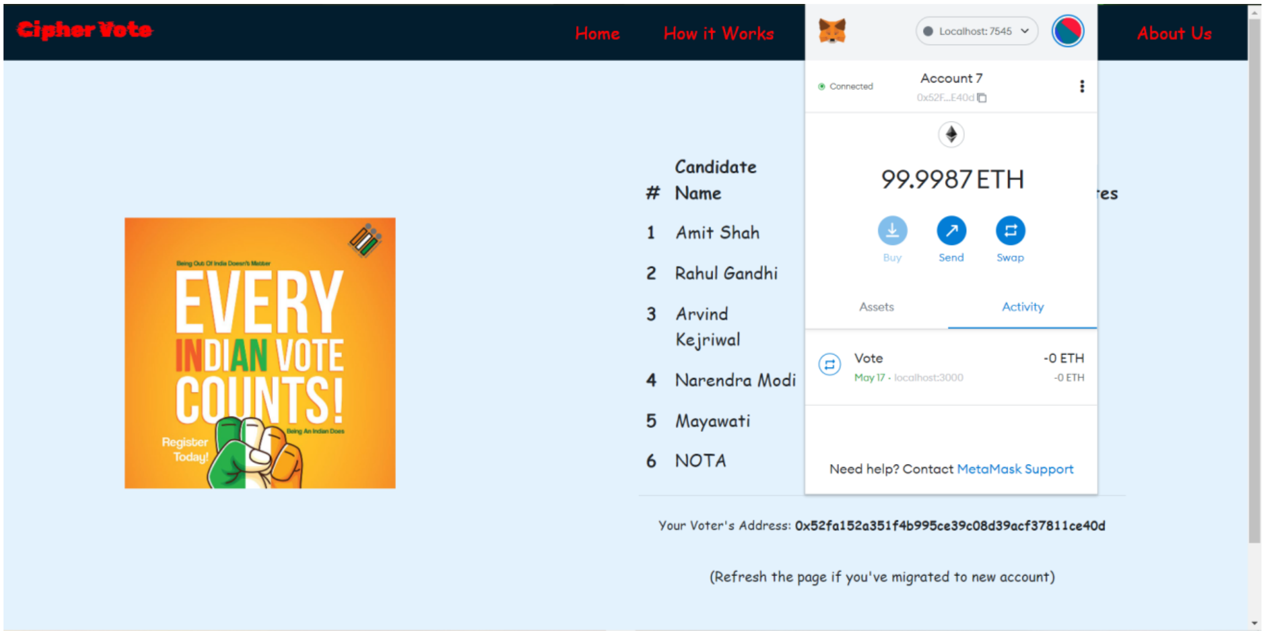
Total: 0.00198732
0.00198732 ETH
Amount + gas fee Max amount: 0.00198732 ETH

#	Candidate Name	Party	Votes
1	Amit Shah	Bharatiya Janata Party	0
2	Rahul Gandhi	Indian National Congress	0
3	Arvind Kejriwal	Communist Party Of India (Marxist)	0
4	Narendra Modi	All India Trinamool Congress	0
5	Mayawati	Bahujan Samaj Party	0
6	NOTA	None of the above	0

Select Candidate: Amit Shah (Bharatiya Janata Party)

Vote

Figure 7: Confirming the transaction to cast vote



#	Candidate Name
1	Amit Shah
2	Rahul Gandhi
3	Arvind Kejriwal
4	Narendra Modi
5	Mayawati
6	NOTA

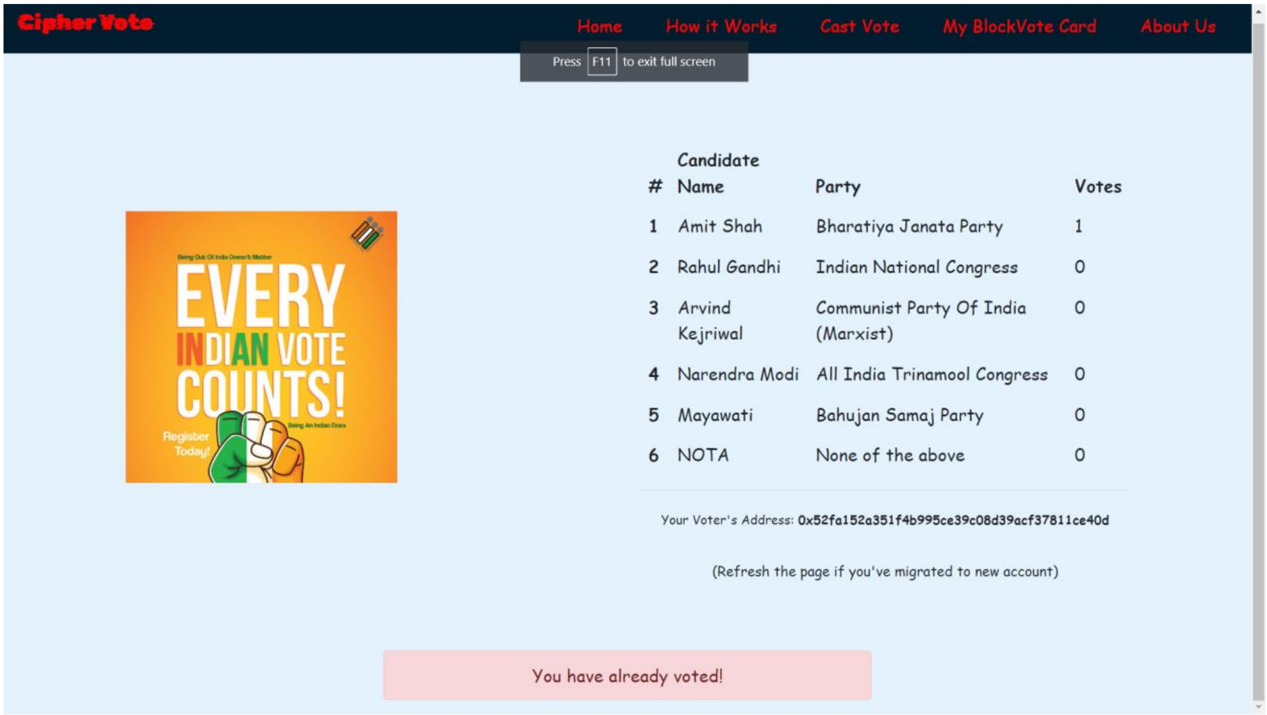
Vote -0 ETH
May 17 · localhost:3000 -0 ETH

Need help? Contact MetaMask Support

Your Voter's Address: 0x52fa152a351f4b995ce39c08d39acf37811ce40d

(Refresh the page if you've migrated to new account)

Figure 8: Transaction confirmed by miners



Press F11 to exit full screen

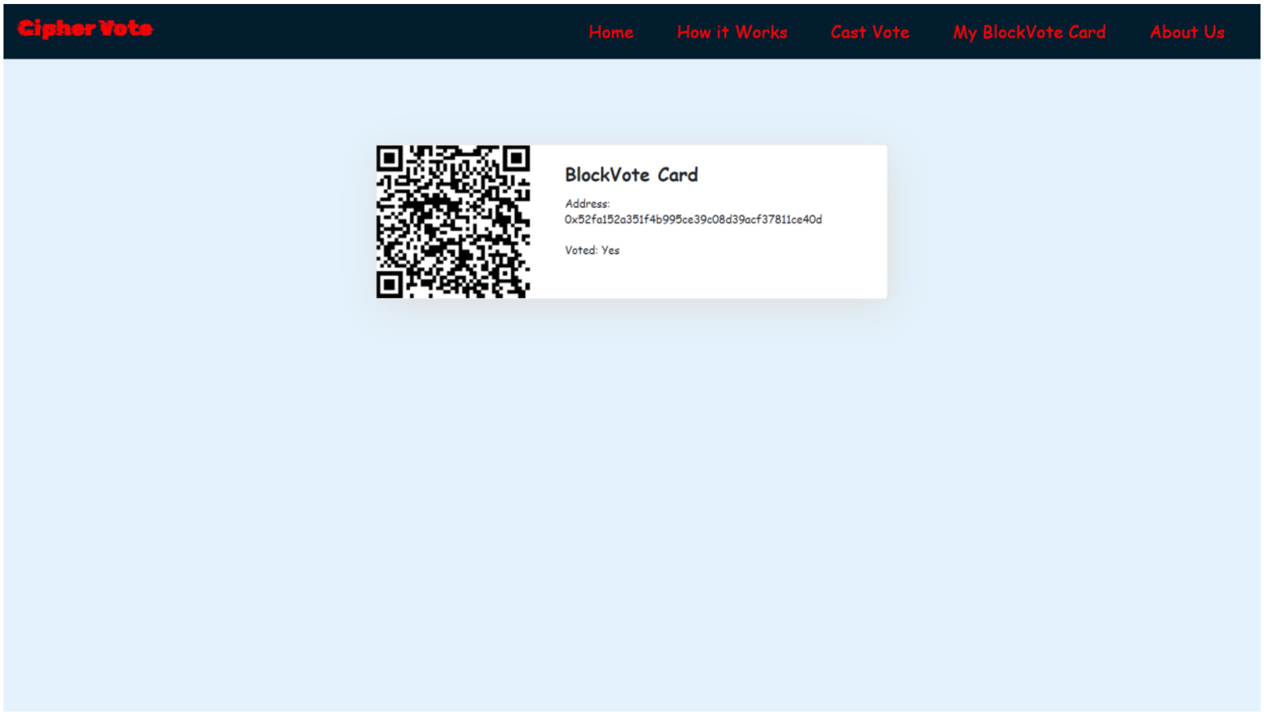
#	Candidate Name	Party	Votes
1	Amit Shah	Bharatiya Janata Party	1
2	Rahul Gandhi	Indian National Congress	0
3	Arvind Kejriwal	Communist Party Of India (Marxist)	0
4	Narendra Modi	All India Trinamool Congress	0
5	Mayawati	Bahujan Samaj Party	0
6	NOTA	None of the above	0

Your Voter's Address: 0x52fa152a351f4b995ce39c08d39acf37811ce40d

(Refresh the page if you've migrated to new account)

You have already voted!

Figure 9: Already Voted Prompt

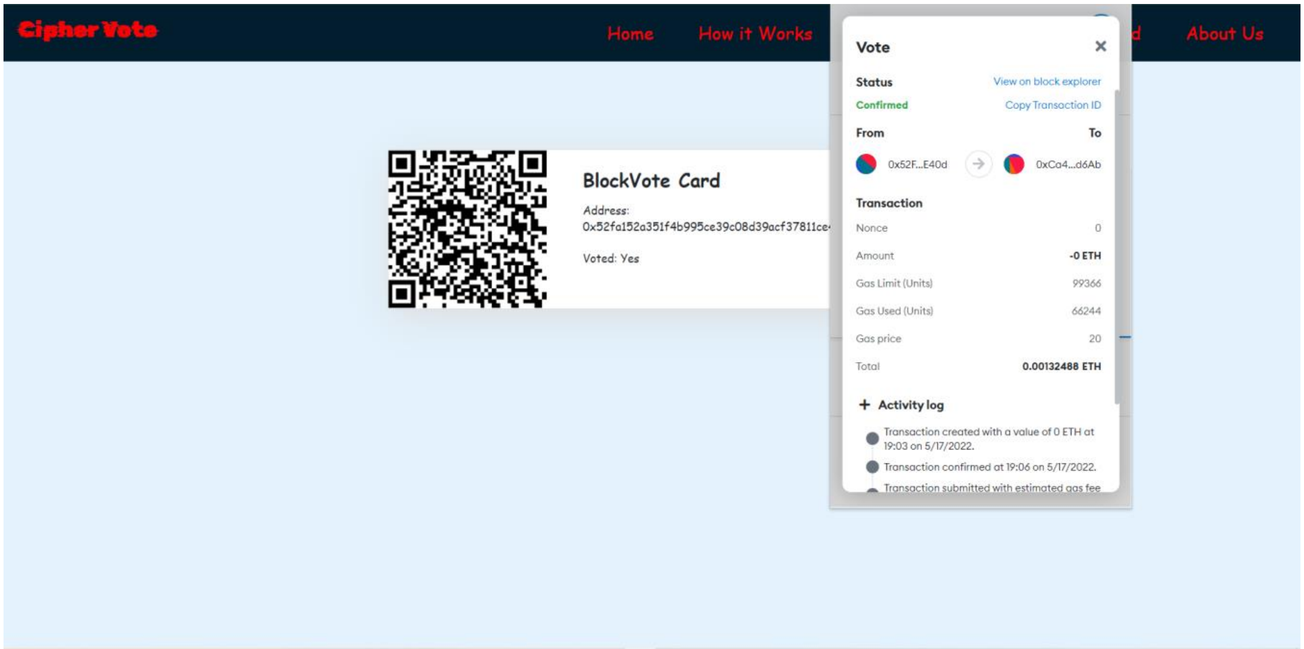


BlockVote Card

Address:
0x52fa152a351f4b995ce39c08d39acf37811ce40d

Voted: Yes

Figure 10: Customized BlockVote Card



BlockVote Card

Address:
0x52fa152a351f4b995ce39c08d39acf37811ce

Voted: Yes

Vote

Status
Confirmed

From 0x52F...E40d
To 0xCa4...d6Ab

Transaction

```
Nonce                    0
Amount               -0 ETH
Gas Limit (Units)    99366
Gas Used (Units)     66244
Gas price               20
Total       0.00132488 ETH
```

- Transaction created with a value of 0 ETH at 19:03 on 5/17/2022.
- Transaction confirmed at 19:06 on 5/17/2022.
- Transaction submitted with estimated gas fee

Figure 11: Transaction Confirmed Log

CHAPTER 3. IMPLEMENTATION

The above system design is translated into a machine-readable form which is termed as coding. It is basically translating the human readable format to a machine friendly one. The code generation step performs this task.

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?

3.1 Coding Standardization

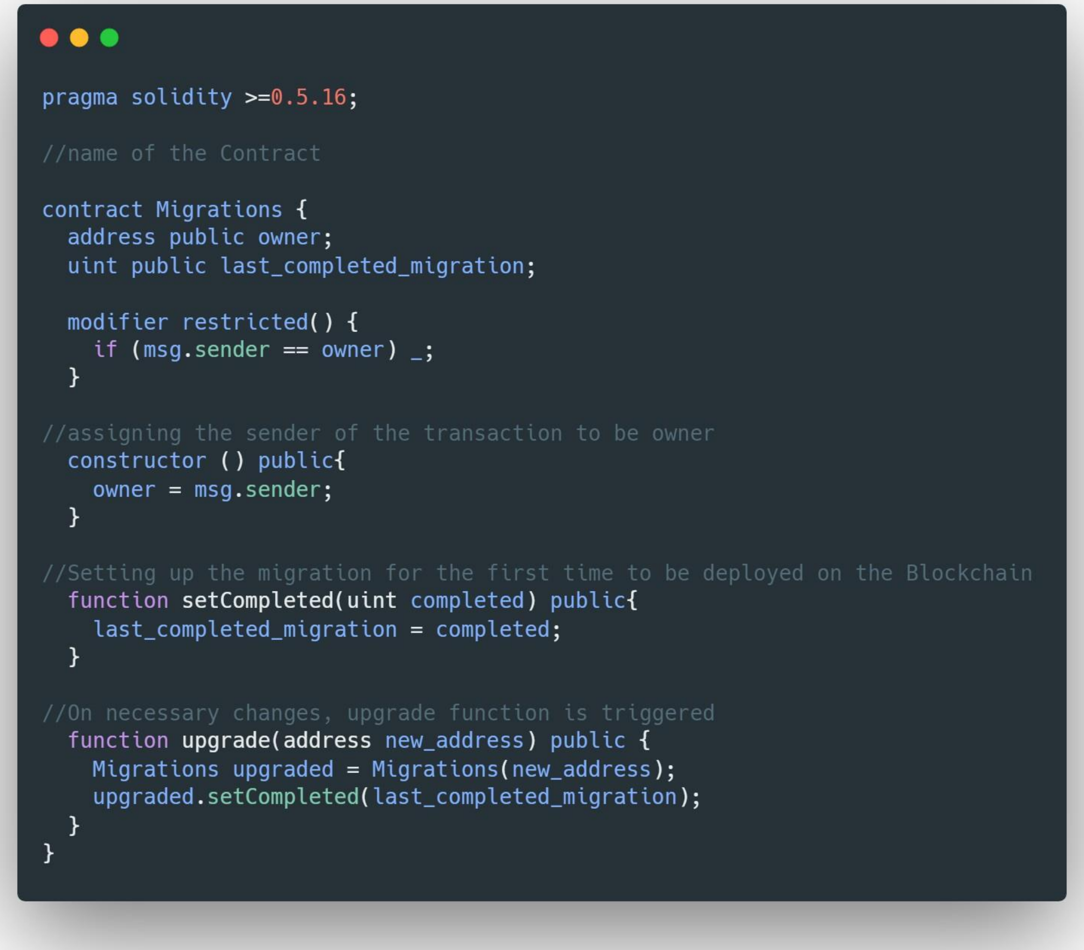
Coding Standardization basically the efficiency of our code which has been converted from the system design. The efficiency primarily depends upon:

- *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 various platform given all the necessary dependencies are installed.

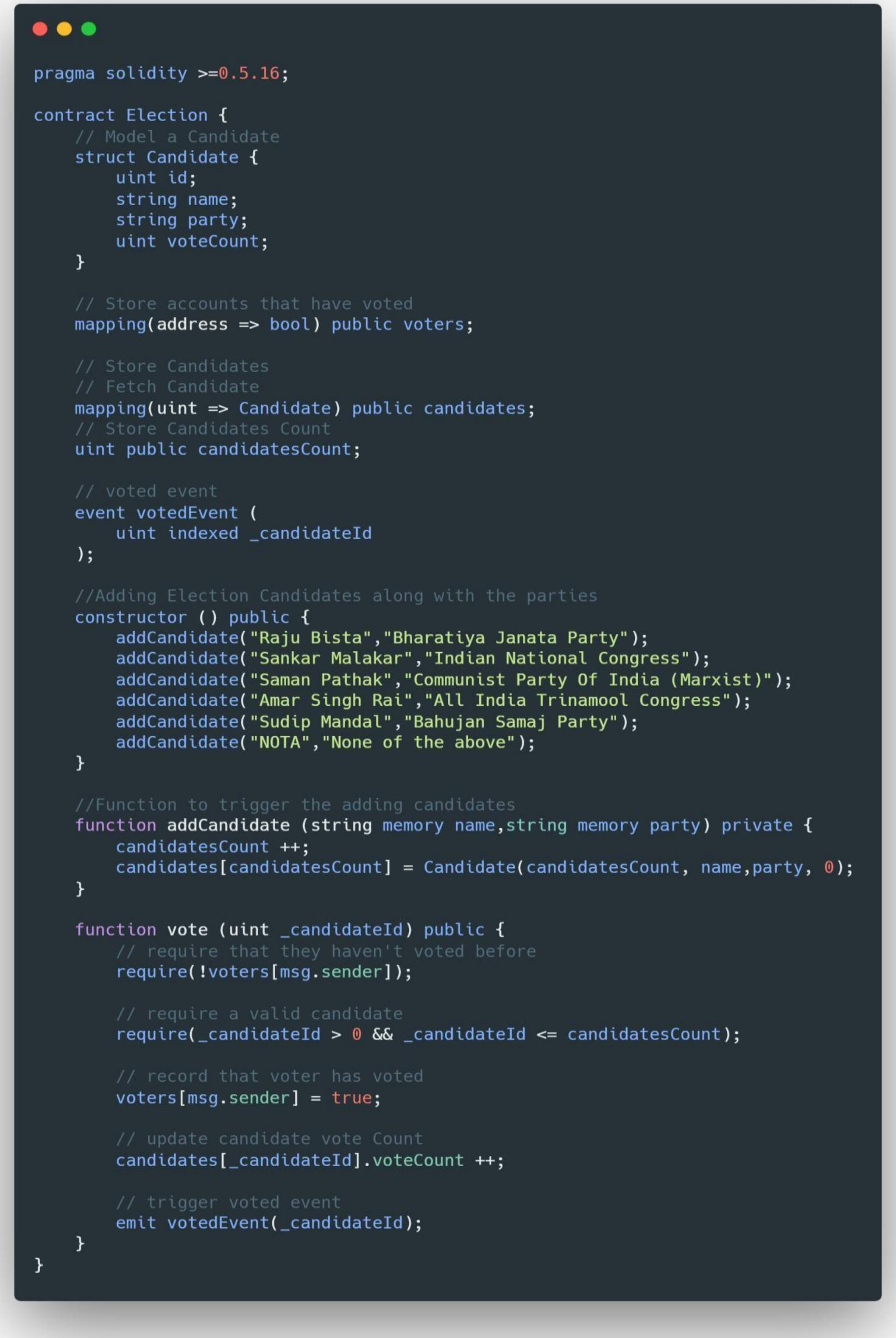
- *Debug Easily*: The coding should be error-free as much as possible.

3.2 Source Code



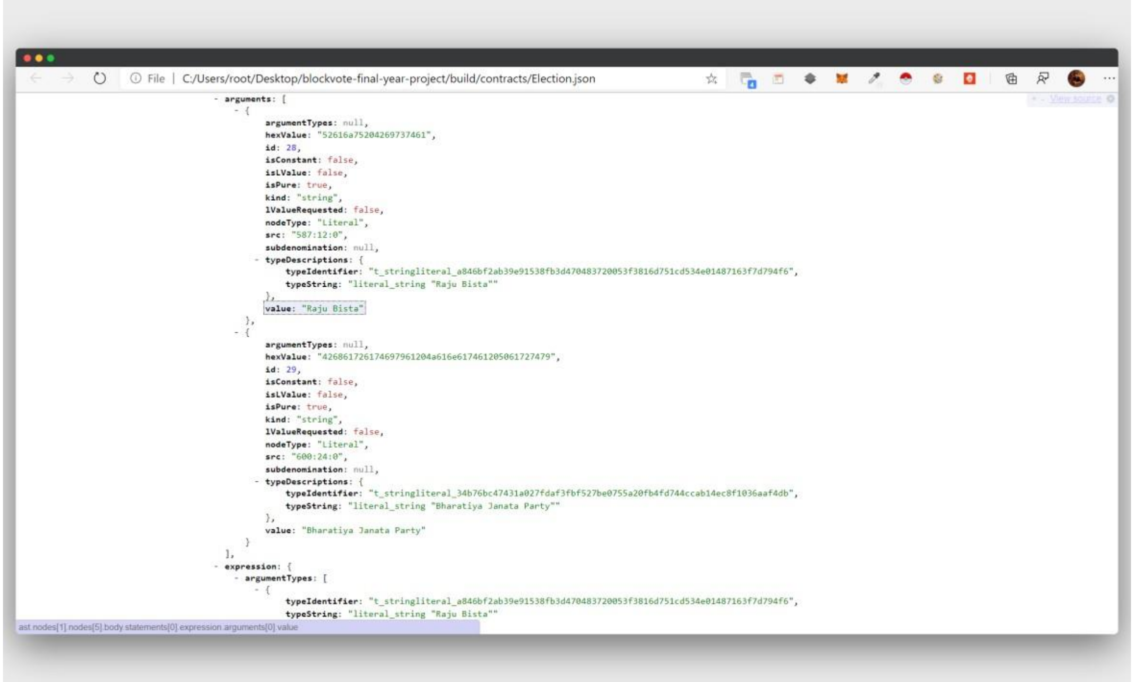
```
pragma solidity >=0.5.16;

//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 12: 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);
    }
}
```

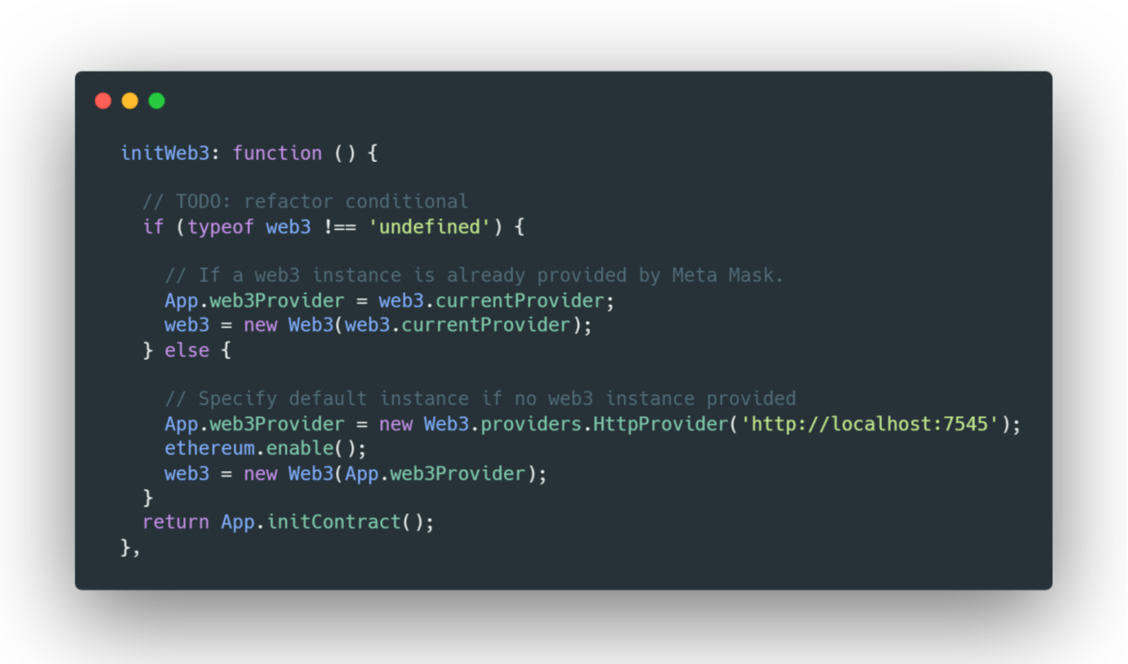
Figure 13 : Election Smart Contract



File | C:/Users/root/Desktop/blockvote-final-year-project/build/contracts/Election.json
View source
ast.nodes[1].nodes[5].body.statements[0].expression.arguments[0].value

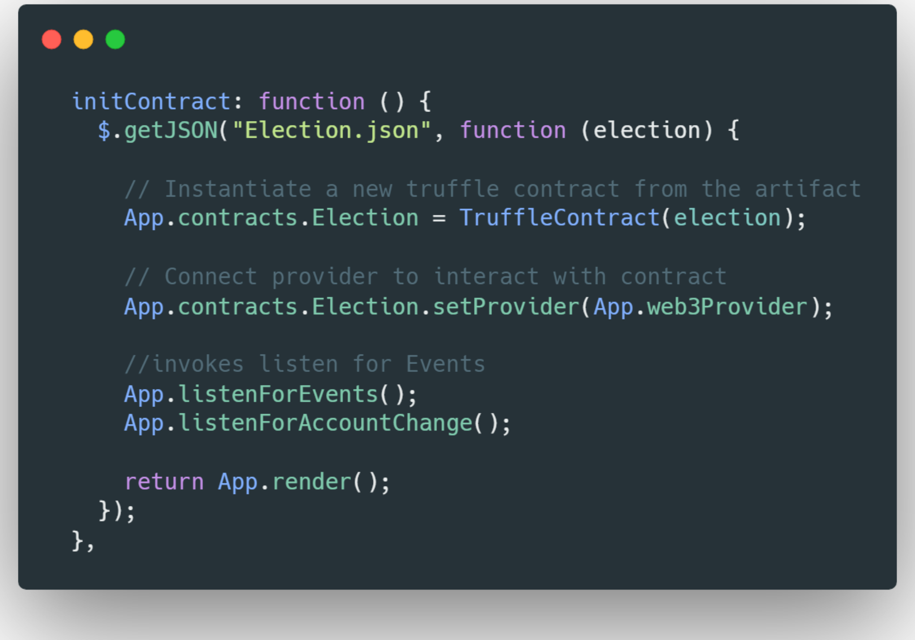
```
- arguments: [
  - {
        argumentTypes: null,
        hexValue: "52616a75204269737461",
        id: 28,
        isConstant: false,
        isLValue: false,
        isPure: true,
        kind: "string",
        lValueRequested: false,
        nodeType: "Literal",
        src: "587:12:0",
        subdenomination: null,
      - typeDescriptions: {
            typeIdentifier: "t_stringliteral_a846bf2ab39e91538fb3d470483720053f3816d751cd534e01487163f7d794f6",
            typeString: "literal_string "Raju Bista""
        },
        value: "Raju Bista"
    },
  - {
        argumentTypes: null,
        hexValue: "4268617261746979612044a616e6174612050617274479",
        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 14: Election API



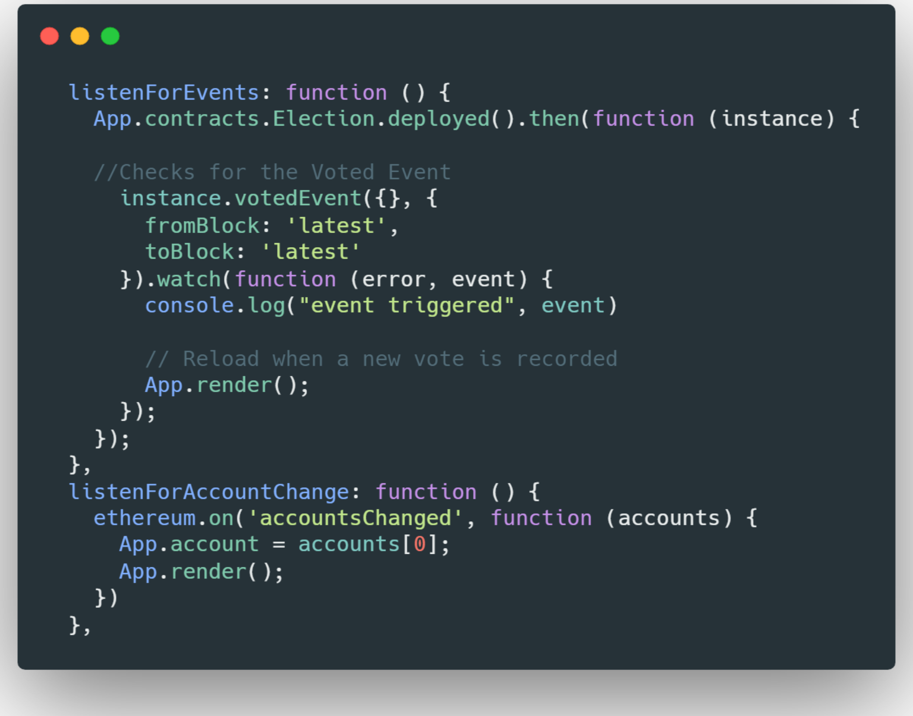
```
initWeb3: function () {

  // 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 15: Initializing the web3 connection on Front-End



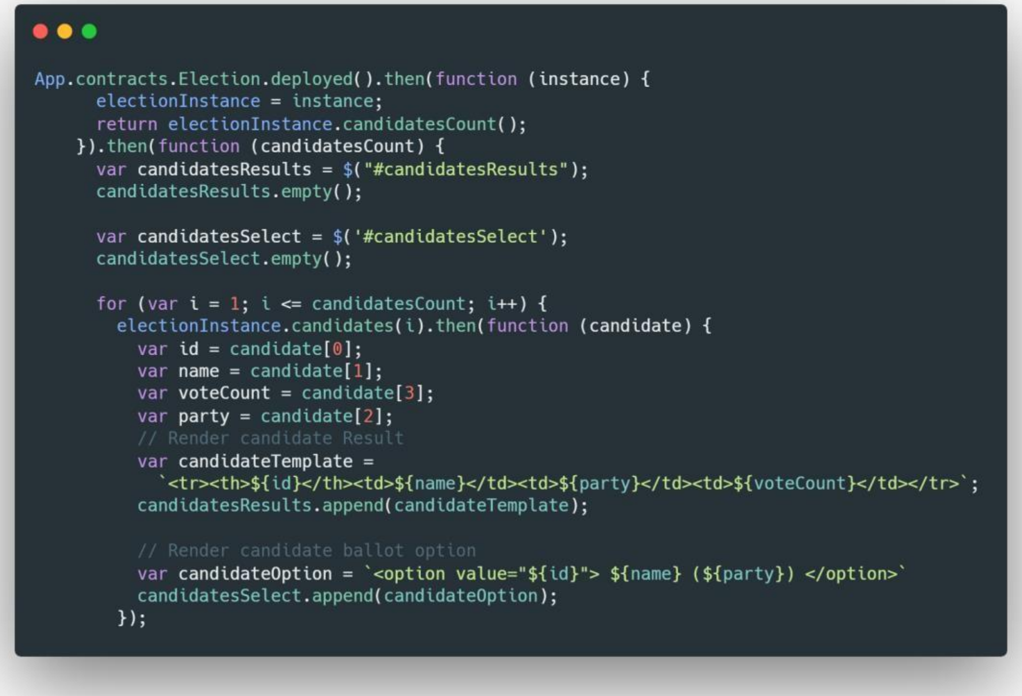
```
initContract: function () {
  $.getJSON("Election.json", function (election) {

    // 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 16: Initializing the smart contract



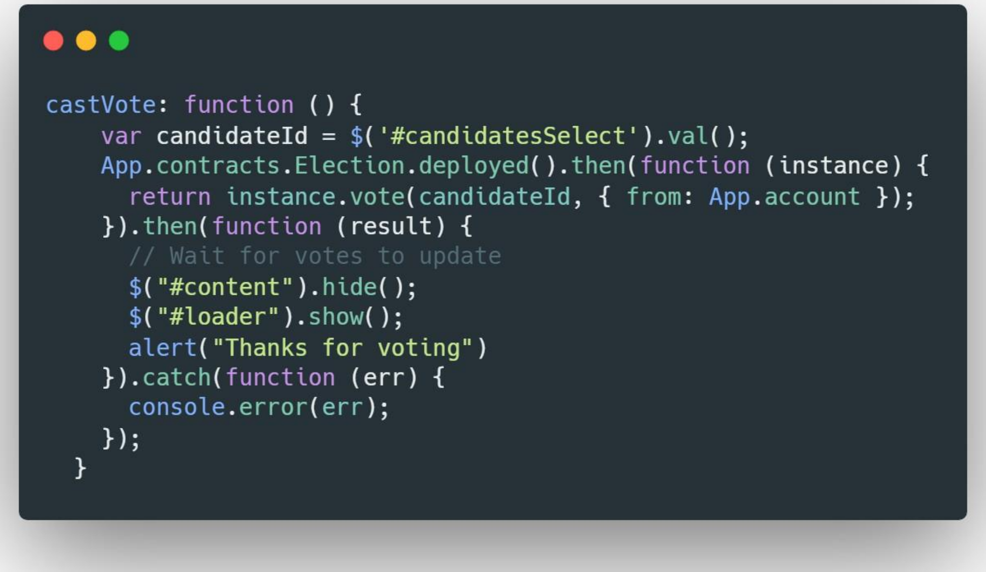
```
listenForEvents: function () {
  App.contracts.Election.deployed().then(function (instance) {

  //Checks for the Voted Event
    instance.votedEvent({}, {
      fromBlock: 'latest',
      toBlock: 'latest'
    }).watch(function (error, event) {
      console.log("event triggered", event)

      // Reload when a new vote is recorded
      App.render();
    });
  });
},
listenForAccountChange: function () {
  ethereum.on('accountsChanged', function (accounts) {
    App.account = accounts[0];
    App.render();
  })
},
```

Figure 17: Trigger voted Events



```
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 18: Front End Integration for the Election



```
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 19: CastVote function to vote

CHAPTER 4. EXPERIMENTAL SETUP

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:

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.

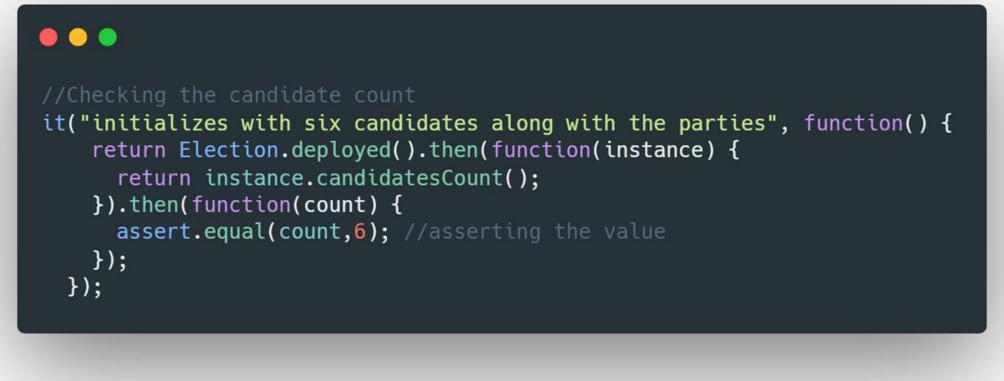
Steps for Unit Testing are:-

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 test cycle until the Unit is free from all types of bugs.

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.

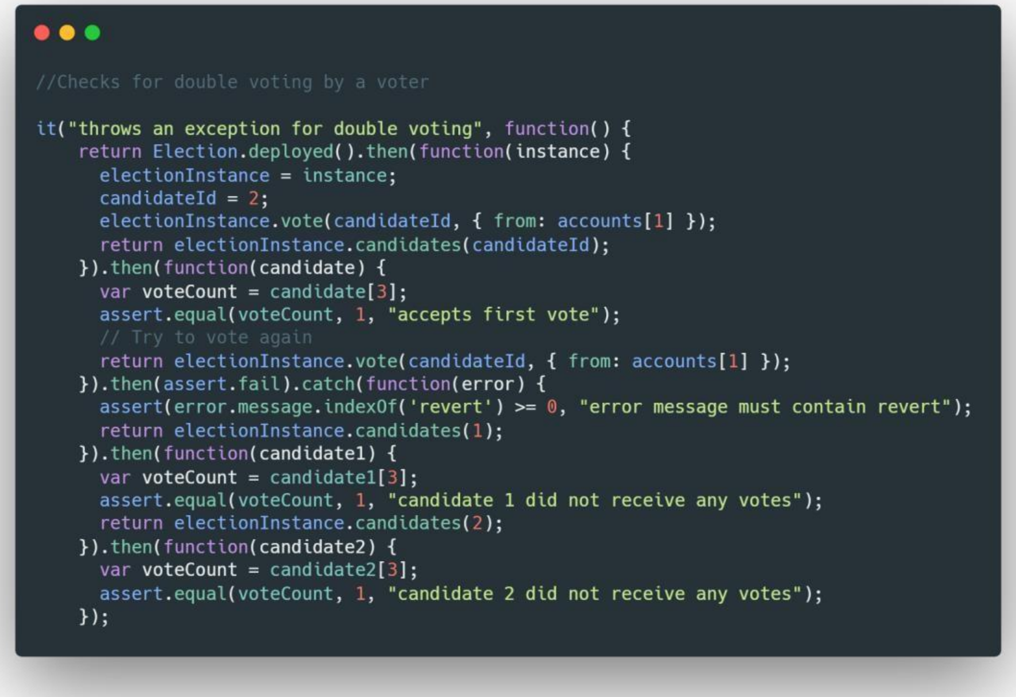
In Bottom Up Integration, we move from the bottom to top i.e. the components below are first written and these are integrated first. The integration happens from bottom to top. If the calling component is yet to be developed, it is replaced by a specially written component called a Driver.

4.1 Testing Designs



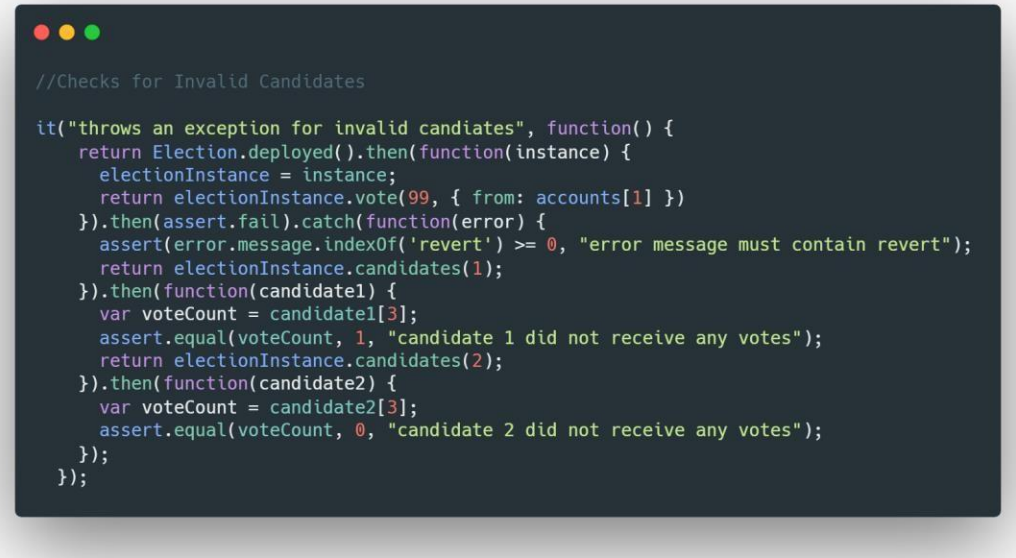
```
//Checking the candidate count
it("initializes with six candidates along with the parties", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
    }).then(function(count) {
      assert.equal(count,6); //asserting the value
    });
  });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

Figure 20: Candidate Count Unit Test



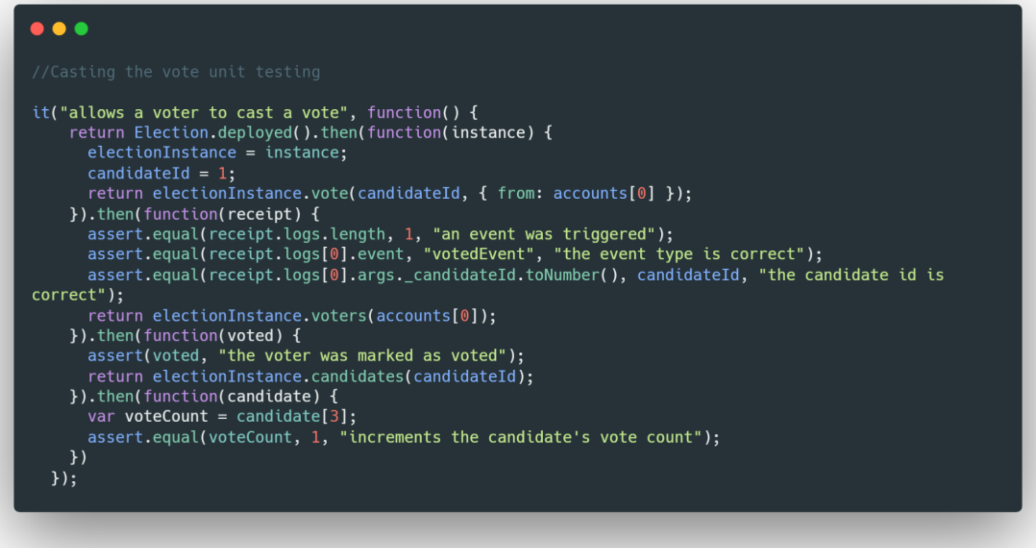
```
//Checks for double voting by a voter

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 21: Double Voting Unit Test



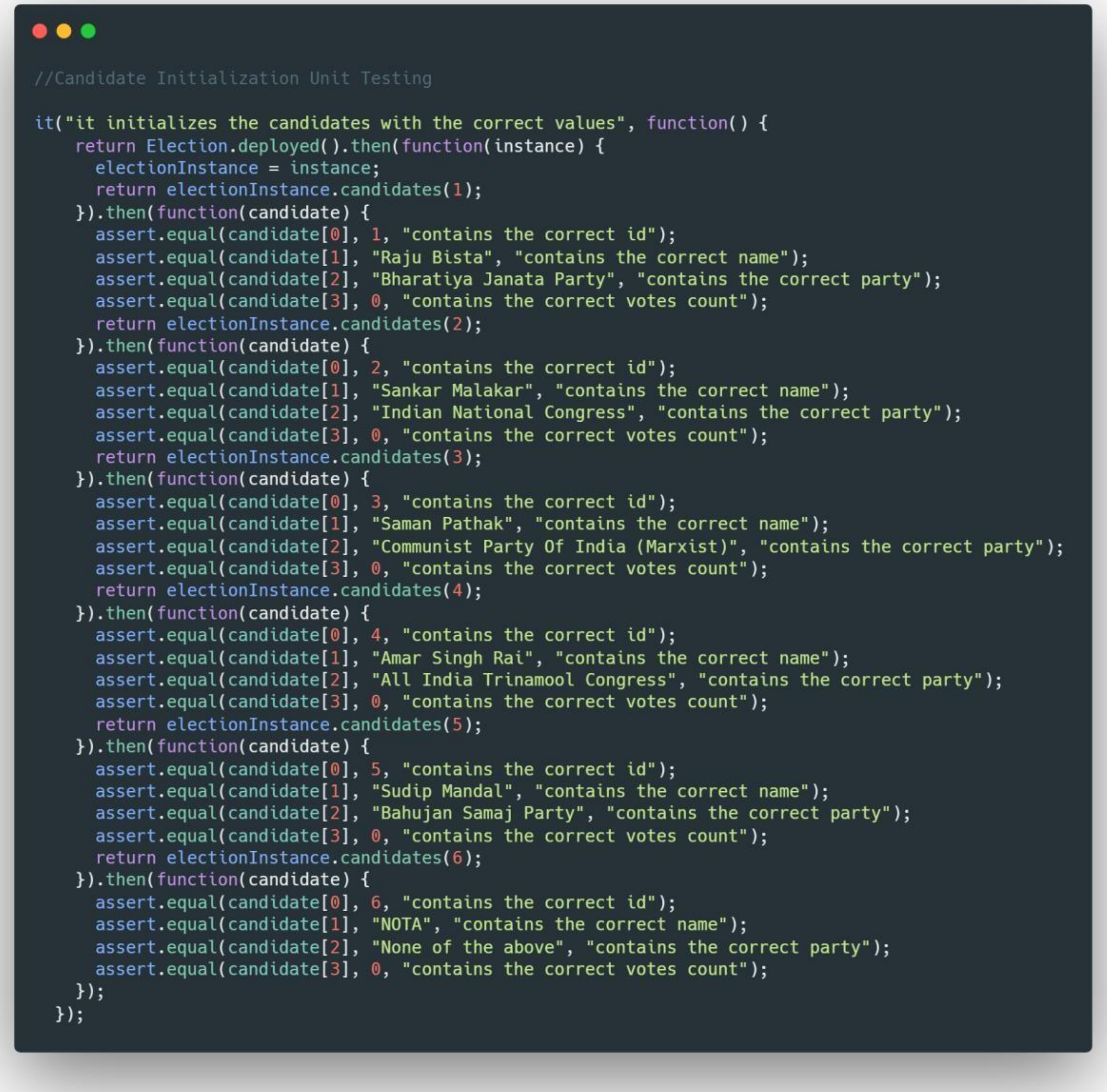
```
//Checks for Invalid Candidates

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 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, 0, "candidate 2 did not receive any votes");
    });
  });
```

Figure 22: Invalid Candidate 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");
    })
  });
```

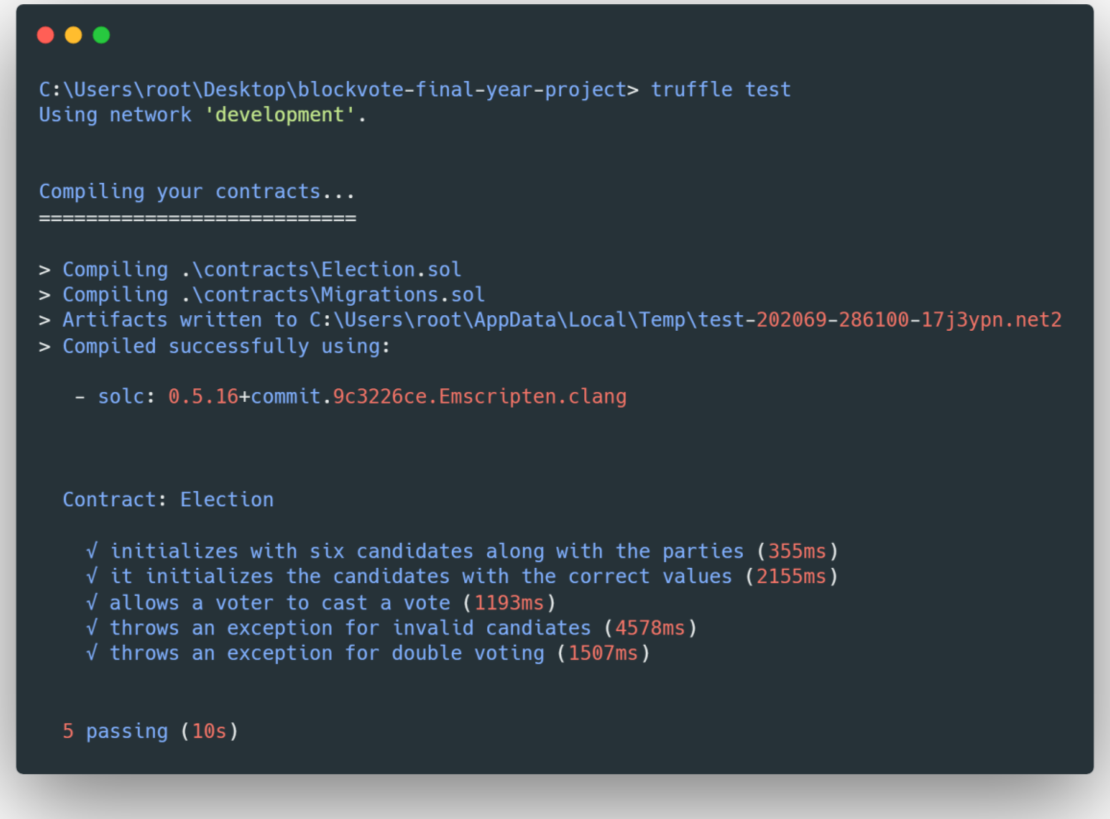
Figure 23: Vote Cast Unit Test



```
//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 24: Candidate Initialization Unit Test

4.2 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 25: Test Report

4.3 System Security Measures

4.3.1 Data Security

Security is about risk management, so it is important to start with an understanding of the risk associated with the blockchain solutions. The specific risks of a blockchain solution depends on the type of blockchain being used. Let's take a look at the various types of blockchains with decreasing level of risks and increasing levels of security:

- **Public Blockchains** are public and anyone can join them and validate transactions. They are generally riskier (for example, cryptocurrencies). This includes risks where anyone can be part of the blockchain without any level of control or restrictions.

- **Private blockchains** are restricted and usually limited to business networks; membership is controlled by a single entity (regulator) or consortium.

- **Permissionless blockchains** have no restrictions on processors.

- **Permissioned blockchains** allow the ledger to be encrypted so that only relevant participants can see it, and only those who meet a need-to-know criterion can decrypt it.

There are a number of other risks with blockchain solutions, and they can be broadly categorized into three areas:

- **Business and governance**: Business risks include financial implications, reputational factors, and compliance risks. Governance risks emanate primarily from the decentralized nature of blockchain solutions, and require strong controls on decision criteria, governing policies, identity, and access management.

- **Process**: These risks are associated with the various processes that a blockchain solution requires in its architecture and operations.

- **Technology**: The underlying technology used to implement various processes and business needs may not always be the best choice, and this can ultimately lead to security risks.

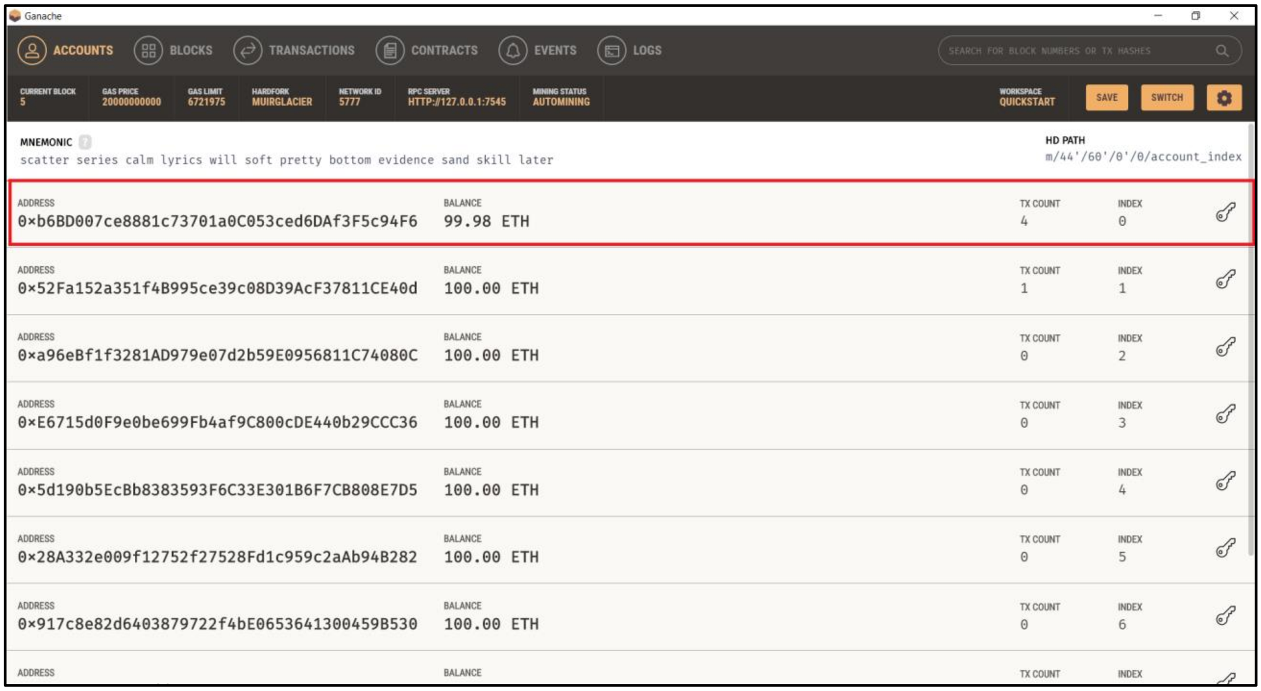
4.3.2 Blockchain Security Threat Models

The security of a solution should also be evaluated in the context of its threat model. Blockchain, by nature, has robust record integrity guarantees, however a number of things can go wrong in other parts of a blockchain-based application that can lead to compromise and loss. Some examples include weak access controls, loose key and certificate management protections, and insufficient communication security. The key to properly securing such an application is to develop a comprehensive threat model for it and mitigate identified weaknesses.

One well-known model is the Spoofing, Tampering, Repudiation, Information disclosure, Denial of service attacks, and Elevation of privilege (STRIDE) model that is used to study relationships between the actors and assets, review threats and weaknesses related to these relationships, and propose appropriate mitigations.

Blockchain applications often incorporate external components — Identity and access management (IAM) systems, multi-factor authentication (MFA), public key infrastructure (PKI), and regulatory and audit systems — that are owned and managed by actors. These systems need to be carefully scrutinized before they can become part of the overall solution as they are developed or controlled by third parties. These should be taken into consideration for the threat model in a blockchain solution.

The user interface of the application is already discussed under the ***System Design.*** Let us have a look at the back-end blockchain server.



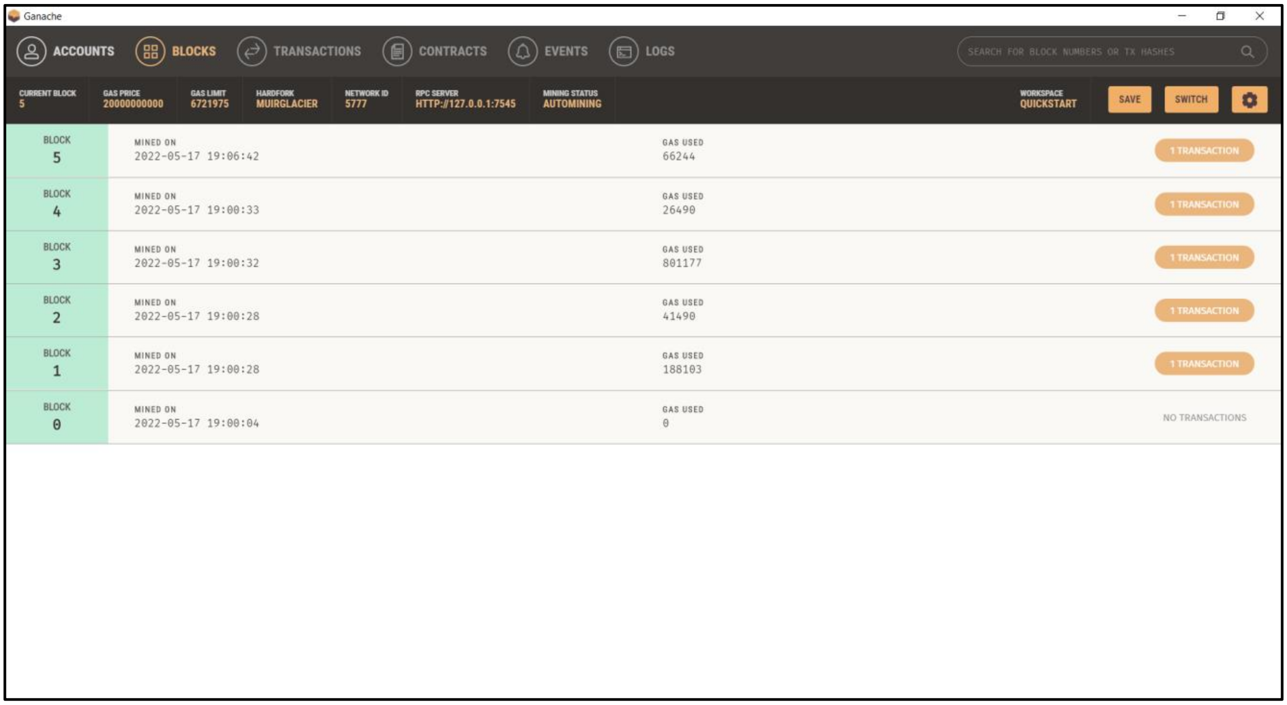
CURRENT BLOCK 5 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE QUICKSTART

MNEMONIC
scatter series calm lyrics will soft pretty bottom evidence sand skill later

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×b6BD007ce8881c73701a0C053ced6DAf3F5c94F6	99.98 ETH	4	0
0×52Fa152a351f4B995ce39c08D39AcF37811CE40d	100.00 ETH	1	1
0×a96eBf1f3281AD979e07d2b59E0956811C74080C	100.00 ETH	0	2
0×E6715d0F9e0be699Fb4af9C800cDE440b29CCC36	100.00 ETH	0	3
0×5d190b5EcBb8383593F6C33E301B6F7CB808E7D5	100.00 ETH	0	4
0×28A332e009f12752f27528Fd1c959c2aAb94B282	100.00 ETH	0	5
0×917c8e82d6403879722f4bE0653641300459B530	100.00 ETH	0	6

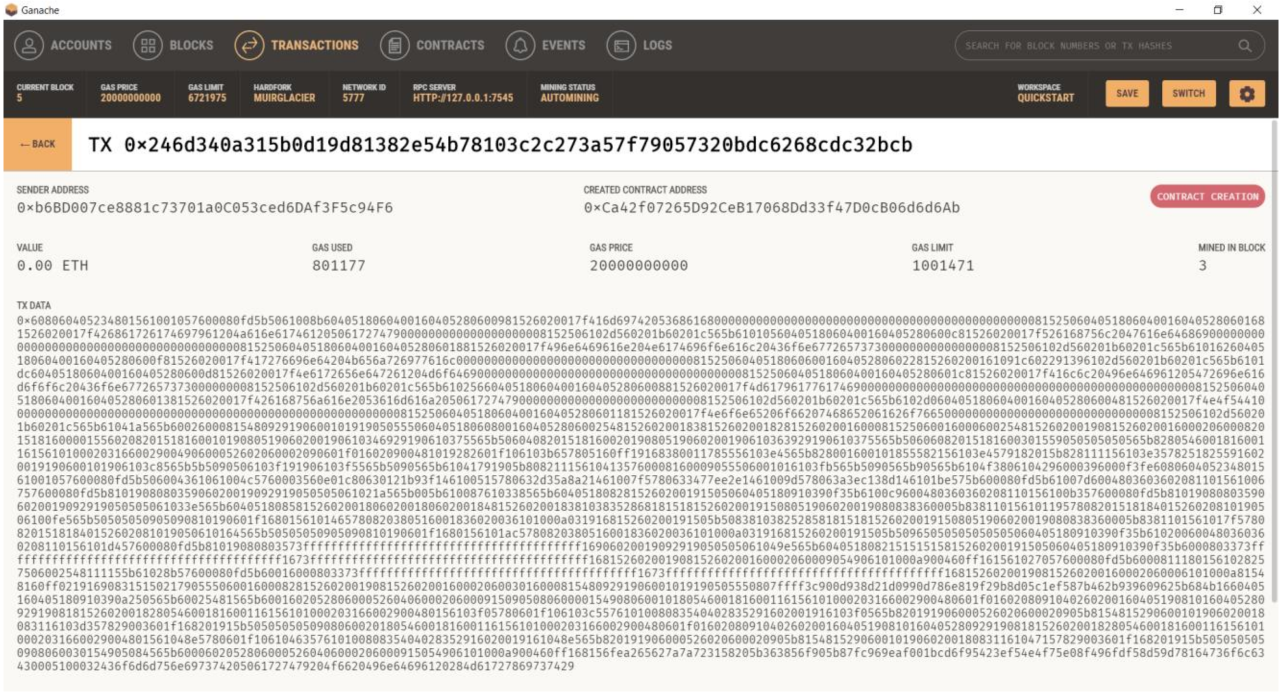
Figure 26: Smart Contract Owner Account



CURRENT BLOCK 5 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE QUICKSTART

BLOCK	MINED ON	GAS USED	
5	2022-05-17 19:06:42	66244	1 TRANSACTION
4	2022-05-17 19:00:33	26490	1 TRANSACTION
3	2022-05-17 19:00:32	801177	1 TRANSACTION
2	2022-05-17 19:00:28	41490	1 TRANSACTION
1	2022-05-17 19:00:28	188103	1 TRANSACTION
0	2022-05-17 19:00:04	0	NO TRANSACTIONS

Figure 27: Blocks Mined after Transactions



```
CURRENT BLOCK  GAS PRICE    GAS LIMIT  HARDFORK     NETWORK ID  RPC SERVER             MINING STATUS   WORKSPACE
5              20000000000  6721975    MUIRGLACIER  5777        HTTP://127.0.0.1:7545  AUTOMINING      QUICKSTART

TX 0×246d340a315b0d19d81382e54b78103c2c273a57f79057320bdc6268cdc32bcb

SENDER ADDRESS                                  CREATED CONTRACT ADDRESS
0×b6BD007ce8881c73701a0C053ced6DAf3F5c94F6      0×Ca42f07265D92CeB17068Dd33f47D0cB06d6d6Ab

VALUE        GAS USED     GAS PRICE      GAS LIMIT     MINED IN BLOCK
0.00 ETH     801177       20000000000    1001471       3

TX DATA
0×608060405234801561001057600080fd5b5061008b6040518060400160405280600981526020017f416d6974205368616800000000000000000000000000000000000000000000008152506040518060400160405280601681
1526020017f4268617261746979612 04a616e617461205061727479000000000000000000008152506102d560201b60201c565b6101056040518060400160405280600c81526020017f526168756c2047616e6468690000000000
00000000000000000000000000000008152506040518060400160405280601881526020017f496e6469616e204e6174696f6e616c20436f6e677265737300000000000000008152506102d560201b60201c565b61016260405
18060400160405280600f81526020017f417276696e64204b656a726977616c00000000000000000000000000000000008152506040518060600160405280602281526020016110 91c602291396102d560201b60201c565b6101
dc6040518060400160405280600d81526020017f4e6172656e647261204d6f64690000000000000000000000000000000000000081525060405180604001604052806 01c81526020017f416c6c20496e646961205472696e616
d6f6f6c20436f6e67726573730000000008152506102d560201b60201c565b6102566040518060400160405280600881526020017f4d617961776174690000000000000000000000000000000000000000000000008152506040
518060400160405280601381526020017f426168756a616e2053616d616a205061727479000000000000000000000000008152506102d560201b60201c565b6102d06040518060400160405280600481526020017f4e4f54410
0000000000000000000000000000000000000000000000000000008152506040518060400160405280601181526020017f4e6f6e65206f662074686520616 26f766500000000000000000000000000008152506102d56020
1b60201c565b61041a565b6002600081548092919060010191905055506040518060800160405280600254815260200183815260200182815260200160008152506001600060025481526020019081526020016000206000820
15181600001556020820151816001019080519060200190610346929190610375565b5060408201518160020190805190602001906103639291906103755 65b5060608201518160030155905050505050565b8280546001816001
16156101000203166002900490600052602060002090601f016020900481019282601f106103b657805160ff19168380011785556103e4565b828001600101855582156103e4579182015b828111156103e3578251825591602
00191906001019061 03c8565b5b5090506103f191906103f5565b5090565b61041791905b808211156104135760008160009055506001016103fb565b5090565b90565b6104f38061042960003960 00f3fe6080604052348015
61001057600080fd5b506004361061004c5760003560e01c80630121b93f146100515780632d35a8a21461007f5780633477ee2e1461009d578063a3ec138d146101be575b600080fd5b61007d6004803603602081101561006
757600080fd5b810190808035906020019092919050505061021a565b005b610087610338565b6040518082815260200191505060405180910390f35b6100c9600480360360208110156100b357600080fd5b81019080803590
6020019092919050505061033e565b604051808581526020018060200180602001848152602001838103835286818151815260200191508051906020019080838360005b8381101561011957808201518184015260208101905
06100fe565b50505050905090810190601f1680156101465780820380516001836020036101000a031916815260200191505b508381038252858181518152602001915080519060200190808383600 05b8381101561017f5780
82015181840152602081019050610164565b50505050905090810190601f1680156101ac5780820380516001836020036101000a031916815260200191505b50965050505050505050506040518091039 0f35b61020060048036036
0208110156101d457600080fd5b81019080803573ffffffffffffffffffffffffffffffffffffffff16906020019092919050505061049e565b60405180821515151581526020019150506040518091039 0f35b6000803373ff
ffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002060009054906101000a900460ff161561027057600080fd5b6000811180156102825
7506002548111155b61028b57600080fd5b60016000803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002060006101000a8154
8160ff02191690831515021790555060016000828152602001908152602001600020600301600081548092919060010191905055 50807ffff3c900d938d21d0990d786e819f29b8d05c1ef587b462b939609625b684b1660405
160405180910390a250565b60025481565b600160205280600052604060002060009150905080600001549080600101805460018160011615610100020316600290048060 1f01602080910402602001604051908101604052 80
92919081815260200182805460018160011615610100020316600290048015610 3f05780601f106103c5576101008083540402835291602001916103f0565b820191906000526020600020905b81548152906001019060200180 18
083116103d357829003601f168201915b505050505090806002018054600181600116156101000203166002900480601f0160208091040260200160405190810160405280929190818152602001828054600181600116156101
0002031660029004801561048e5780601f106104635761010080835404028352916020019161048e565b820191906000526020600020905b81548152906001019060200180831161047157829003601f168201915b505050505
0908060030154905084565b6000602052806000526040600020600091505490610100 0a900460ff168156fea265627a7a723158205b363856f905b87fc969eaf001bcd6f95423ef54e4f75e08f496fdf58d59d78164736f6c63
430005100032436f6d6d756e697374205061727479204f6620496e64696120284d61727869737429
```

Figure 28: Contract Creation Transaction

5.1 Gantt Chart

Gantt chart is a type of a bar chart that is used for illustrating project schedules. Gantt charts can be used in any projects that involve effort, resources, milestones and deliveries. At present, Gantt charts have become the popular choice of project managers in every field. Gantt charts allow project managers to track the progress of the entire project. Through Gantt charts, the project manager can keep a track of the individual tasks as well as of the overall project progression.

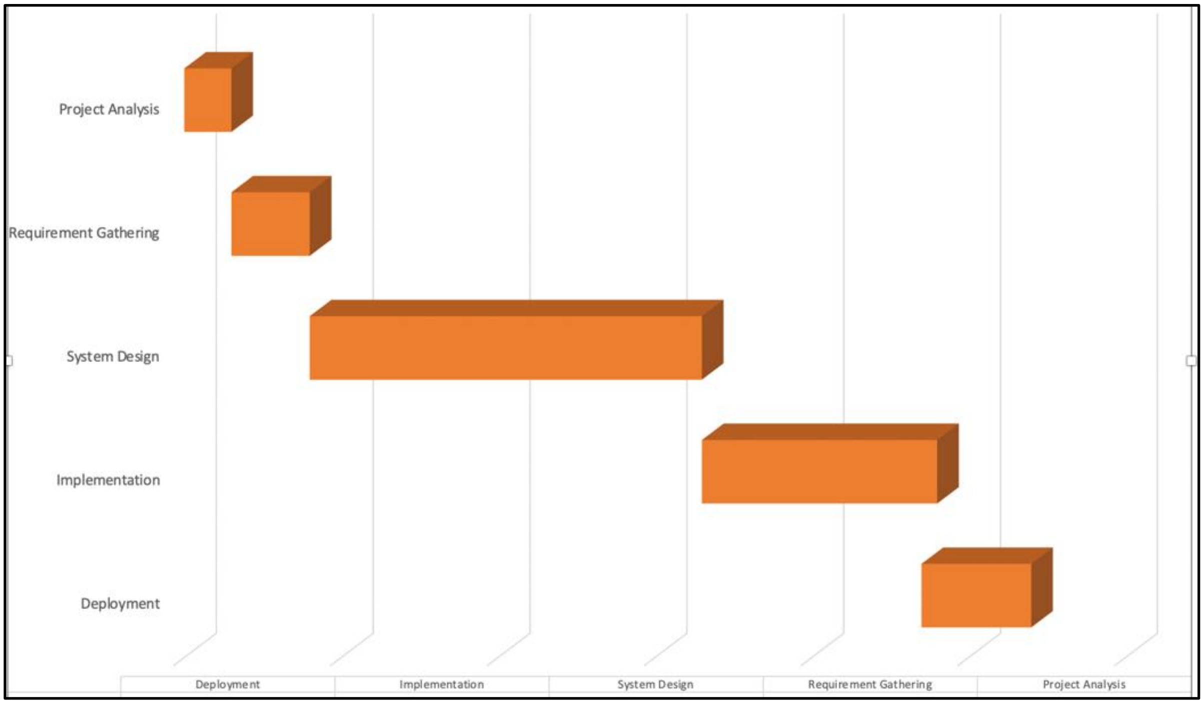


Figure 29: Gantt Chart

Following is the resource allocation table:

Table 3: Resource Allocation

Task Name	Start Date	Days To Complete
Project Analysis	02/02/2024	10
Requirement Gathering	12/02/2024	3
System Design	15/02/2024	17
Implementation	02/03/2024	23
Deployment	25/03/2024	8

CHAPTER 6. OUR OBSERVATION

In examining the research landscape surrounding blockchain-based e-voting systems, several key observations emerge regarding the demonstrated benefits, challenges, and areas for future investigation.

Security emerges as a prominent concern across all analyses, both as a demonstrated benefit and an ongoing challenge. While blockchain technology offers inherent features such as integrity, immutability, and durability, specific vulnerabilities like attacks on keys or smart contracts persist. Proposed remediation techniques such as zero-knowledge proofs, signature schemes, and homomorphic encryption aim to address these concerns.

Privacy, particularly regarding voter anonymity and data protection, consistently ranks high among challenges and future research priorities. This underscores the need for improved privacy-preserving mechanisms within blockchain-based e-voting systems.

Scalability emerges as a significant challenge, often ranked alongside security and privacy concerns. The limitations of blockchain solutions in handling large transaction volumes underscore the need for scalable architectures and consensus mechanisms.

Usability, although not core to blockchain platforms, remains an important consideration, especially in the context of integrating front-end e-voting interfaces. While slightly lower in rank compared to security and privacy, usability is recognized as a vital aspect of overall system functionality.

Coercion-freeness, akin to usability, is consistently noted as an area requiring further demonstration, albeit with potential improvements through blockchain's transparent and secure ledger mechanisms.

Technical concerns, including interoperability, integration with other platforms, and specific blockchain research on consensus protocols and smart contracts, feature prominently in challenges and future work sections.

Transparency and auditability stand out as undisputed benefits of blockchain-based e-voting systems, offering a clear advantage in enhancing trust and accountability in the electoral process.

Other properties such as verifiability, accessibility, accuracy/reliability, and acceptability are recognized as relevant but not critical concerns.

In light of these findings, there is convincing evidence supporting the benefits of blockchain in enhancing security, transparency, decentralization, and privacy in e-voting systems. However, addressing the highlighted challenges is crucial for widespread adoption. Future research should prioritize these areas to advance the understanding and application of blockchain in e-voting, potentially inspiring its application in other domains requiring similar levels of security and efficiency. Collaborative efforts among researchers can foster innovative applications of blockchain in public service beyond e-voting.

CHAPTER 7. CONCLUSION AND FUTURE SCOPE

Blockchain technology demonstrates promise in implementing e-voting systems, particularly in ensuring transparency and auditability. However, security and privacy remain central concerns, with blockchain technology offering potential solutions but also facing significant challenges in these areas.

Scalability emerges as a critical limitation of blockchains, highlighting the need for scalable architectures to support large-scale e-voting processes. Beyond core platform concerns, usability, verifiability, accessibility, reliability, and acceptability are also important considerations that require further attention in wider e-voting system implementations.

The study effectively clarifies both the potential and limitations of blockchain-based e-voting systems by integrating an analysis of fundamental properties with practical technological implementations. It provides insights into concrete solution techniques and offers a roadmap for future research, contributing to a comprehensive understanding of the topic and informing decision-making in the development of e-voting systems.

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 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. It also opens the door for a more direct form of democracy, allowing voters to express their will on individual bills and propositions.

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] S. Wolchok, et al., “Security analysis of India’s electronic voting machines,” Proc. 17th ACM Conference on Computer and Communications Security, Acad. Med., 1-14, 2010

[2] J. D. Ohlin, “Did Russian cyber interference in the 2016 election violate international law,” *Tex. L. Rev.*, vol. 95, p. 1579, 2016.

[3] A. B. Ayed, "A conceptual secure blockchain-based electronic voting system." International Journal of Network Security & Its Applications 9.3, 2017, pp. 01-09.

[4] R. Hanifatunnisa and Budi Rahardjo, “Blockchain based e-voting recording system design.” 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.

[6] Y. Liu and Q.Wang, “An E-voting Protocol Based on Blockchain.” IACR Cryptal. Enprint Arch. Vol 8, 2017.

[7] Shahzad, B., J. Crowcroft ,“Trustworthy Electronic Voting Using Adjusted Blockchain Technology”, *IEEE Access*, vol. 7, pp. 24477-24488, 2019.

[8] K. M. Khan et al., “Secure digital voting system based on blockchain technology,” *Int. J. Electron. Gov. Res.*, vol. 14, no. 1, pp. 53-62, 2018

[9] K. R. Alam et al., “An untraceable voting scheme based on pairs of signatures,” *Int. J. Netw. Secur.*, vol. 20, 2018.

[10] B. Adida et al., “Electing a university president using open-audit voting: Analysis of real-world use of Helios,” *EVT/WOTE*, vol. 9, no. 10, 2009.

[11] E. Ben-Sasson et al., *SNARKs for C: Verifying Program Executions Succinctly*, 2013.

[12] D. Chaum, “Secret-ballot receipts: True voter-verifiable elections,” IEEE Security Privacy Mag., vol. 2, no. 1, 38-47, 2004

[13] A. Parsovs, *Homomorphic Tallying for the Estonian Internet Voting System. IACR Cryptology*, 2016.

[14] Hudson, “EIP’s Secret-ballot receipts: True voter-verifiable elections,” *IEEE Sec. Privacy*, 2017.

[15] S. Nakamoto, “Bitcoin: a peer-to-peer electronic cash system”, [Online]. Available: https://bitcoin.org/bitcoin.pdf .

[16] Ali Kaan Koç, Emre Yavuz, Umut Can Çabuk, Gökhan Dalkılıç “Towards Secure E-Voting Using Ethereum Blockchain”

[17] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger", Ethereum Project Yellow Paper, vol. 151, pp. 1-32, 2014.

[18] C.D. Clack, V.A. Bakshi, and L. Braine, “Smart contract templates: foundations, design landscape and research directions”, Mar 2017, arXiv:1608.00771.

[19] E. Maaten, “Towards remote e-voting: Estonian case”, Electronic Voting in Europe-Technology, Law, Politics and Society, vol. 47, pp. 83-100, 2004.

[20] U.C. Çabuk, A. Çavdar, and E. Demir, "E-Demokrasi: Yeni Nesil Doğrudan Demokrasi ve Türkiye’deki Uygulanabilirliği", [Online] Available: https://www.researchgate. net/profile/Umut_Cabuk/ publication/ 308796230_E-Democracy_The_Next_Generation_Direct_Democracy_and_ pplicability_in_ Turkey/links/5818a6d408 a ee7cdc685b40b/E-Democracy-The-Next-GenerationDirect Democracy-and-Applicability-in-Turkey.pdf.

[21] "Final report: study on eGovernment and the reduction of administrative burden (SMART 2012/0061)", 2014, [Online]. Available: https://ec.europa.eu/digital-singlemarket/en/news/finalreport-study-egovernment-and-reductionadministrative-burdensmart-20120061

[22] F. Hao and P.Y.A. Ryan, Real-World Electronic Voting: Design, Analysis and Deployment, CRC Press, pp. 143-170, 2017.

[23] N. Braun, S. F. Chancellery, and B. West. "E-Voting: Switzerland's projects and their legal framework–In a European context", Electronic Voting in Europe: Technology, Law, Politics and Society. Gesellschaft für Informatik, Bonn, pp.43-52, 2004.

[24] Nir Kshetri, Jeffrey Voas, “Blockchain-Enabled E-Voting”.

[25] P. McCorry, S.F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy", International Conference on Financial Cryptography and Data Security. Springer, Cham, pp. 357-375, 2017.

[26] U.C. Çabuk, T. Şenocak, E. Demir, and A. Çavdar, “A Proposal on initial remote user enrollment for IVR-based voice authentication systems”, Int. J. of Advanced Research in Computer and Communication Engineering, vol 6, pp.118-123, July 2017.

[27] Y. Takabatake, D. Kotani, and Y. Okabe, “An anonymous distributed electronic voting system using Zerocoin“, IEICE Technical Report, pp. 127-131, 2016.

[28] Hu, J.; Zhu, P.; Qi, Y.; Zhu, Q.; Li, X. A patent registration and trading system based on blockchain. *Expert Syst. Appl.* **2022**, *201*, 117094.

[29] Zhu, P.; Hu, J.; Li, X.; Zhu, Q. Using blockchain technology to enhance the traceability

of original achievements. *IEEE Trans. Eng. Manag.* **2023**, *70*, 1693–1707.

[30] Abdelmaboud, A.; Ahmed, A.I.A.; Abaker, M.; Eisa, T.A.E.; Albasheer, H.; Ghorashi, S.A.; Karim, F.K. Blockchain for IoT applications: Taxonomy, platforms, recent advances, challenges and future research directions. *Electronics* **2022**, *11*, 630.

[31] Taş, R.; Tanrıöver, Ö.Ö. A systematic review of challenges and opportunities of blockchain for E-voting. *Symmetry* **2020**, *12*, 1328.

[32] Jafar, U.; Ab Aziz, M.J.; Shukur, Z. Blockchain for Electronic Voting System—Review and Open Research Challenges. *Sensors* **2021**, *21*, 5874.

[33] Pawlak, M.; Poniszewska-Marańda, A. Trends in blockchain-based electronic voting systems. *Inf. Process. Manag.* **2021**, *58*, 102595.

[33] Huang, J.; He, D.; Obaidat, M.S.; Vijayakumar, P.; Luo, M.; Choo, K.-K.R. The application of the blockchain technology in voting systems: A review. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–28.

[34] Jafar, U.; Ab Aziz, M.J. A state of the art survey and research directions on blockchain based electronic voting system. In Proceedings of the Second International Conference, ACeS 2020, Penang, Malaysia, 8–9 December 2020; Revised Selected Papers 2. Springer: Singapore, 2021.

[35] Devi, U.; Bansal, S. Secure e-Voting System—A Review. In Proceedings of the Hybrid Intelligent Systems, Olten, Switzerland; Porto, Portugal; Vilnius, Lithuania; Kochi, India, 12–14 December 2023; Springer Nature: Cham, Switzerland, 2023; pp. 1209–1224.

[36] Benabdallah, A.; Audras, A.; Coudert, L.; El Madhoun, N.; Badra, M. Analysis of Blockchain Solutions for E-Voting: A Systematic Literature Review. *IEEE Access* **2022**, *10*, 70746–70759.

[37] Jafar, U.; Ab Aziz, M.J.; Shukur, Z.; Hussain, H.A. A Systematic Literature Review and Meta-Analysis on Scalable Blockchain-Based Electronic Voting Systems. *Sensors* **2022**, *22*, 7585.

[38] Vladucu, M.-V.; Dong, Z.; Medina, J.; Rojas-Cessa; R.Vladucu, M.-V.; Dong, Z.; Medina, J.; Rojas-Cessa, R. E-Voting Meets Blockchain: A Survey. *IEEE Access* **2023**, *11*, 23293–23308.

[39] Luxoft. Available online: **https://www.luxoft.com/** (accessed on 18 November 2023).

[40] Votem. Available online: **https://votem.com/** (accessed on 20 November 2023).

[41] Voatz. Available online: **https://voatz.com/** (accessed on 20 November 2023).

[42] Polyas. Available online: https://www.polyas.com/ (accessed on 21 November 2023).

[43] Kaspersky Box. accessed on 21 November 2023).

[44] A. Kugusheva and Y. Yanovich, “Ring signature-based voting on blockchain,” in Proc. 2019 2nd International Conference on Blockchain Technology and Applications, Xi’an, China, vol. 2020. New York, NY, USA: Association for Computing Machinery, 2019, pp. 70–75. [doi: 10.1145/3376044.3376054].

[45] X. Haiyan, W. Lifang, and W. Yuechuan, “A new fair electronic contract signing protocol,” in Proc. Advances in Intelligent Networking and Collaborative Systems (INCoS-2019), Ōita, Japan. Cham, Switzerland: Springer International Publishing, 2020, pp. 289–295. [doi: 10.1007/978-3-030-29035-1_28].

[46] F. P. Hjálmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjálmtýsson, “Blockchain-based E-voting system,” in Proc. 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, vol. 2018. Piscataway, NJ, USA: IEEE, Jul. 2–7 2018, pp. 983–986. [doi: 10.1109/CLOUD.2018.00151].

[47] M. Kumar, C. P. Katti, and P. C. Saxena, “A secure anonymous e-voting system using identity-based blind signature scheme,” in Proc. 13th International Conference. Mumbai, India: ICISS. Cham, Switzerland: Springer International Publishing, Dec. 16–20 2017, 29–49. [doi: 10.1007/978-3-319-72598-7_3].

[48] A. Russo, A. F. Anta, M. I. G. Vasco, and S. P. Romano, “Chirotonia: A Scalable and Secure e-Voting Framework based on Blockchains and Linkable Ring Signatures,” in Proc. 2021 IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, vol. 2021. Piscataway, NJ, USA: IEEE, Dec. 6–8 2021, pp. 417–424. [doi: 10.1109/ Blockchain 53845.2021.00065].

[49] O. Ikundi, K. C. Nwosu, and M. Abdulgader, “LegitVote: A blockchain-based system to facilitate E-voting process,” In Proceedings of the 2022 International Conference on Computer and Applications (ICCA). Cairo, Egypt: IEEE: Piscataway. NJ, USA, 2022, pp. 1–7. [Google Scholar] [doi: 10.1109/ICCA56443.2022.10039527].

[50] F. Fusco, M. I. Lunesu, F. E. Pani, and A. Pinna, “Crypto-voting, a Blockchain based e-Voting System,” in Proc. 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018), vol. 3. Seville, Spain: KMIS, Sept. 18–20 2018, pp. 223–227. [doi: 10.5220/0006962102230227].

[51] S. K. Vivek, R. S. Yashank, Y. Prashanth, N. Yashas, and M. Namratha, “E-voting systems using blockchain: An exploratory literature survey,” in Proc. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, vol. 2020. Piscataway, NJ, USA: IEEE, Jul. 15–17 2020, pp. 890–895. [doi: 10.1109/ICIRCA48905.2020.9183185] .

[52] S. Mello-Stark and E. A. Lamagna, “The need for audit-capable E-voting systems,” in Proc. 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), Taipei, Taiwan, vol. 2017. Piscataway, NJ, USA: IEEE, Mar. 27–29 2017, pp. 535–540. [doi: 10.1109/WAINA.2017.87] .

[53] J. Hsu and G. Bronson, “E-voting technologies usability: A critical element for enabling successful elections,” in Emerging Challenges in Business, Optimization, Technology, and Industry, Proc. Third International Conference on Business Management and Technology, Vancouver, BC, Canada. Cham, Switzerland: Springer International Publishing, 2018, 61–78. [doi: 10.1007/978-3-319-58589-5_5].