Major Project Report

On

BLOCKCHAIN BASED E-VOTING SYSTEM USING ETHEREUM SMART CONTRACTS

By:

Sneha Gupta (Regd. No: 201811502)

Hridayan Phukan (Regd. No.: 201600535)

In partial fulfilment of requirements for the award of degree in Bachelor of Technology in Computer Science and Engineering (2020).

Under the Project Guidance of

Dr. Ruhul Islam,
Associate Professor,

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
SIKKIM MANIPAL INSTITUTE OF TECHNOLOGY.
(A constituent college of Sikkim Manipal University)
Majhitar, Rangpo, East Sikkim - 737136

Dated: 28 June, 2021

SMIT SIKKIM MANIPAL UNIVERSITY
SIKKIM MANIPAL INSTITUTE OF TECHNOLOGY

PROJECT COMPLETION CERTIFICATE

This is to certify that the below mentioned student of Sikkim Manipal Institute of Technology have worked under my supervision and guidance from **27 Feb 2021 to 28 June 2021** and have successfully completed the project entitled **"Blockchain Based E-Voting System using Ethereum Smart Contract"** in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering.

University Registration No	Name of Student(s)	Course
201811502	Sneha Gupta	B. Tech
201600535	Hridayan Phukan	B. Tech

Dr. Ruhul Islam
Associate Professor
Sikkim Manipal Institute of Technology

PROJECT REVIEW CERTIFICATE

This is to certify that the work recorded in this project report entitled "**Blockchain Based E-Voting System using Ethereum Smart Contracts**" has been jointly carried out by Ms. Sneha Gupta (Reg. 201811502) and Mr. Hridayan Phukan (Reg. 201600535) of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering. This report has been duly reviewed by the undersigned and recommended for final submission for Major Project Viva Examination.

Dr. Ruhul Islam,

Associate Professor,

Computer Science & Engineering Department,

Sikkim Manipal Institute of Technology,

Majhitar, Sikkim - 737136

CERTIFICATE OF ACCEPTANCE

This is to certify that the below mentioned student of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology (SMIT) have worked under the supervision of Dr. Ruhul Islam of Computer Science Department of SMIT from 27 Feb, 2021 to 28 June, 2021 on the project entitled **"Blockchain Based E-Voting System using Ethereum Smart Contracts"**.

The project is hereby accepted by the Department of Computer Science & Engineering, SMIT in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering.

University Registration No	Name of Student(s)	Project Venue
201811502	Sneha Gupta	SMIT
201600535	Hridayan Phukan	SMIT

Dr. Kalpana Sharma
HOD & Professor
Computer Science & Engineering Department
Sikkim Manipal Institute of Technology
Majhitar, Sikkim - 737136

DECLARATION

I, hereby declare that the work recorded in this project report entitled **"Blockchain Based E-Voting System using Ethereum Smart Contracts"** in partial fulfillment for the requirements of award of B.Tech in Computer Science & Engineering from Sikkim Manipal Institute of Technology (A constituent college of Sikkim Manipal University) is a faithful and bonafide project work carried out at **SMIT** under the supervision and guidance of **Dr. Ruhul Islam** of Sikkim Manipal Institute of Technology, Sikkim.

The results of this investigation reported in this project have so far not been reported for any other Degree / Diploma or any other Technical Forum.

The assistance and help received during the course of the investigation have been duly acknowledged.

Sneha Gupta (Regd. No.: 201811502) _____

Hridayan Phukan (Regd. No.: 201600535) _____

ACKNOWLEDGEMENT

We take this opportunity to express our sincere thanks and deep gratitude to all whose cooperation, opinion, encouragement and guidance helped us in completing this project.

It was our immense pleasure to work under Dr. Kalpana Sharma, HOD and Professor, Computer Science and Engineering Department, and we wish to express our sincere gratitude for providing such platform where we could enhance our technical as well as professional skills.

We would like to thank Dr. Ruhul Islam, Associate Professor, Computer Science and Engineering Department for being supportive and patient with our approaches on course of completion of this project, as well as for the valuable time he invested into reviewing this project during the tenure of this project.

We also extend our sincerest appreciation and gratitude to all the other project co-ordinators, Dr. Biswaraj Sen, Mrs. Chitrapriya N, Mr. Santanu Kumar Mishra, and Mr. Sourav Paul for being extremely informative and providing us all the necessities that were required to carry out the project, as well as the panel members of Computer Science and Engineering Department, Sikkim Manipal Institute of Technology, for their valuable suggestions and precious time towards the project work.

Place: SMIT, Majhitar

Date: 28/06/2021

Sneha Gupta (Reg. No. 201811502)

Hridayan Phukan (Reg.No.201600265)

DOCUMENTED CONTROL SHEET

1.	Report No.	CSE/Major Project/In-House/B.Tech/InH21/2021
2.	Title of the Project	Blockchain Based E-Voting System using Ethereum Smart Contracts
3.	Type of Report	Technical
4.	Author(s)	Sneha Gupta (Regd. No.: 201811502) Hridayan Phukan (Regd. No.: 201600535)
5.	Organizing Unit	Sikkim Manipal Institute of Technology
6.	Language of the document	English
7.	Abstract	The project aims to use Smart Contracts of the Ethereum protocol to simulate a secure and decentralized voting system.
8.	Security Classification	General
9.	Distribution Statement	General

ABSTRACT

Building a secure electronic voting system that offers the fairness and privacy of current voting schemes, while providing the transparency and flexibility offered by electronic systems has been a challenge for a long time. In this work-in-progress paper, we evaluate an application of blockchain as a service to implement distributed electronic voting systems. The paper proposes a novel electronic voting system based on blockchain that addresses some of the limitations in existing systems and evaluates some of the popular blockchain frameworks for the purpose of constructing a blockchain-based e-voting system. In particular, we evaluate the potential of distributed ledger technologies through the description of a case study; namely, the process of an election, and the implementation of a blockchain-based application, which improves the security as well as the anonymity of hosting an election.

While developing this project, we keep in mind two of the most important questions that 'Can a blockchain be directly used to rapidly deploy meaningful and sufficiently performing trusted applications with added value over traditional approaches?' and 'Can blockchain be used to deploy non-trivial applications to achieve acceptable performance even with the currently present constraints?'. We hope to answer these questions with the help of this project.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Electronic voting systems have been the subject of active research for decades, with the goal to minimize the cost of running an election, while ensuring the election integrity by fulfilling the security, privacy and compliance requirements. Replacing the traditional pen and paper scheme with a new election system has the potential to limit fraud while making the voting process traceable and verifiable. Blockchain is a distributed, immutable, incontrovertible, public ledger.

This new technology has three main features:

a. Immutability: Any proposed "new block" to the ledger must reference the previous version of the ledger. This creates an immutable chain, which is where the blockchain gets its name from, and prevents tampering with the integrity of the previous entries.
b. Verifiability: The ledger is decentralized, replicated and distributed over multiple locations. This ensures high availability (by eliminating a single point of failure) and provides third-party verifiability as all nodes maintain the consensus version of the ledger.
c. Distributed Consensus: A distributed consensus protocol to determine who can append the next new transaction to the ledger. A majority of the network nodes must reach consensus before any new proposed block of entries becomes a permanent part of the ledger. These features are in part achieved through advanced cryptography, providing a security level greater than any previously known record-keeping system. Blockchain technology is therefore considered by many, including us, to have a substantial potential as a tool for implementing a new modern voting process.

This paper evaluates the use of blockchain as a service to implement an electronic voting (e-voting) system.

1.1 General Overview of the Problem:

Current voting systems like ballot box voting or electronic voting suffer from various security threats such as DDoS attacks, polling booth capturing, vote alteration and manipulation, malware attacks, etc., and also require huge amounts of paperwork, human resources, and time. This creates a sense of distrust among existing systems.

Some of the disadvantages are:

a. Long Queues.
b. Security Breaches.
c. Less eco-friendly and more time-consuming.

1.2 Literature Survey:

Sl. No.	Journal and Author Details	Project Relevance	Findings	Research Gap
1.	"Developing an Ethereum Blockchain Application", by Nikolaos P. Triantafyllidis, University of Amsterdam, System and Network Engineering 2016.	Details of the construction and use of Smart Contracts in an application.	This paper outlines the design and analysis of Ethereum smart contracts.	In this project, there is no control in place to check if a user is already participating or not.
2.	"Implementation of Smart Contracts in Web-based Electronic Voting Systems", by Faik Dzulfikar, Ajib Susanto, published in Journals Transformtika, July 2020.	Details of the voting scenario and design w.r.t. blockchain.	This paper discusses the details of Ethereum blockchain and its practical use in an application.	The scalability of Ethereum network is unknown, which affects the use of these contracts for nation-wide elections.
3.	"Towards Secure Applications using Ethereum Blockchain", by Ali Kaan Koc, et. al, Dokuz Eylul University Conference Paper, March 2018.	Details of the architecture of an existing blockchain system for secure applications.	This paper outlines the process of implementing smart contracts into web-based applications.	Reduce the use of supporting software and audit the system better.

Table 1: Literature Survey

1.3 Problem Definition:

In the traditional voting system, the process of voting is less secured and the cost of hosting an election is also high. Traditional E-voting system may face following problems:

a. Anonymous vote-casting: Each vote may or may not contain any choice per candidate, should be anonymous to everyone including the system administrators, after the vote is submitted through the system.
b. Individualized ballot processes: How a vote will be represented in the involving web applications or databases is still an open discussion. While a clear text message is the worst idea, a hashed token can be used to provide anonymity and integrity. Meanwhile, the vote should be non-reputable, which cannot be guaranteed by the token solution.
c. Ballot casting verifiability by (and only by) the voter: The voter should be able to see and verify his/her own vote, after he/she submitted the vote. This is important to achieve in order to prevent, or at least to notice, any potential malicious activity.
d. High initial setup costs: Though sustaining and maintaining online voting systems is much cheaper than traditional elections, initial deployments might be expensive, especially for businesses.

1.4 Analysis of the Problem:

Using blockchain, voting process can be made more secure, transparent, immutable, and reliable, as it stores everything as a transaction hash; and hence gives us a receipt of our vote (in a form of a transaction ID, a 64-character hash value) and we can use it to ensure that our vote has been cast securely.

Now, suppose a digital voting system (website/app) has been launched to digitize the process and all confidential data is stored on a single admin server/machine. The centralization of the said system makes it vulnerable to network attacks, such as DDoS attacks, which could make the system inaccessible to anyone. To avoid this, we propose the system to be integrated with the Blockchain technology, as opposed to the traditional server architecture, so that we can address all the security concerns of the traditional approach, and protect it from malicious tampering, as well as decentralize the system.

So, a gist of advantages is listed below:

a. The votes casted will be secure and unique to the users who can access the system.
b. We can vote anytime/anywhere (even during Pandemics like COVID-19 where it’s impossible to hold elections physically).

c. The confidential data of the users will be secured as well as immutable, reducing tamper risks and malicious intents.
d. Due to the decentralized nature of the application, the votes cast will be transparent and verified by the other peers.

1.5 Solution Strategy:

The blockchain technology may address many issues regarding e-voting schemes mentioned in above section and make e-voting cheaper, easier, and much more secure to implement. It is a considerably new paradigm that can help to form decentralized systems, which assure the data integrity, availability, and fault tolerance.

The implementation of the smart contract for the application will be done using Solidity, which is a high-level programming language maintained by the Ethereum community and is suggested as the main contract language. Solidity supports Object Oriented Programming and loosely resembles JavaScript. All the contracts needed in the project will be written in the online editor, Remix, before it is deployed into the Ethereum network.

The research methodology model that will be used in the development of the application is the Prototype Model. The Prototype Model is a software development method in which a prototype, an early approximation of a final system, is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete application can be developed. This model is best for scenarios where not all of the project requirements are known in detail ahead of time as it is an iterative and trial-error process.

Moreover, we aim to include the following requirements for our proposed application:

a. Authentication: Only registered voters will be allowed to vote.

b. Anonymity: The system prevents any interaction between the votes casted by the voters and their identities.

c. Accuracy: Votes once cast are permanently recorded and cannot be modified or changed under any circumstances.

d. Verifiability: The system will be verifiable such that the number of votes is accounted for.

1.6 Organization of the Report:

1.6.1 Introduction:

This section includes the overall view of the project i.e., the basic problem definition and general overview of the problem, which describes the project in layman terms. It also gives a brief about the techniques used for approaching the problem statement. Then is the survey, which includes the manual survey that is relevant to the project. This section also specifies the software used along with their versions and the proposed solution strategy i.e., the steps followed for the implementation of the project.

1.6.2 Design Strategy of the Solution:

This section consists of detailed diagrams that shows the architectural flow of process that handles the back end of the application and creation of the front end and its interaction with the back end. The diagrams included are the Object-Oriented Designs, i.e., the block diagram, use case diagram, flowchart and the sequence model diagram.

1.6.3 Detailed Test Plan:

This section consists of the testing criteria used to test the application. It consists of a table that contains number of test cases and its result details. The table consist of four parts. First column contains the module names on which the test cases are prepared. Second column describes the testing purpose. The last two columns are the details of the expected outcome and the actual result.

1.6.4 Implementation Details:

This section includes the explanation of the structure of the project and the pseudo-code for developing the different modules of both front end and the back end.

1.6.5 Results and Discussion:

This section discusses the results of the entire implementation and contains screenshots of the application.

1.6.6 Summary and Conclusion:

This section contains of three parts. The first part consists of the summary of achievements that describes the implementation done successfully. Then, the second part consists of the limitations of the project, which describe the constraints that were not met by the project. The third part, consisting of the scope of future work, describes what future changes can incorporated in the project to enhance its functionality.

1.6.7 Gantt Chart:

It illustrates the project schedule and the activities performed in time frame. The chart lists the activities to be performed on vertical axis and the time intervals on horizontal axis.

1.6.8 References:

This section states the research papers and the sites referred for this project in its study and implementation.

2. DESIGN STRATEGY FOR THE SOLUTION

2.1 Block Diagram of the application:

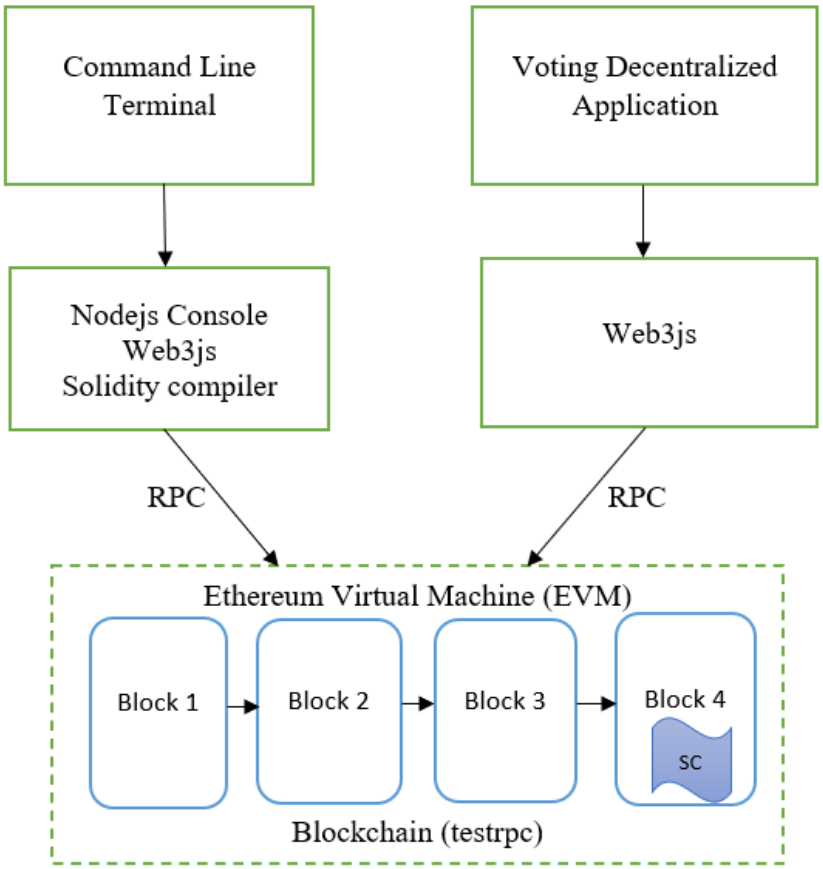


Fig 1: Block Diagram

2.2 Flow Chart of the Smart Contract:

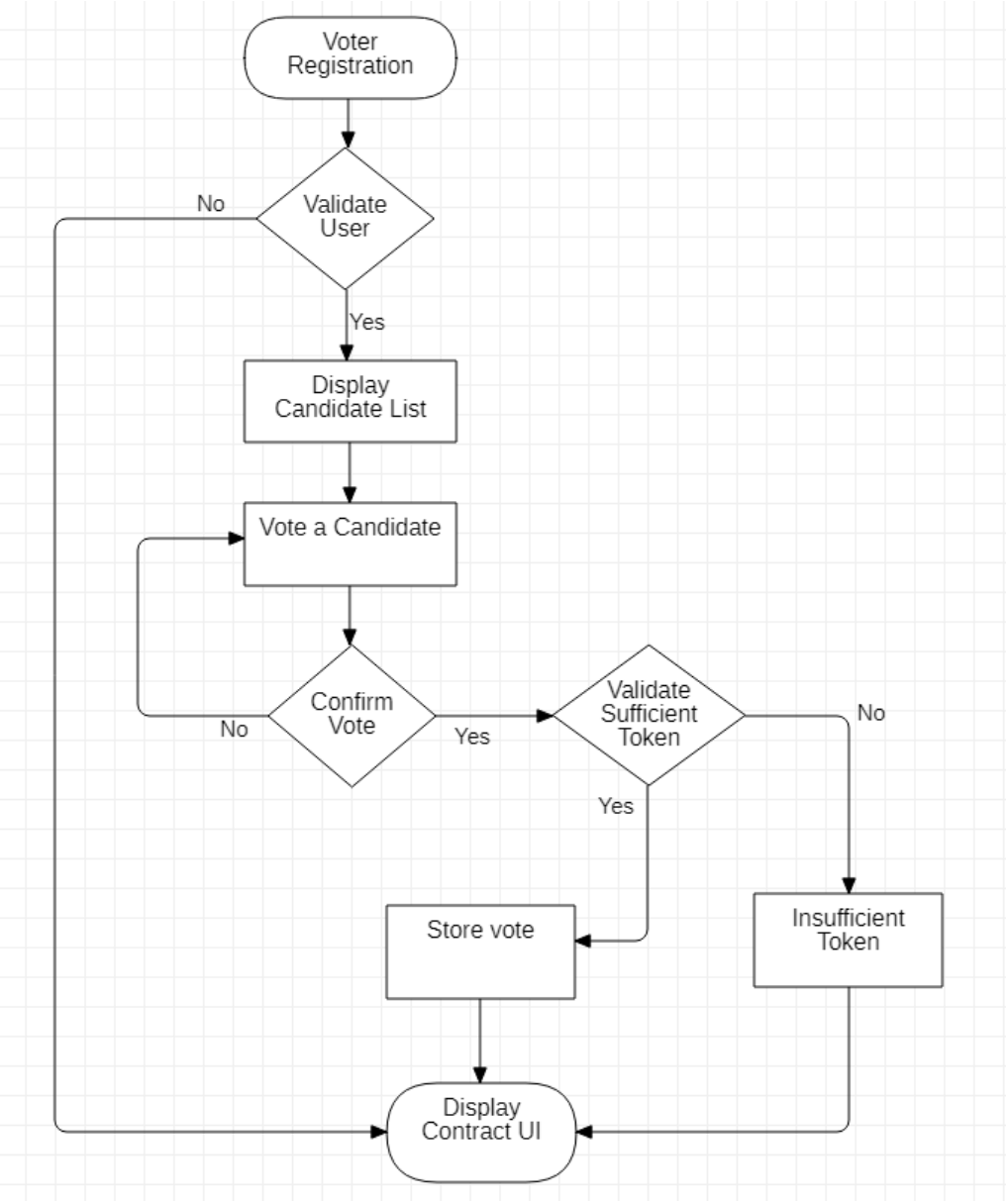


Fig 2: Flow Chart

2.3 Use case diagram of the application:

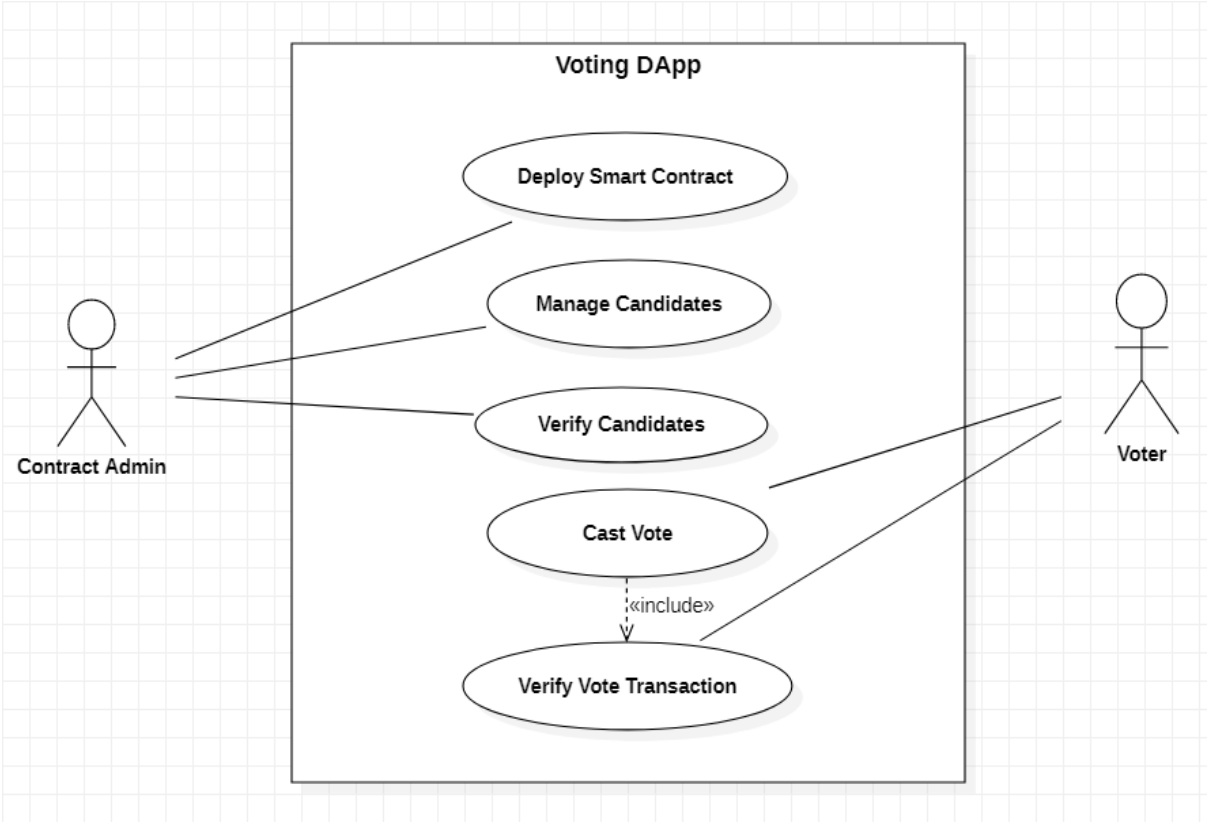


Fig 3: Use Case Diagram

2.4 Class Diagram of the Application:

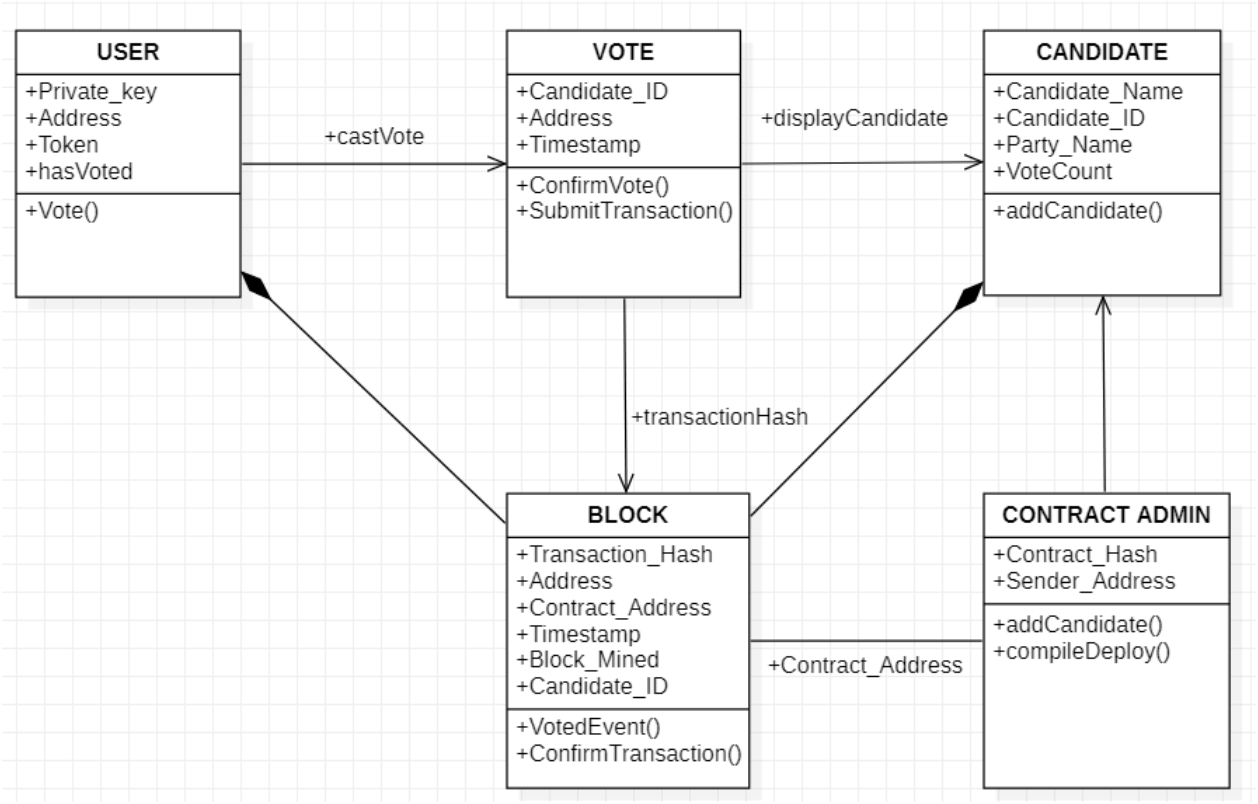


Fig 4: Class Diagram

2.5 Sequence Diagram of the Contract:

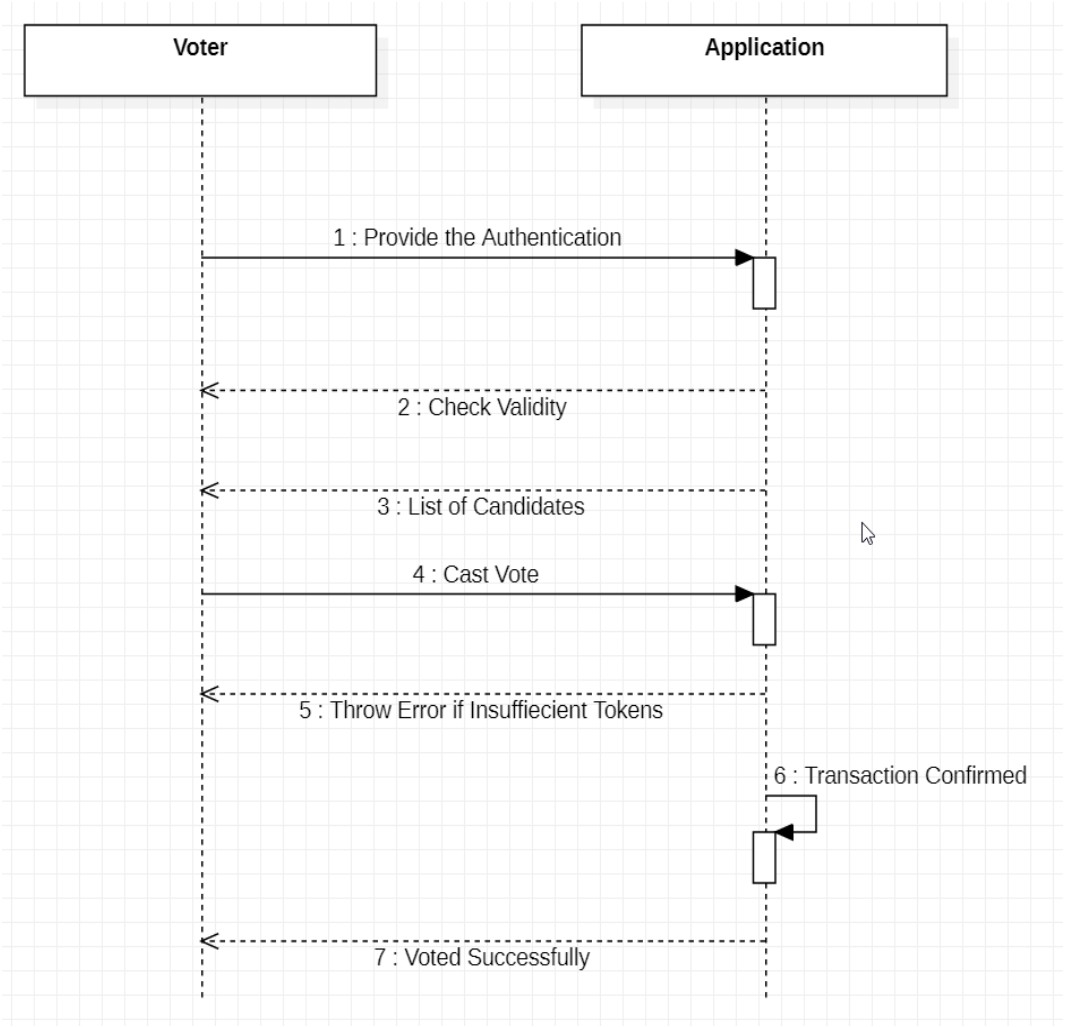


Fig 5: Sequence Diagram

3. DETAILED TEST PLAN

Detailed Test Plan can be considered as a document containing certain plans for all the testing activities to be done in order to achieve a quality product. A test plan document can be derived from an SRS document or any use case document. It is generally prepared by the testing integrators who decide what to test, what not to test, how to test, when to test and who will do what test. A test plan is generally kept up to date in order to achieve a successful testing of the application.

The overall purpose of testing is to ensure that the Smart Contract meets all of its functional and business requirements. We test the contract using Mocha framework which is a testing library that enables us to run tests serially, allowing flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. We write unit tests around the contract, say if it initializes with the correct number of candidates, or if a voter has voted before or not, and repeat the test cycle until the unit is free from bugs, if any. If a test fails because of an error, then that error is reported, and a new version of the Contract is expected that has had the error fixed.

The main test cases of the Smart Contract are mentioned in the table below:

SL. NO.	**Test Case**	**Test Purpose**	**Expected Outcome**	**Actual Result**
1.	Initialize with correct candidate values.	To ensure the contract consists of correct Candidates.	It initializes the candidates with correct values	Test Pass.
2.	Allow a voter to cast vote.	To ensure the contract enables the voting functionality.	It allows a user to cast a vote	Test Pass.
3.	Throws an exception for invalid candidates.	To ensure the contract instantiates with the specified number of candidates.	It throws an exception for invalid candidates.	Test Pass.
4.	Throws an exception for double voting.	To ensure voter casts only one vote.	It throws an exception for double voting	Test Pass.

Table 2: Test Plan of the Smart Contract

4. IMPLEMENTATION DETAILS

We have designed the Smart Contract which simulates a voting process, using the Truffle development environment. We also tested the contract using unit testing procedures and deployed it in our local blockchain, which is managed by the Ganache server. The Smart Contract consists a list of arbitrary candidates, and a vote method which carries out the vote function of the contract. There are also other aspects like modifiers to enable only one account to vote at a time and check if an account has voted before. This contract will serve as the backend of our application which in turn will enable voters to interact with the application and cast their vote.

The pseudo-code of the contract is as given below:

a. Pseudo-code for addCandidate function:

function addCandidate (string name, string party) private {

1. Increment candidates count by 1.
2. Put current candidate in the Candidate struct along with its count.

}

b. Pseudo-code for Vote function:

function vote (candidateId) public {

1. Require that candidate has not voted before.
2. Require that candidate is valid.
3. Record Voter has voted
4. Update candidate vote count.
5. Trigger voted event.

}

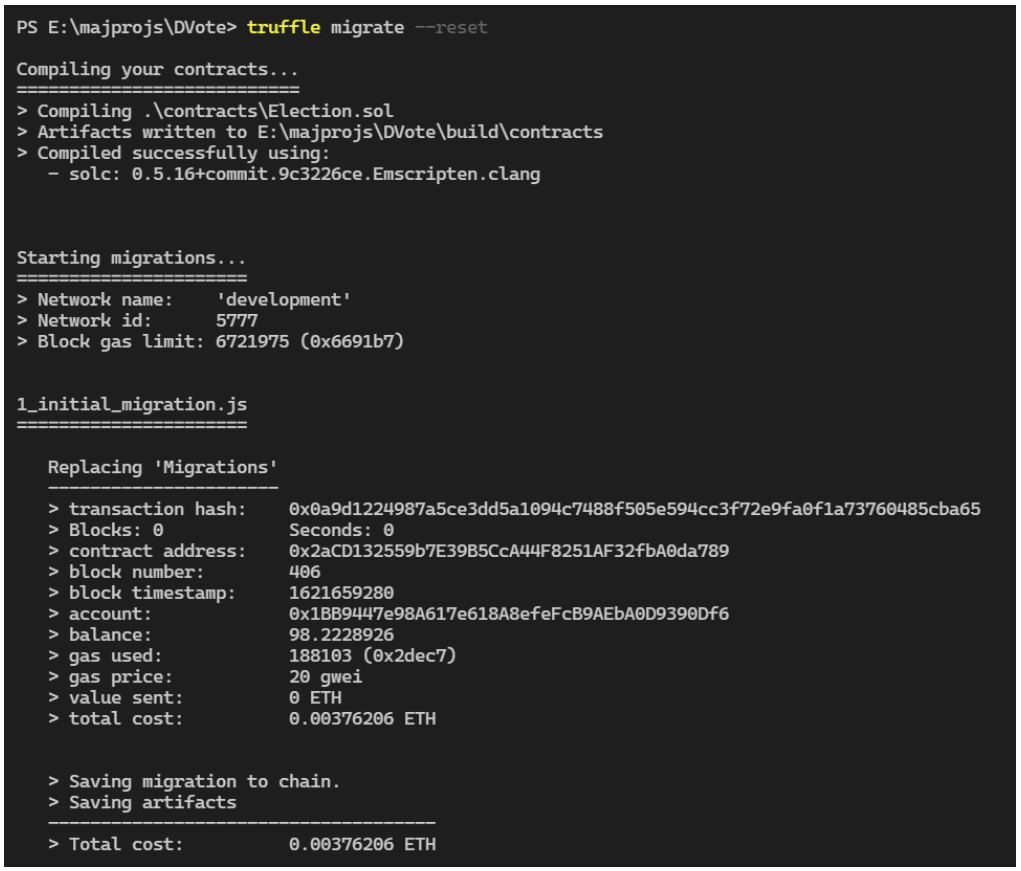
c. Pseudo-code for Candidate struct:

```
struct Candidate {
        uint id;
        string name, party;
        uint voteCount;
}
```

We, then, compile the Smart Contract using the solidity compiler. This compiler program is in-built in the Truffle Suit framework, which can be accessed using ‘truffle’ commands. Once compiled and deployed, the Smart Contract is mined in the local blockchain, which is served by Ganache Blockchain and the Contract gets stored in a block. This phase is known as the Contract Creation phase.

When the contract is deployed into the blockchain, our application can access the contract using web3js. web3.js is a collection of libraries that allow us to interact with a local or remote Ethereum node using HTTP, IPC or WebSocket. The web3 JavaScript library interacts with the Ethereum blockchain, and retrieves user accounts, send transactions, interact with smart contracts, and more.

Below are some snapshots of the Contract Creation phase, the user accounts provided by Ganache, the blockchain itself as well as the test report of our Smart Contract:



(i)

```
2_deploy_contracts.js
=====================

   Replacing 'Election'
   --------------------
   > transaction hash:    0xb1e1b7c7344ce703d4bee7c193b0552c491cc4ef4f85772ea6955eda70574601
   > Blocks: 0            Seconds: 0
   > contract address:    0xD235aF3bADAe3864FFB5B819373Ad41b754E29CF
   > block number:        408
   > block timestamp:     1621659284
   > account:             0x1BB9447e98A617e618A8efeFcB9AEbA0D9390Df6
   > balance:             98.20686924
   > gas used:            759678 (0xb977e)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.01519356 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.01519356 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.01895562 ETH
```

(ii)

Fig 6: Contract Creation

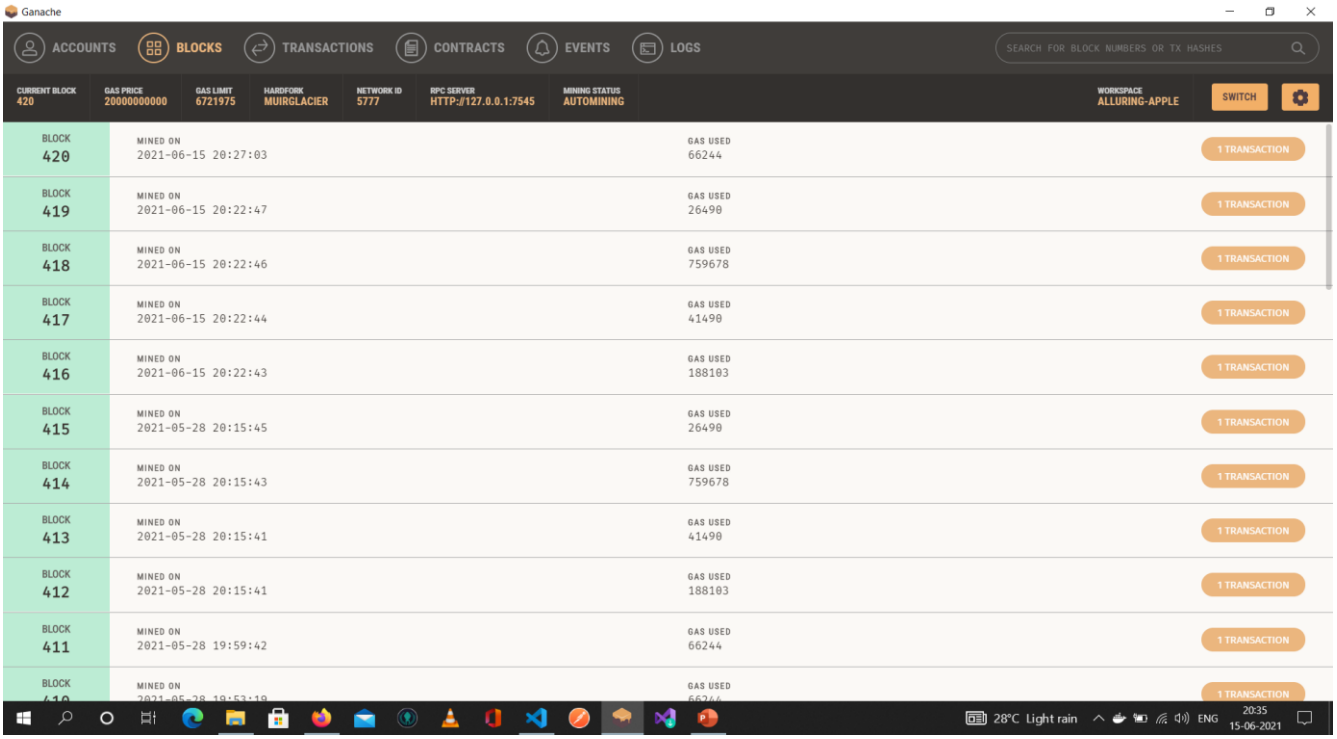


Fig 7: Blocks mined during the whole tenure of the project

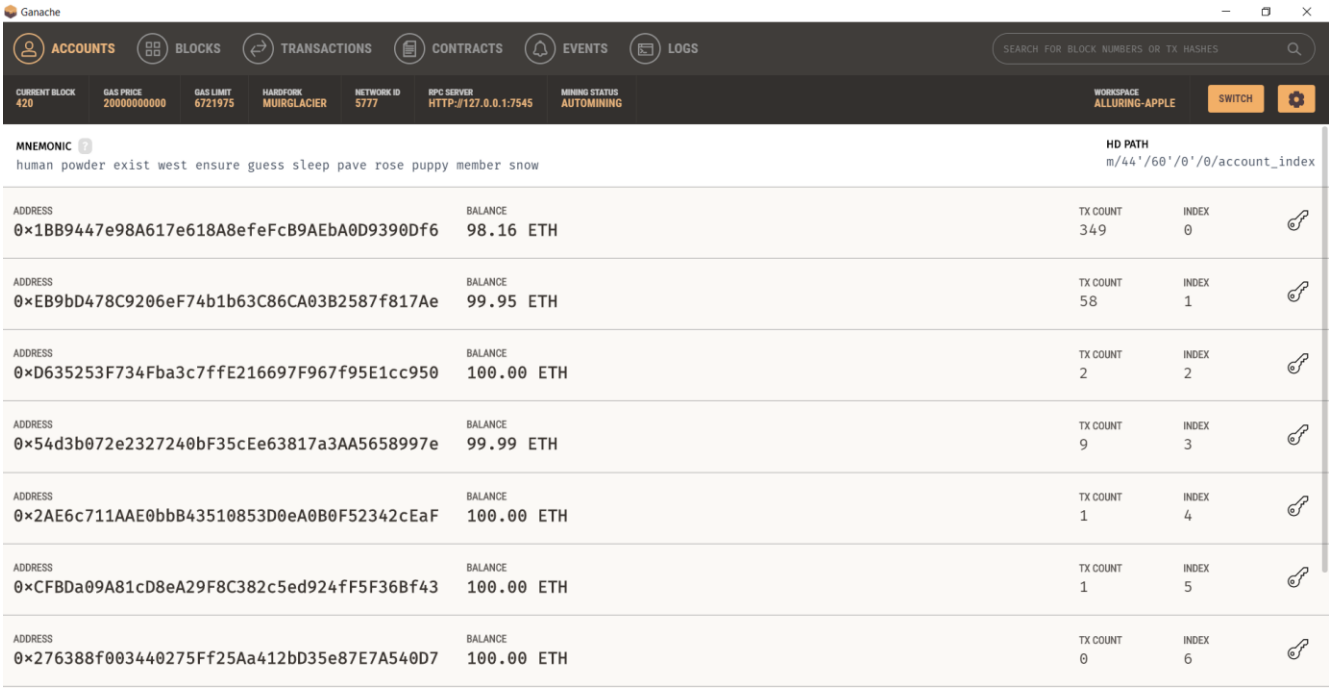


Fig 8: Contract Owner Accounts

```
PS E:\majprojs\DVote> truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Artifacts written to C:\Users\hrida\AppData\Local\Temp\test--12120-eNhyVTierBhd
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



  Contract: Election
    ✓ initializes with five candidates along with the parties (211ms)
    ✓ it initializes the candidates with the correct values (1111ms)
    ✓ allows a voter to cast a vote (822ms)
    ✓ throws an exception for invalid candiates (956ms)
    ✓ throws an exception for double voting (2288ms)


  5 passing (6s)
```

Fig 9: Test Report of the Smart Contract

5. RESULTS AND DISCUSSION

The application developed requires the data for each user who will vote, wherein the data needed is only limited to the private key which is used as a unique key for each user account. This data is available from Ganache as pre-existing contract owner accounts. The data for each user is as below:

Sl. No.	User Account	Private Key
1.	Account 1	889c020f8232f187119f9dbc0ce8366daed36b7a64e29a092043a29104d71afc
2.	Account 2	4a0b9d698780c87e5d3a30e55d93784eb44f46b38f2b9e37f372c3bf6dfff2be
3.	Account 3	fea594efde3ac096468310ed084cbfd68b9dd7013b4f699ce6cba5f05b2812ee
4.	Account 4	a50e64a12f3208d30e1011727f3477e0ee3986be53ea51ca0f6f3993a93e7f0f
5.	Account 5	b6fcb347a791f924ef6abfc3710207012ca735a03ec7934fd7ee7f258bbf0961

Table 3: User Accounts

To begin the voting process, users have to prepare the following:

a. Prepare an Internet Connection and make sure that the supporting programs such are Ganache, Truffle and Metamask are running successfully. Metamask is the cryptocurrency wallet extension, available as a browser extension, which can be used to interact with the blockchain as well as the Smart Contract.
b. Then, confirm whether there is already an active account, and if not, then create a new account with the help of private keys mentioned above.
c. After this, once the account is active in Metamask, voters can start voting by selecting the candidates.

Users can interact with the application in the following stages:

1. Upon loading the application, users are greeted with a landing page as shown below.

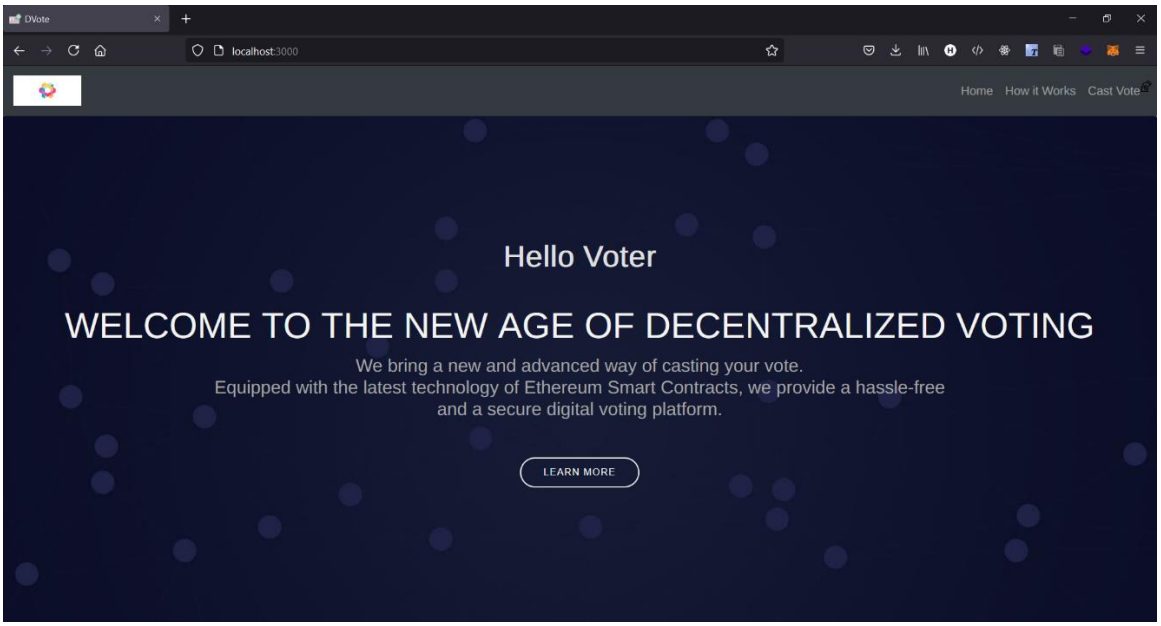


Fig 10: Landing Page

2. User, then, has to check if their Ethereum Account is connected to the application through Metamask. If connected, the user account shall look like as shown below.

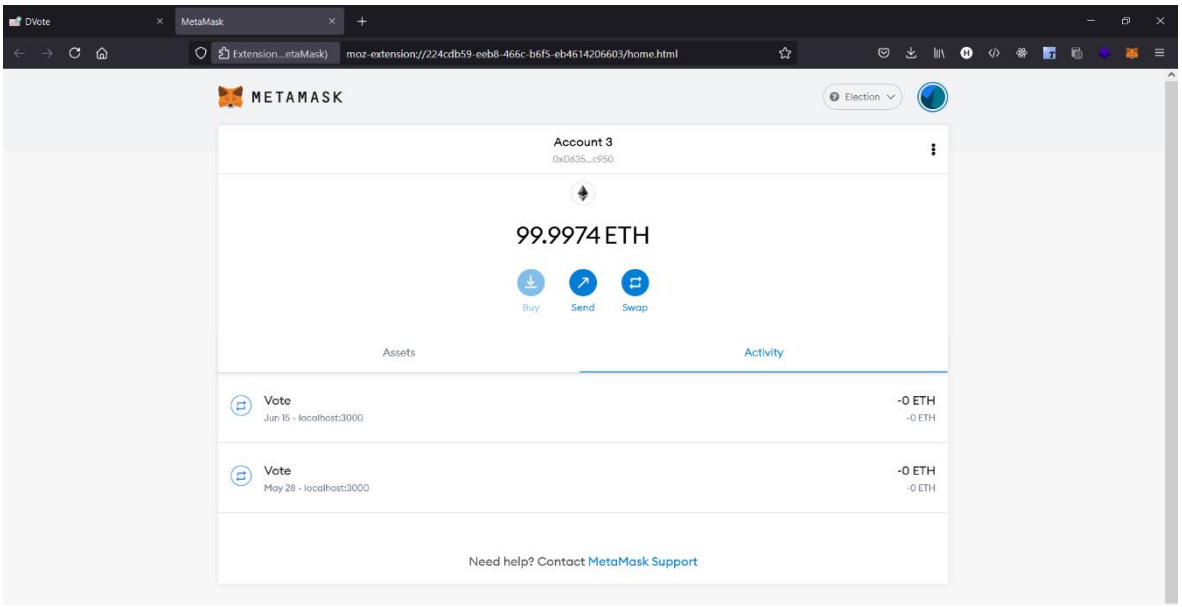


Fig 11: Account Details Page

3. If the account is not authenticated through Metamask, then the Candidate List won't appear, until the correct account is connected.
4. After selecting the correct account, the user can vote directly by selecting the candidate from the Candidate List displayed, and then clicking on the 'Vote' button.
5. After the 'Vote' button is pressed, a new page will appear, as shown below, to confirm that the voter account will send a Gas Fee as a condition for voting. This transaction is a proof that the voting has been done and entered into the blockchain, as well as ensure that the same account cannot vote twice.

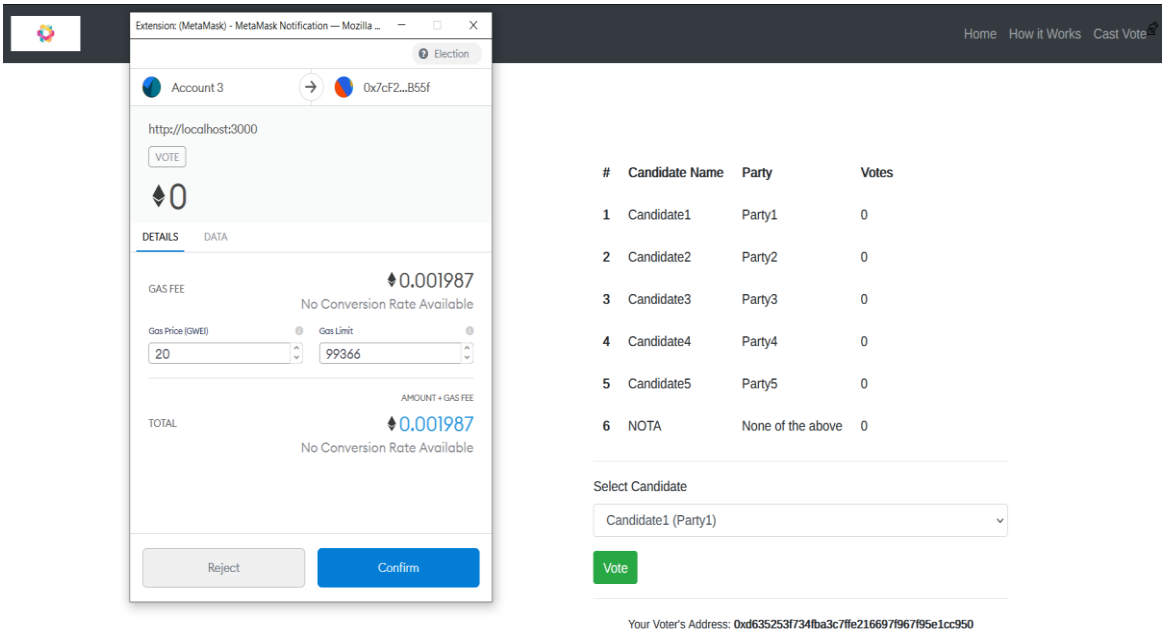


Fig 12: Vote Page

6. Then, after confirming the transaction from the Ethereum account, the vote is sent and immediately reloads the Vote page to display the cast vote in real-time, as shown below.

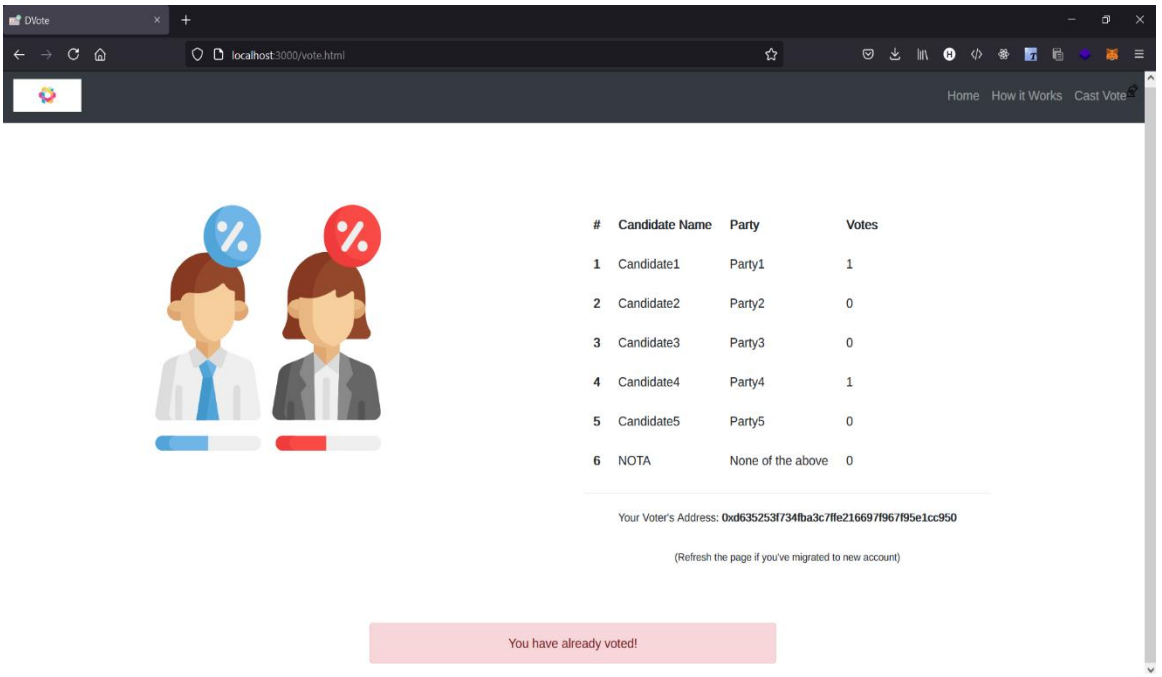


Fig 13: Voted Event

7. After the Vote is cast, users cannot vote again. This is one of the core functionalities of the Smart Contract, which prevents double voting scenarios.
8. The transaction details of this Vote event can be visited through the block mined on which this transaction is emitted through Ganache, as shown below. The block will contain all the necessary details such as the Contract Name, the Contract Address, the Transaction Hash, Block Time and the Event emitted when this block is mined.

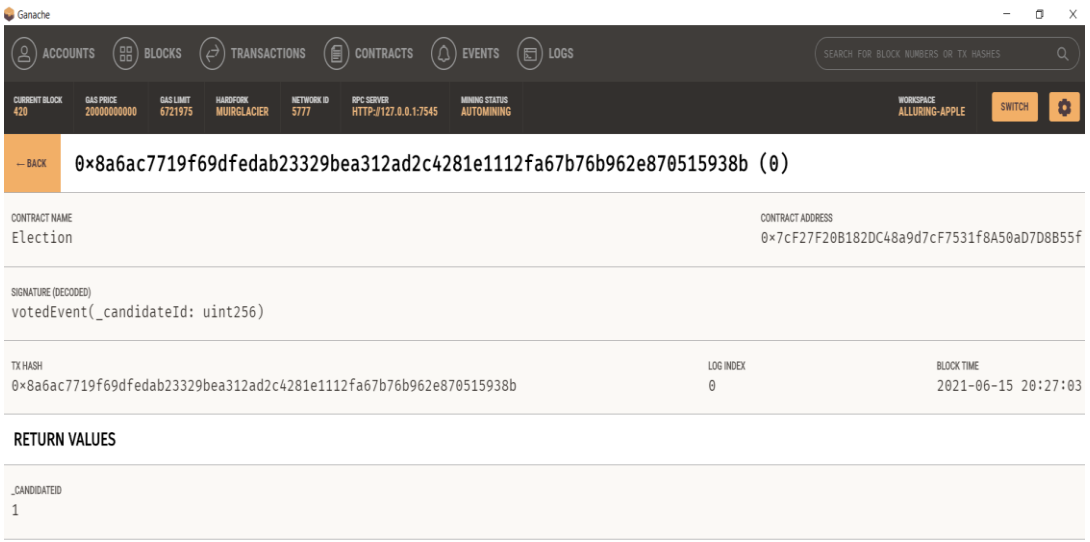


Fig 14: Transaction Details through Ganache.

9. The user can check the transaction details after the vote has been cast, via Metamask, as shown below.

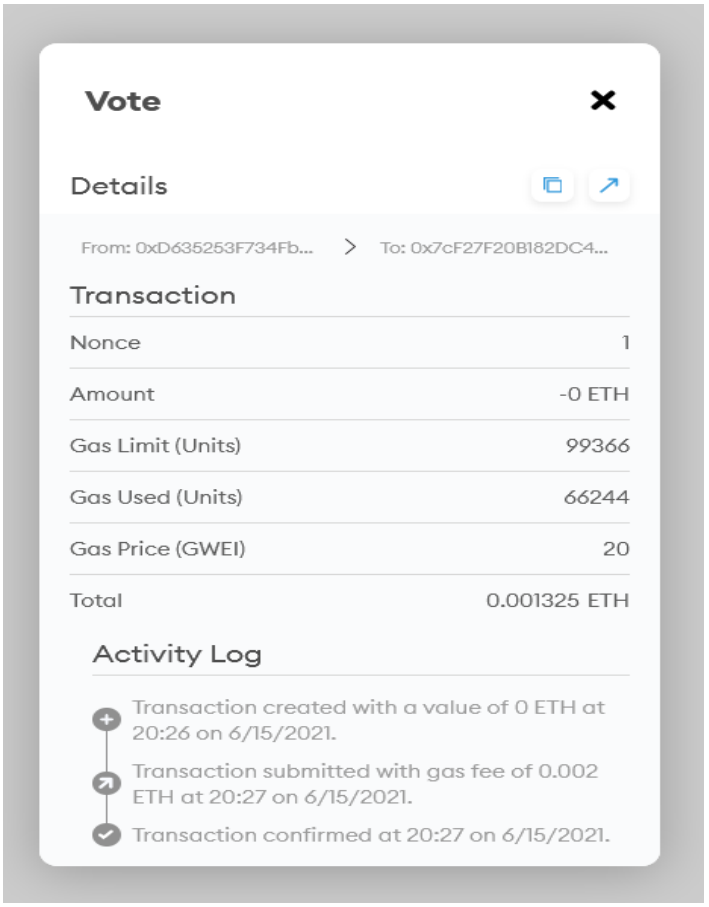


Fig 15: Transaction Details through Metamask.

10. Another user can access the application from another Ethereum account, and a fresh list of candidates will be displayed along with any vote cast before.
11. They can vote in the same way as many times as required, provided no two accounts try to vote twice, which is restricted in the Smart Contract.

6. SUMMARY AND CONCLUSION

6.1 Summary of Achievements:

By building this proposed smart contract of ours, we have succeeded in moving e-voting to the blockchain platform and we addressed some of the fundamental issues that legacy e-voting applications have, by using the power of the Ethereum network and the blockchain structure. As a result of our project, the concept of blockchain and the security methodology which it uses can become adaptable to the concept of polls and electronic voting applications. This achievement may even pave the way for other blockchain applications that have impact on every aspect of human life.

So, from this project, we can conclude the following observations:

a. The proposed project, based on Ethereum Blockchain, work well and can validate the voter's identity well as well as prevent repeating of the voting process.
b. The proposed project can store data safely and reliably on the blockchain.
c. The proposed project makes the voting process much faster and safer, with the inclusion of blockchain technology powered by the Ethereum Protocol.

Lastly, to answer the research questions we posed in our abstract above, we can surmise that in general, a fully decentralized application can be developed and launched with minimal efforts with the help of the Ethereum Protocol and its Smart Contracts, as well as use it to develop a non-trivial application, such as our proposed project of Blockchain Based E-Voting System using Ethereum Smart Contracts, wherein traditional approaches can be too complex as well as very difficult to implement.

6.2 Limitation of the Project:

The limitations of this application are as:

a. The application depends on Metamask, a cryptocurrency wallet which is used to interact with any Ethereum application, as the sole authenticator for a user. This in turn provides a security on the account level but not on a personal level.
b. The application requires that a user has the certain amount of Ether tokens required to cast a vote or access the Contract.
c. The application requires an active instance of the blockchain running so as to receive the data of the Contract. This, though, can be eliminated if we can migrate the Contract onto the main Ethereum network or store it in the IPFS service.

d. The scope of our project is limited for small-scale polls and voting processes. Since, the Ethereum network's scalability is unknown and needs further research, we cannot suggest the use of these contracts for a large-scale poll with tens of lakhs of voters, at least for now.

6.3 Future Scope:

This application is very flexible so that the maintenance and further amendments based on the changing environment and requirements can be made easily. It can be further developed to include more operations and analysis, as changes are required in the application to adapt to the external developments. Further enhancements can be made to the application at any later point in time. The versions of the framework may not affect the flow of the application as the application may automatically upgrade or update with the upgrade or update of the versions of the framework.

Moreover, since we only used a subset of all the features offered by Ethereum, one could look much deeper into the implementation of the project, and use all the features and tools available, such as real-time token generation through ERC20 standards, to leverage even more power of Ethereum. Also, a proper authentication system for the users, be it biometric or otherwise, can be setup to enhance the security of the application as well as the Smart Contract on a personal level.

7. FINAL USER'S MANUAL

This project is powered by the Ethereum Protocol and its software armaments. For a user to successfully run this application, they have to have access to the following libraries and frameworks available online:

a. **Truffle Suite**: This is a development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), which makes the development of decentralised applications easier.

b. **Ganache:** This is a personal blockchain for Ethereum development that can be used to deploy contracts, develop your applications, and run tests. It is available as both a desktop application as well as a command-line tool (formerly known as the TestRPC).

c. **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.

After the aforementioned frameworks are available to the user, they can access the source code of the application from this GitHub Link and execute the following steps to run the code:

1. Run the 'npm install' command to install all the dependencies required for the project.
2. Run the 'truffle migrate' to compile and deploy the Smart Contract into the Ganache Blockchain.
3. Run the 'truffle test' command to check if the Contract passes the tests specified, be it old or new.
4. Run the 'npm run dev' command to execute the application into the browser.
5. Make sure the application is connected to Metamask as well as an Ethereum Account before casting a vote.

8. GANTT CHART

Activity	Time Frame (for the year 2021)			
Tasks Performed	1 March, 2021-31 March, 2021	1 April, 2021-30 April, 2021	1 May, 2021-31 May, 2021	1 June, 2021-30 June, 2021
Literature Survey				
Feasibility Study				
Problem Identification				
SRS and Design				
Implementation				
Analysis				
Documentation				

	Proposed Activity
	Activity Achieved
	On-going Activity

Fig 16: Gantt Chart

9. REFERENCES:

[1] Ahmed Ben Ayed, *"A Conceptual Secure Blockchain-Based Electronic Voting System"*, International Journal of Network Security & Its Applications (IJNSA) Vol. 9, No. 3, May 2017.

[2] Chris Dannen, *"Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners"*, published by Apress Media LLC.

[3] Kevin Solorio, Randall Kanna and David H. Hoover, *"Hands-On Smart Contract Development with Solidity and Ethereum"*, published by O'Reilly Media Inc.

[4] Michele Marchesi, Lodovica Marchesi and Roberto Tonelli, *"An Agile Software Engineering Method to Design Blockchain Applications",* Software Engineering Conference Russia (SERC 2018).

[5] Ritesh Modi, *"Solidity Programming Essentials: A beginner's guide to build smart contracts for Ethereum and blockchain"*, Packt Publishing, 2018.