Design and Development of a Decentralized Voting System Using Blockchain

A project report submitted to the department of Computer Science and Engineering of the World University of Bangladesh in partial fulfillment of the requirement for award of the degree of Bachelor of Science in Computer Science & Engineering

Submitted By:

Niamul Haque

ID: WUB 03/18/42/2672

Md. Toriqul Islam

ID: WUB 03/18/42/2677

Supervised By:

Ayesha Siddika

Assistant Professor

Department of Computer Science and Engineering

World University of Bangladesh

Department of Computer Science and Engineering

June, 2022

LETTER OF TRANSMITTAL

June, 2022

To

Ayesha Siddika

Assistant Professor

Department of Computer Science & Engineering

World University of Bangladesh (WUB)

Plot: 5-8, Avenue 6 & Lake Drive Road Sector:17/H, Uttara

Dhaka-1230 Bangladesh.

Subject: Submission of Project Report.

Dear Sir,

We are pleased to submit the report entitled **“Design and Development of a Decentralized Voting System Using Blockchain”.** It was a great pleasure to work on such an important topic. The report is prepared according to the requirements and guidelines of the Department of Computer Science and Engineering. World University of Bangladesh (WUB).

We believe that the report will help you in evaluating our project work. It would be a great pleasure for us to interpret any part of whole of the report whenever necessary.

Sincerely Yours

Niamul Haque

WUB 03/18/42/2672

Sincerely Yours

Md. Toriqul Islam

WUB 03/18/42/2677

World University of Bangladesh

DECLARATION

We hereby solemnly declare that the project work entitled **“Design and Development of a Decentralized Voting System Using Blockchain”,** has been supervised by **Ayesha Siddika,** Assistant Professor of the department of Computer Science & Engineering. World University of Bangladesh. We ensure that the project report has not been submitted either in whole or part for any degree or Diploma in my university previously.

We hereby warrant that the work we have presented does not breach any existing copyright rule.

We further undertake to indemnify the university against any loss of damage arising from breach of the forgoing obligation.

Sincerely Yours

____

Niamul Haque

WUB 03/18/42/2672

Sincerely Yours

____

Md. Toriqul Islam

WUB 03/18/42/2677

Department of Computer Science and Engineering

World University of Bangladesh

CERTIFICATE

I hereby certify that the Project Report on **“Design and Development of a Decentralized Voting System Using Blockchain”,** is a confide record of project work done by **Niamul Haque and Md. Toriqul Islam** for partial fulfillment of the requirements for award of the degree of the bachelor of Science in Computer Science and Engineering from World University of Bangladesh.

The project report has been carried out under my guidance and is a record of the bona-fide work carried out successfully by the students.

Supersisor

………………..

Ayesha Siddika

Assistant Professor

Department of Computer Science and Engineering

World University of Bangladesh (WUB)

ACKNOWLEDGEMENTS

At first, we express our gratefulness to almighty God for His divine blessing makes us possible to complete the project successfully.

We privileged to express our sense of gratitude to our respected Supervisor, **Ayesha Siddika, Assistant Professor, Department of Computer Science and Engineering, World University of Bangladesh.** Her unparalleled knowledge and judgment along with intuition was an immense support in completing the project.

We also get our gratitude salute of the Design and Development of a Decentralized Voting System using blockchain project team members. They are the pioneers of this project and they always guide us about the project completion, so we are very thankful for their cordial support.

Next, we would like to express deep gratitude to all the friends that have accompanied us through this journey.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

Sincerely Yours	Sincerely Yours
________________	________________
Niamul Haque	Md. Toriqul Islam
WUB 03/18/42/2672	WUB 03/18/42/2677

ABSTRACT

E-voting reduced the cost of election and provided convenience to some extent as compared to the traditional approach of pen and paper but it was considered to be unreliable as anyone having access to the machine physically can obstruct the machine and alter the votes. Also, in order to control the entire procedure from electronic voting to electoral results and tracking the outcomes, a central system is required. Voters are not completely secure as vote can be targeted easily. It also possesses a great threat to the right to vote and transparency. For long, different e-voting systems have been provided with the goal of increasing security and minimizing cost. With the launch of Ethereum, a decentralized platform which runs decentralized applications on it, a secured voting system now seems possible.

Making a project successful requires hard work and study. For study purposes, some previous works have been studied and gathered a lot of information which makes the project possible. The job portal-based application and some other things have been learned from the study.

The methodology is a very important process for a project. Without this process, a project can't build up properly. It is a blueprint of a project. This project has used a static waterfall methodology which has a feedback path from one phase to its existing phase.

It will allow accounts connected to the network to vote in the election and all votes come in the real time. Application's logical code will deploy in smart contract to a local Ethereum blockchain. It will have a test-driven development (TDD) part, means tests against the smart contract to ensure that the code is bug free. It will also have a client-side application that will allow users connected to the network to interact with the smart contract and vote in the election. User-Interface will develop using HTML, CSS, JavaScript.

This paper provides a solution for removing inconveniences from conventional elections using blockchain that has emerged as an exciting technology for various application due to its unique characteristics that outperform other technologies. The goal of this research is to establish a system for e-voting that is decentralized rather than centralized by using blockchain technology that guarantees protection to electorate's identity, data transfer privacy and variability by an open and transparent voting process.

Table of Contents Page

List of Figures:

CHAPTER 1: INTRODUCTION

1.1 Introduction

In a centralized system, A web Application depends on a central server. All the code to that web application lives on this central server, and all the data lives in a centralized database. That means the organizing authority can access and change code, data, and the system's rules at any time.

The blockchain is a peer-to-peer network of nodes. Each of the nodes will get a copy of all the data that is shared across the blockchain. All this data is connected in the bundles of records called blocks, chained together to create the public ledger. All the nodes on the network work together to ensure that all the data on the general legend remain secure and unchanged. On the blockchain, all the data doesn't lie on a central server; instead, the data is decentralized. It's distributed across every node connected to the network. The device connected to the blockchain is a node.

Blockchain technology is decentralized, distributed, immutable, irreversible and provides data security. These technological features operate through advanced cryptography, providing a security level more significant than any previously known database. The ledger is shared through every node connected to the network. That means data and logic are transmitted through the connected node. So, Blockchain can be referred to as a decentralized public database with replicates distributed over several nodes simultaneously. In Blockchain, there is no authority in charge of managing and maintaining the ledger of transactions. Blockchain technology allows a secure validation of a transaction's data integrity.

This project provides a solution for removing inconveniences from conventional elections using blockchain. That has emerged as an exciting technology for various applications due to its unique characteristics that outperform other technologies. This project aims to establish a decentralized system for e-voting rather than a centralized one by using blockchain technology that guarantees the protection of the electorate's identity, data transfer privacy, and variability through an open and transparent voting process.

1.2 Objectives

1. To study the security challenges associated with electronic voting systems and identify requirements.
2. To design and develop an electronic voting system
    - Implement a Ethereum smart contracts and decentralized apps underlying the particular case study in the scope of this project.

1.3 Justification of Study

Voting is a democratic practice of a group to share their opinions, decide, and express their choice against debates and controversies like ballot questions, candidate elections, political parties, etc. The traditional voting system is based on pen and paper, which fails in some security concerns. It isn't traceable and verifiable the whole voting process. The results of voting events have always been questionable and not trustworthy to voters. The electronic voting system is a new election system that has become popular. Voting online pull's in different controversies also gained popularity. But the organizing authority on anyone with physical access to such a system or server can hack or alter the votes, thereby affecting all votes cast on this system.

So, the election's fundamental rules get broken, and voters' freedoms and human rights are violated. The problem with the current ballot voting system is being taken advantage of by people or organizations looking to gain power. The current ballot system does offer anonymity to the voter, but the counting process is not transparent. People are supposed to trust the result provided by an Election commission or a government body. This makes the process of counting a significant vulnerability in the current process. There are also major electoral scams such as voter fraud, ballot stuffing and booth capturing. All these make it very difficult for organizers of an election to distinguish between the actual votes and votes added without authorization.

To solve the problem, we use Blockchain technology, which is decentralized, distributed, immutable, irreversible, and provide data security. We will develop a decentralized voting platform based on the Ethereum blockchain. It allows accounts connected to the network to vote in the election, and all votes come in real-time.

1.4 Scope of Study

Electoral systems are comprised of a set of rules that determine how elections should be conducted and how their results are decided. These rules encapsulate all aspects of the voting process, such as when an election is to take place, who can vote, who can stand as a candidate, how the ballots are to be labeled and cast, and how the votes are to be counted, among others. To illustrate an example, in a traditional voting system, a person of age would attend their parish or consular district, provide their identification card to a polling agent and receive a ballot paper with a checkbox list of the possible candidates to vote for. The person in question proceeds to an isolated area, where they may select a single candidate or none from the list with a checkmark. The ballot is then delivered to the ballot box and counted by a teller once the deadline for voting finishes. After all the votes are counted, the results are calculated and made available to the public by the government by the many tellers across the country. An electronic voting system that mimics real-world scenarios has long been desired. However, up until this point, it had not been possible to fully address all the mandatory properties of a real-world voting scheme simultaneously. These properties are, among others, security, anonymity, coercion resistance and non-traceable. Recently, with the onset of new technologies and research, however, it is not only possible to fulfill these same properties but also to improve the anonymity and convenience of voting. One embodiment of such technology is the Ethereum Blockchain.

It possesses the following properties:

- It is practically immutable, as altering information would require copious amounts of computing power;
- It is transparent, as anyone can view its data;
- It is decentralized, as there is no central authority behind it;
- It is used to build smart contracts and Dapps.

CHAPTER 2: LITERATURE REVIEW

2.1 Review of Relevant Literature

In their study discussed a number of challenges in e-voting system including legal challenges, social and cultural challenges, technical challenges, attacks, etc (E. Elewa, A. AlSammak, A. AbdElRahman, T. ElShishtawy, 2015)

In their work identified an experiment to test the validity of election results and to enhance transparency and voter participation within electronic elections. This proposal was based upon two aspects i.e. distributed collection of pole tape, made by mobile devices by voters and crowdsourcing election data verification by electoral authority (Aranha DF, Ribeiro H, Paraense ALO, 2016)

If voters make use of voting protocol correctly then there will be no chance of attack on results of elections and they give a statistical method to improve the security of e-voting (Gjøsteen K, Lund AS, 2016)

Paper explores how to properly develop voting machine interfaces to promote the role of electors and electoral administrators and then use such interfaces for complex elections. The problem of relying on a remote voting device is said to be firmly linked to the interface provided which in turn influences votes as verification of voting is an important issue. Much of the work in remote–electronic voting involves cryptography voting protocols design and verification to safeguard desired property (Budurushi J, Renaud K, Volkamer M, Woide M, 2016)

In order to make specific recommendations on the type of voting system that is best suited to that particular context, they introduced a model for the comparison of voting schemes in any given electoral setting and the model was applied to the specific context of Estonian internet voting. (Neumann S, Volkamer M, Jurlind B, Prandrini M, 2016)

As there were various problems with electronic voting especially related to physical security, people began to look for solutions to the problems that's when blockchain came into the field of e-voting, initially blockchain was used for bitcoin. This work discusses how to take the advantages of blockchain technology in the process of e-voting to make it safe, secure, anonymous, etc (Ayed, A.B., 2017)

In their work make use of smart contracts in decentralized blockchain technology for e-voting to engage all voters in evaluating and recording ballots. It increases the trust of electorate and decreases the misuse of election capital (Hsiao JH, Tso R., Chen CM., Wu ME., 2018)

In their study used NetVote for user interface of the program, it uses decentralized application. The dApp admin helps electoral administrators to decide electoral policy, generate voting, register rules, opening and closing of voting. Identification of voters is done by other applications like biometric readers. To test and check the results of election TallydApp is used. This NetVote reinforce three kinds of elections: private election, open election, Token holder elections (Jonath Alexander, Steven Lander, Ben Howerton, 2018)

CHAPTER 3: METHODOLOGY

3.1 Methodology

The waterfall model interprets the software development process in a linear, sequential flow. Software development life cycle (SDLC) models have been created like a waterfall, fountain and fix, spiral build and rapid prototyping, incremental, and synchronize and stabilize. Those are the oldest models, and the waterfall model is the best known. The waterfall approach was the first SDLC Model to be used in Software Engineering to ensure the project's development. The Software Development Life Cycle provides system designers and developers to follow a sequence of activities. A Software Development Life Cycle (SDLC) agglutinates to essential phases for developers, such as planning, analysis, designs, and implementation, and are explained in the section below.

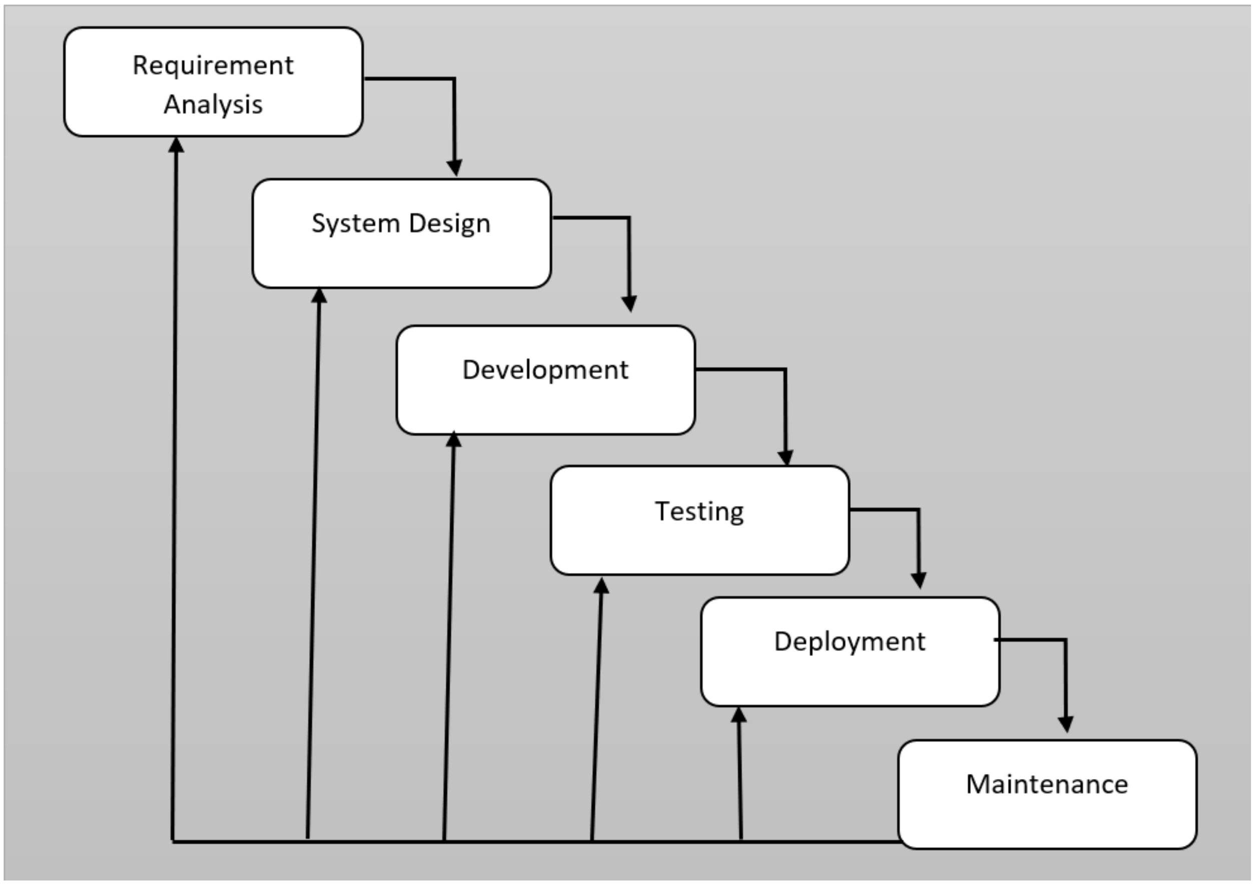


Figure 3.1: Waterfall Model (Winston Walker Royce, 1970)

3.2 Description of Methodology

There are several steps to take an overview of our production house website. They are given below-

Requirement Analysis: This phase captures all possible needs for the system to be developed and documents them in a requirement specification document. We have reviewed many specifications of our project. We identified the hardware and software required for the development by using the waterfall model.

System Design: The goal of this phase is to convert the requirements gathered into a form that can be coded in a programming language. It includes high-level and detailed design as well as the overall software architecture. We created many diagrams to construct an efficient development model, those are use case diagram, data flow diagram and entity relationship diagram.

Development: With inputs from the system design, the system is first code and developed in small programs called units, which are integrated into the next phase. We used Visual Studio Code, Dependency NPM (Node Package Manager), Truffle framework, Ganache, Metamask, React, and Coding language- (Solidity, HTML, CSS, JavaScript).

Testing: In this phase, the software will be tested for its functionality, and if any bug is found, make sure that the bug is fixed. First, we will do the unit test for each unit after development. When the unit test is completed, we will integrate each department into a system and do the integration test. We can ensure that the web-based system is functioning correctly and can be accepted by real-time users by performing website testing. If any bug is found, we will try to fix it.

Deployment: The functional and non-functional testing is done, and the product is deployed in the customer environment or released into the local and global market.

Maintenance: Some issues come up in the client environment. To fix those issues, patches are released. Also, to enhance the product, some better versions are released. Maintenance is done to deliver these changes in the customer environment.

3.3 Justification of Methodology

For our project methodology, we followed the waterfall model because this model is simple and easy to understand and use. It is easy to manage due to the rigidity of the model because each phase of our project has specific deliverables and a review process. In this model, phases are processed and completed one at a time. Phases do not overlap. It allows correcting the errors that are committed, and these changes are reflected within the later phases. Requirements are very well documented, cleared, and fixed. Coding and testing go much faster because they follow to complete small items to code and test. Risk Management and Testing are easy. The waterfall model works well for our project because our requirements are clearly defined and very well understood.

CHAPTER 4: REQUIREMENT ANALYSIS & DESIGN

4.1 System Requirement

This project has two system requirement, Hardware and Software.

4.1.1 Software Requirements

1. Dependency NPM (Node Package Manager)
2. Truffle framework
3. Ganache
4. Metamask
5. Coding language; solidity, HTML, CSS, JavaScript.
6. Windows (OS).

4.1.2 Hardware Requirement

1. 64-bit hardware

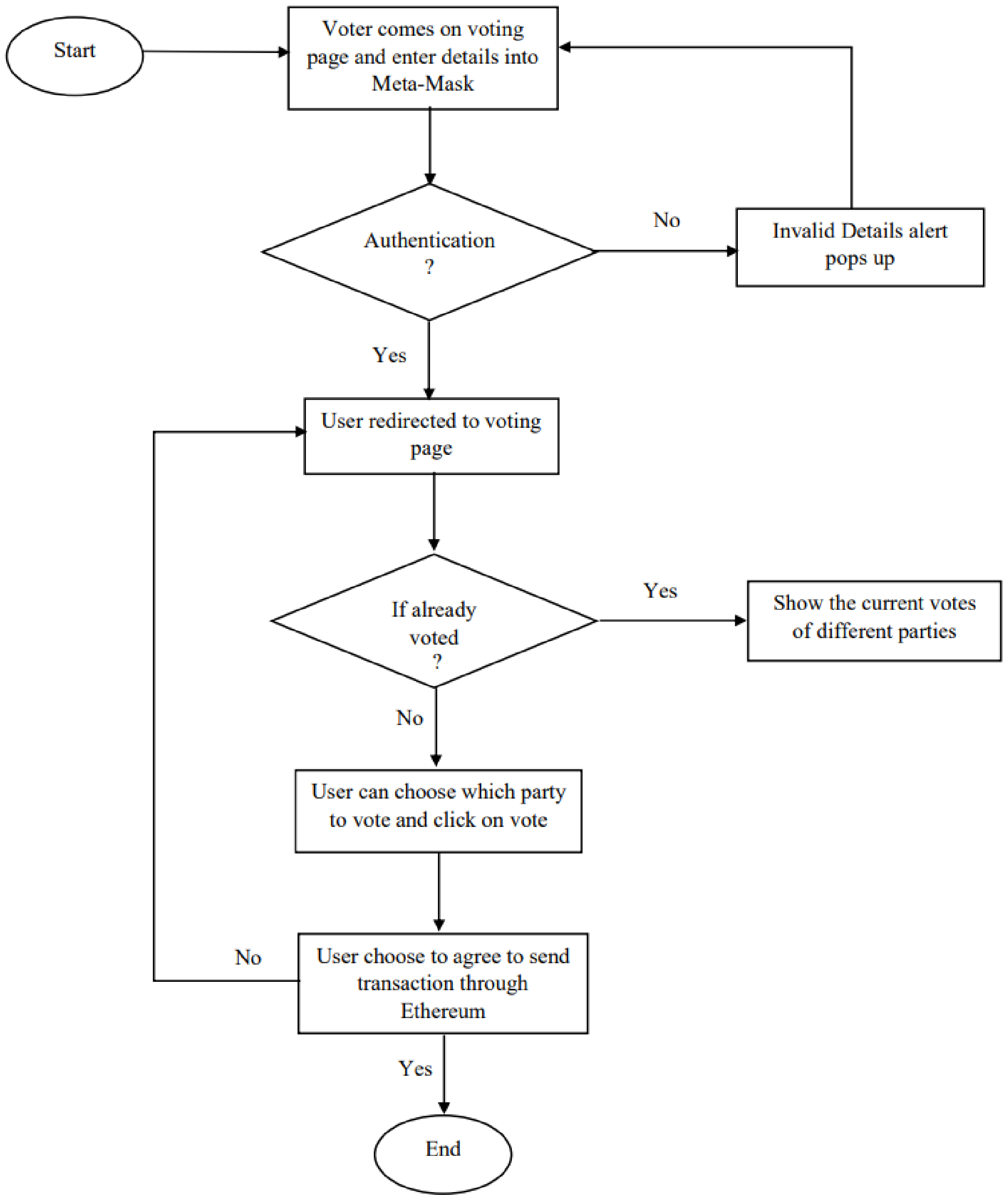
4.2 User Requirements

- Add candidate.
- Import voter account.
- Account authentication.
- Voter can vote once
- Win with maximum vote.

4.3 System Design

4.3.1 Flow chart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.



Start
Voter comes on voting page and enter details into Meta-Mask
Authentication ?
No
Invalid Details alert pops up
Yes
User redirected to voting page
If already voted ?
Yes
Show the current votes of different parties
No
User can choose which party to vote and click on vote
No
User choose to agree to send transaction through Ethereum
Yes
End

Figure 4.1: Flow chart

4.3.2 Use Case Diagram

The use case diagram is a way to summarize details of our system and the users within our system. It is generally shown as a graphic depiction of interactions among different elements in our system.

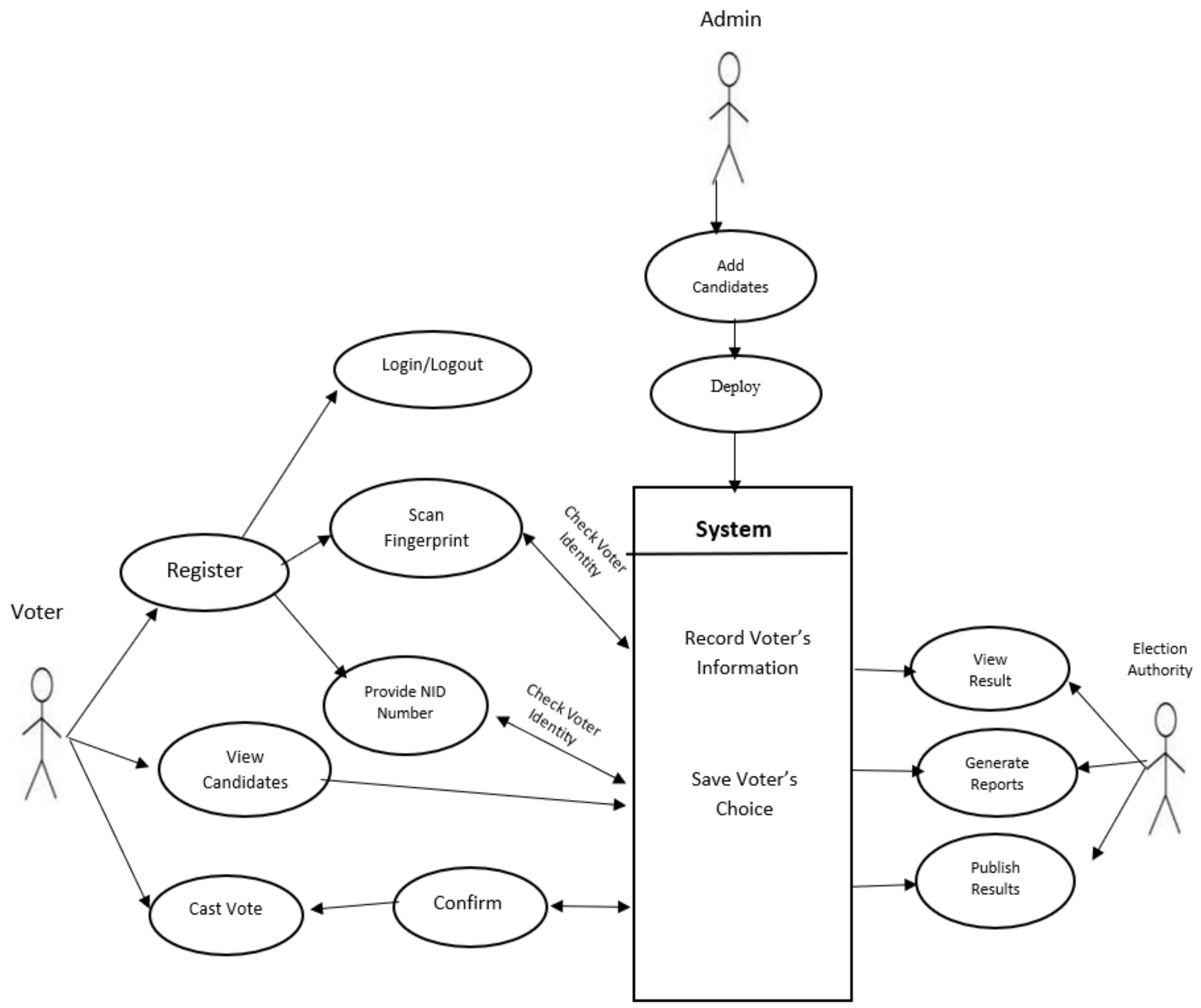


Figure 4.2: Use-Case-Diagram

4.3.3 Data Flow Diagram

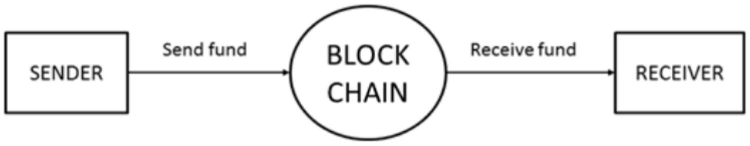


Figure 4.3: DFD Design

1st Level DFD:

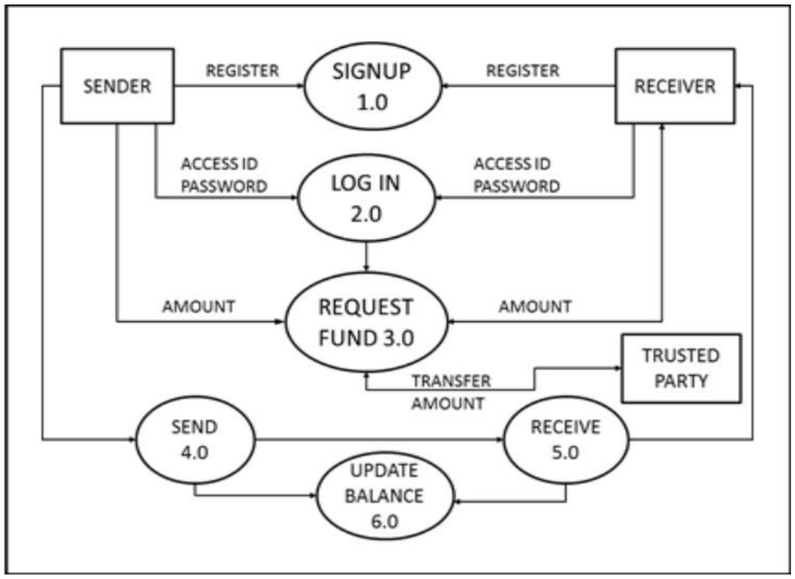


Figure 4.4: 1st Level DFD

2nd Level DFD:

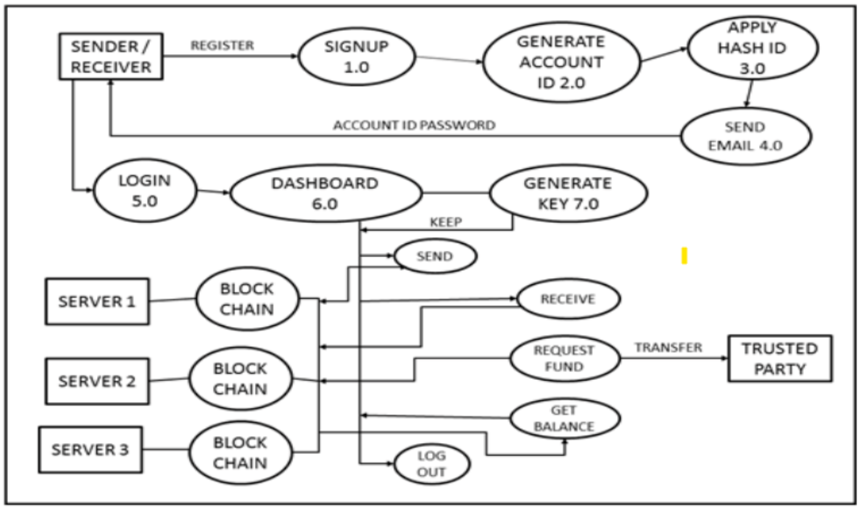


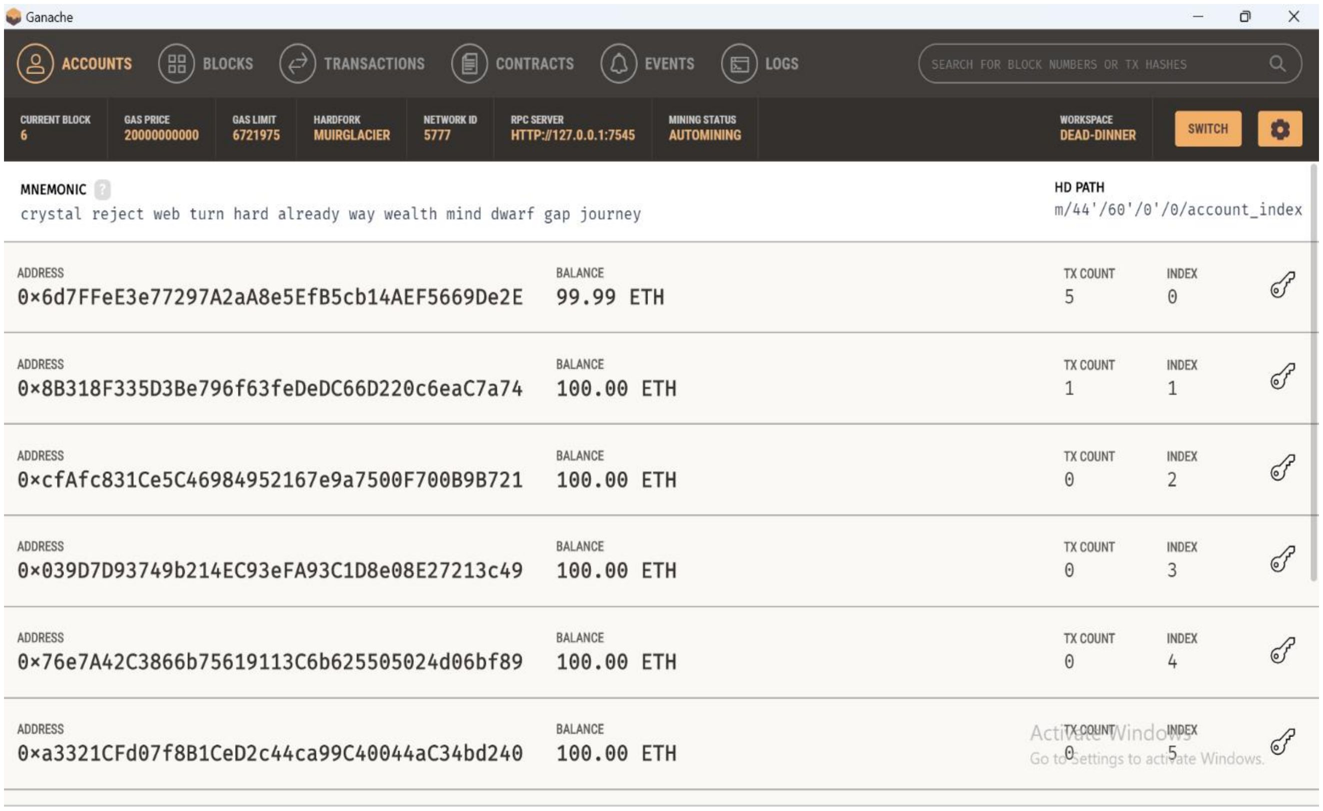
Figure 4.5: 2nd Level DFD

CHAPTER 5: PROJECT DESCRIPTION

5.1 Setting up

The first thing that we need to do is run local blockchain by starting up Ganache.

After setting up ganache there will be not any transaction as we have not done any transaction yet. As we can see from the snapshot below there is no transaction.



CURRENT BLOCK 6 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE DEAD-DINNER

MNEMONIC
crystal reject web turn hard already way wealth mind dwarf gap journey

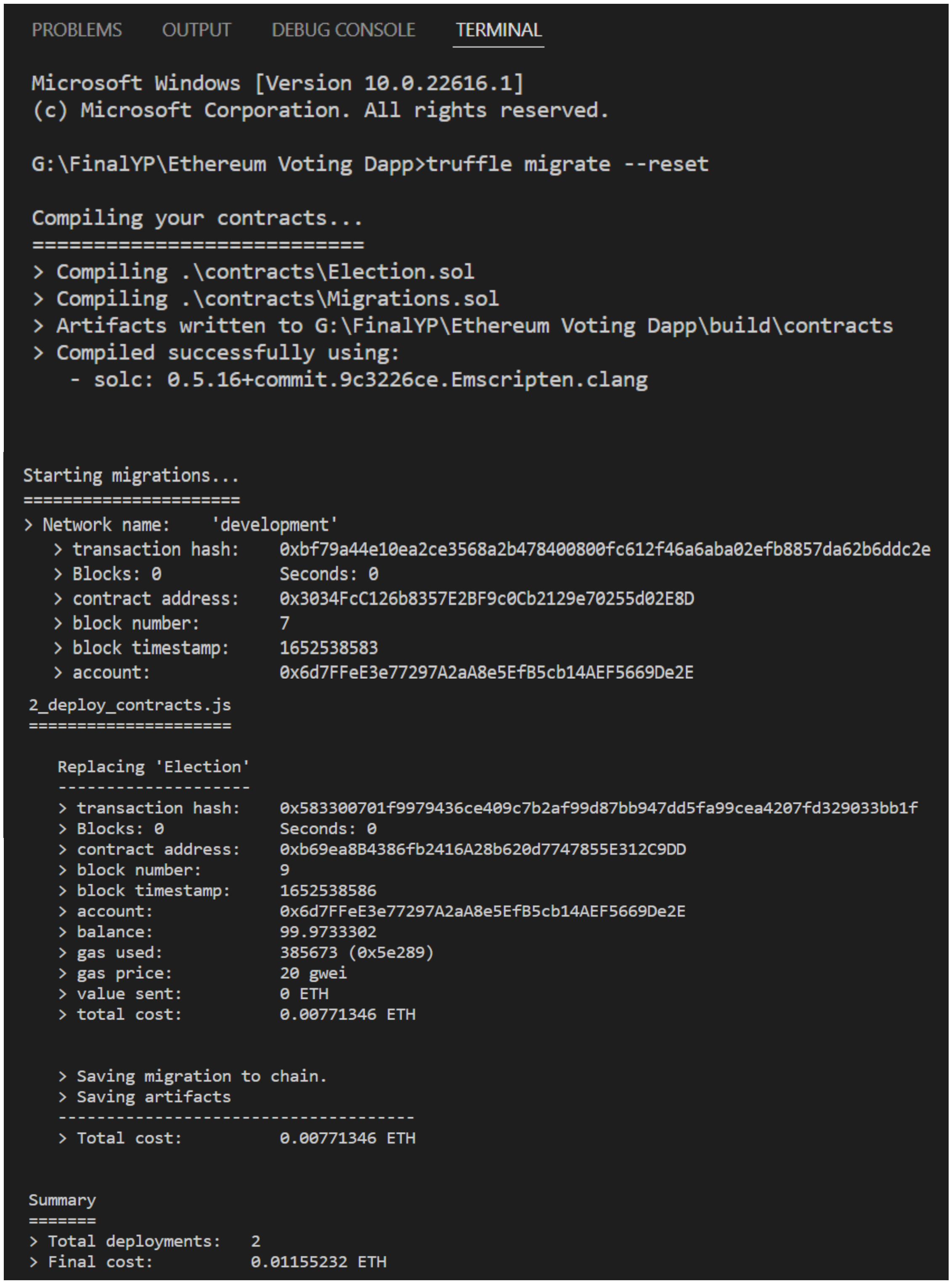
HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×6d7FFeE3e77297A2aA8e5EfB5cb14AEF5669De2E	99.99 ETH	5	0
0×8B318F335D3Be796f63feDeDC66D220c6eaC7a74	100.00 ETH	1	1
0×cfAfc831Ce5C46984952167e9a7500F700B9B721	100.00 ETH	0	2
0×039D7D93749b214EC93eFA93C1D8e08E27213c49	100.00 ETH	0	3
0×76e7A42C3866b75619113C6b625505024d06bf89	100.00 ETH	0	4
0×a3321CFd07f8B1CeD2c44ca99C40044aC34bd240	100.00 ETH	0	5

Activate Windows
Go to Settings to activate Windows.

Figure 5.1: Screenshot of Setting up Ganache.

Now we use truffle framework to transfer the smart contract to the blockchain by giving command on the command line. We have also used NPM directory by cmd. Following commands are being used for this purpose:

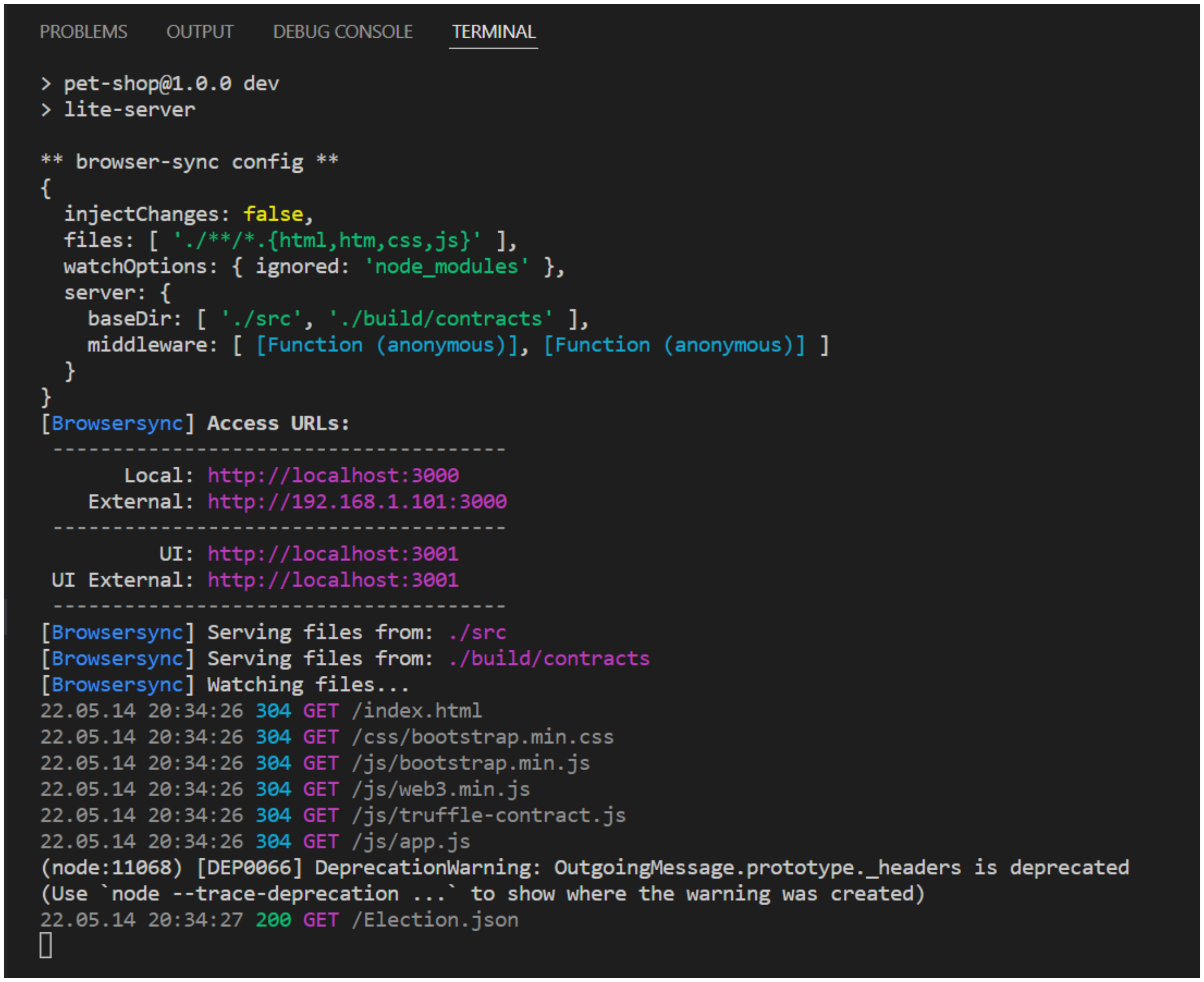


PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
Microsoft Windows [Version 10.0.22616.1]
(c) Microsoft Corporation. All rights reserved.

G:\FinalYP\Ethereum Voting Dapp>truffle migrate --reset

Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to G:\FinalYP\Ethereum Voting Dapp\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



Starting migrations...
======================
> Network name:    'development'
   > transaction hash:    0xbf79a44e10ea2ce3568a2b478400800fc612f46a6aba02efb8857da62b6ddc2e
   > Blocks: 0            Seconds: 0
   > contract address:    0x3034FcC126b8357E2BF9c0Cb2129e70255d02E8D
   > block number:        7
   > block timestamp:     1652538583
   > account:             0x6d7FFeE3e77297A2aA8e5EfB5cb14AEF5669De2E

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

   Replacing 'Election'
   --------------------
   > transaction hash:    0x583300701f9979436ce409c7b2af99d87bb947dd5fa99cea4207fd329033bb1f
   > Blocks: 0            Seconds: 0
   > contract address:    0xb69ea8B4386fb2416A28b620d7747855E312C9DD
   > block number:        9
   > block timestamp:     1652538586
   > account:             0x6d7FFeE3e77297A2aA8e5EfB5cb14AEF5669De2E
   > balance:             99.9733302
   > gas used:            385673 (0x5e289)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00771346 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00771346 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.01155232 ETH
```

Figure 5.2: Snapshot of Command Line for Truffle Framework.

After migrating the smart contract, we start the project using NPM directory by cmd.



```
> pet-shop@1.0.0 dev
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './src', './build/contracts' ],
    middleware: [ [Function (anonymous)], [Function (anonymous)] ]
  }
}
[Browsersync] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://192.168.1.101:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 --------------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
22.05.14 20:34:26 304 GET /index.html
22.05.14 20:34:26 304 GET /css/bootstrap.min.css
22.05.14 20:34:26 304 GET /js/bootstrap.min.js
22.05.14 20:34:26 304 GET /js/web3.min.js
22.05.14 20:34:26 304 GET /js/truffle-contract.js
22.05.14 20:34:26 304 GET /js/app.js
(node:11068) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
22.05.14 20:34:27 200 GET /Election.json
```

Figure 5.3: Command Line of NPM Directory

5.2 User Interface

User interface is through which users can interact with the e-voting system. The picture bellow is how user will see the interface. The loading screen will continue to display loading until the electorate login through metamask.

Below screen will be displayed when the electorate is logging in through metamask.

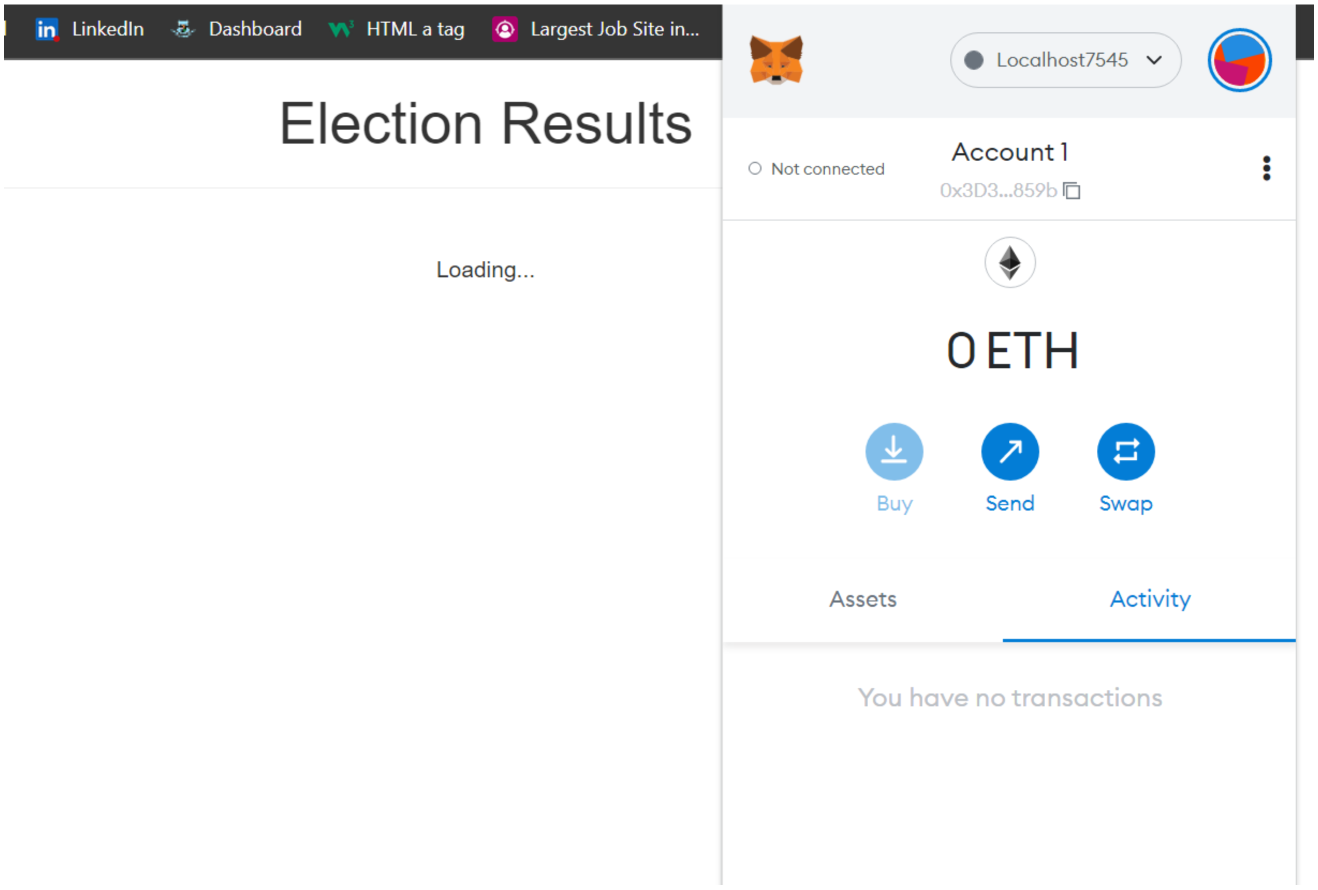
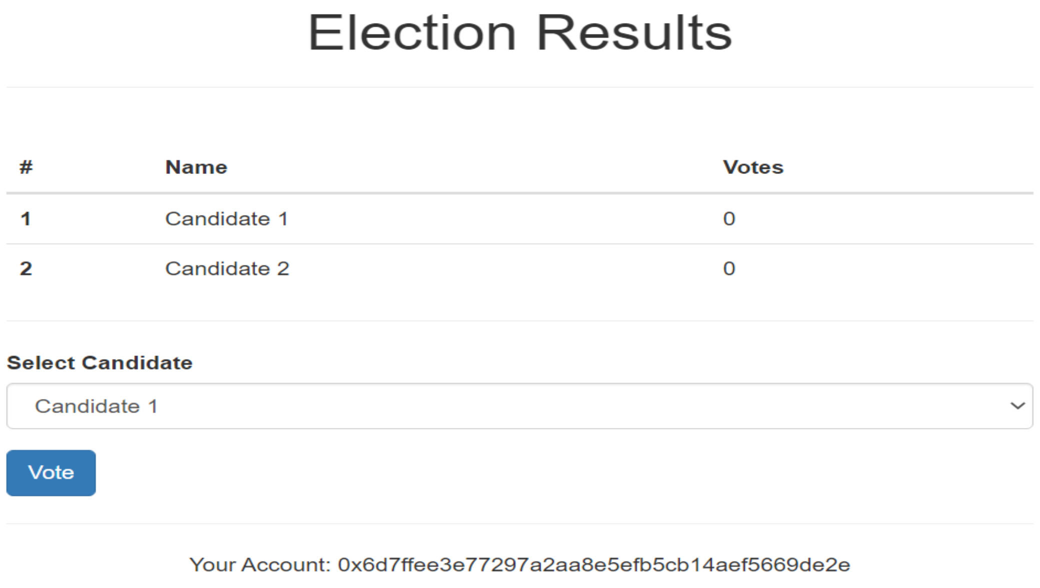


Figure 5.4: Constituency Logging in Via Metamask.

After the user has logged in, main screen comes up with zero vote, the user cannot vote until they import their account by entering private key.



Election Results

#	Name	Votes
1	Candidate 1	0
2	Candidate 2	0

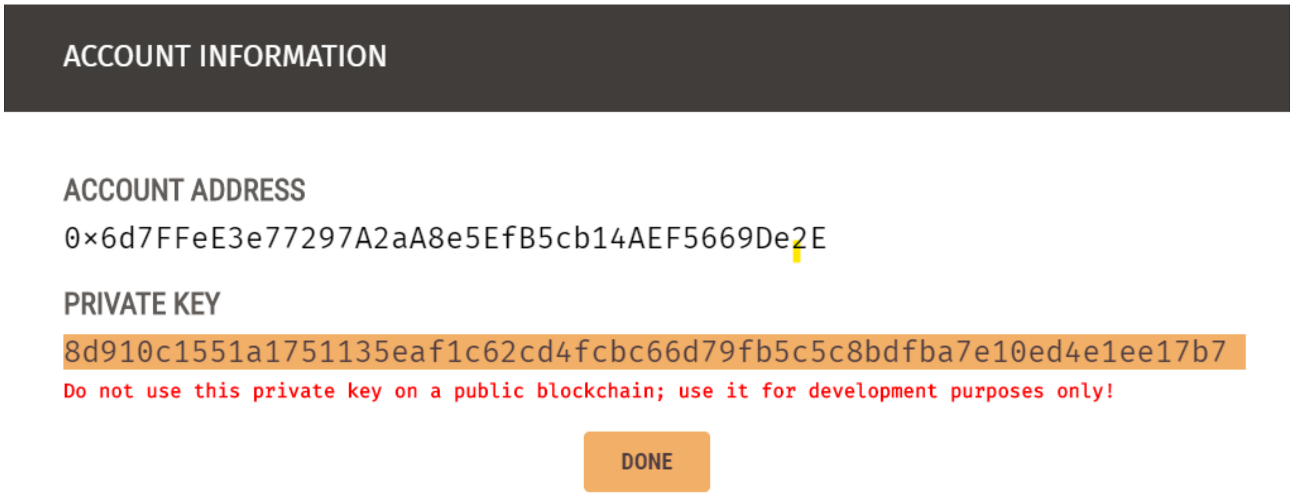
Select Candidate

Candidate 1

Vote

Your Account: 0x6d7ffee3e77297a2aa8e5efb5cb14aef5669de2e

Figure 5.5: Main Screen



ACCOUNT INFORMATION

ACCOUNT ADDRESS

0×6d7FFeE3e77297A2aA8e5EfB5cb14AEF5669De2E

PRIVATE KEY

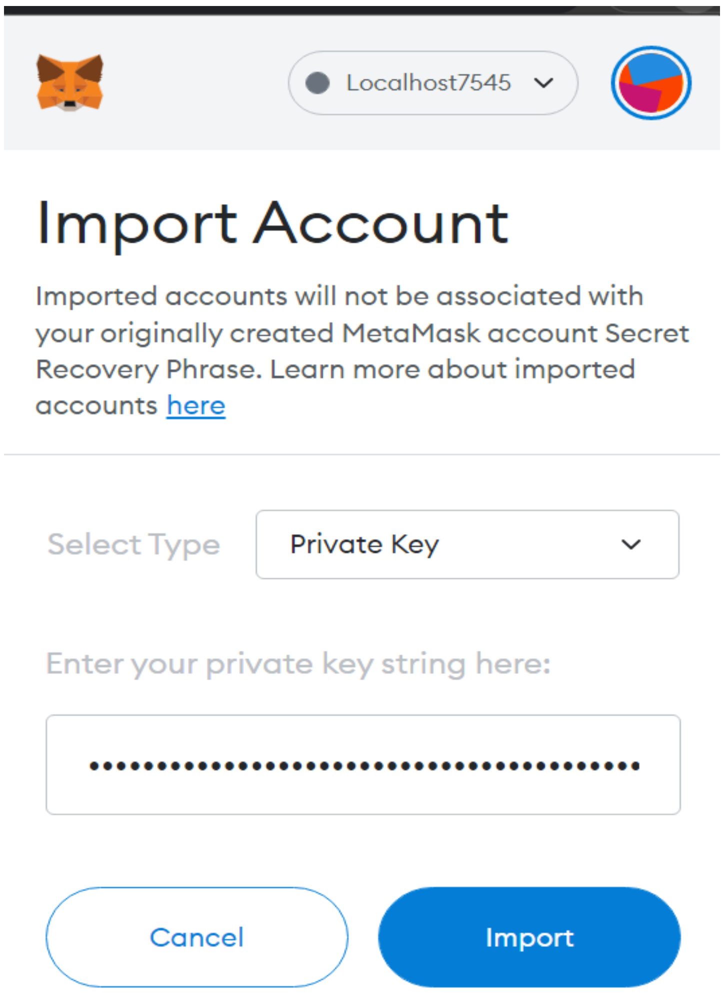
8d910c1551a1751135eaf1c62cd4fcbc66d79fb5c5c8bdfba7e10ed4e1ee17b7

Do not use this private key on a public blockchain; use it for development purposes only!

DONE

Figure 5.6: Private Key.

Voter imports their account by entering the private key above.



Localhost7545

Import Account

Imported accounts will not be associated with your originally created MetaMask account Secret Recovery Phrase. Learn more about imported accounts here

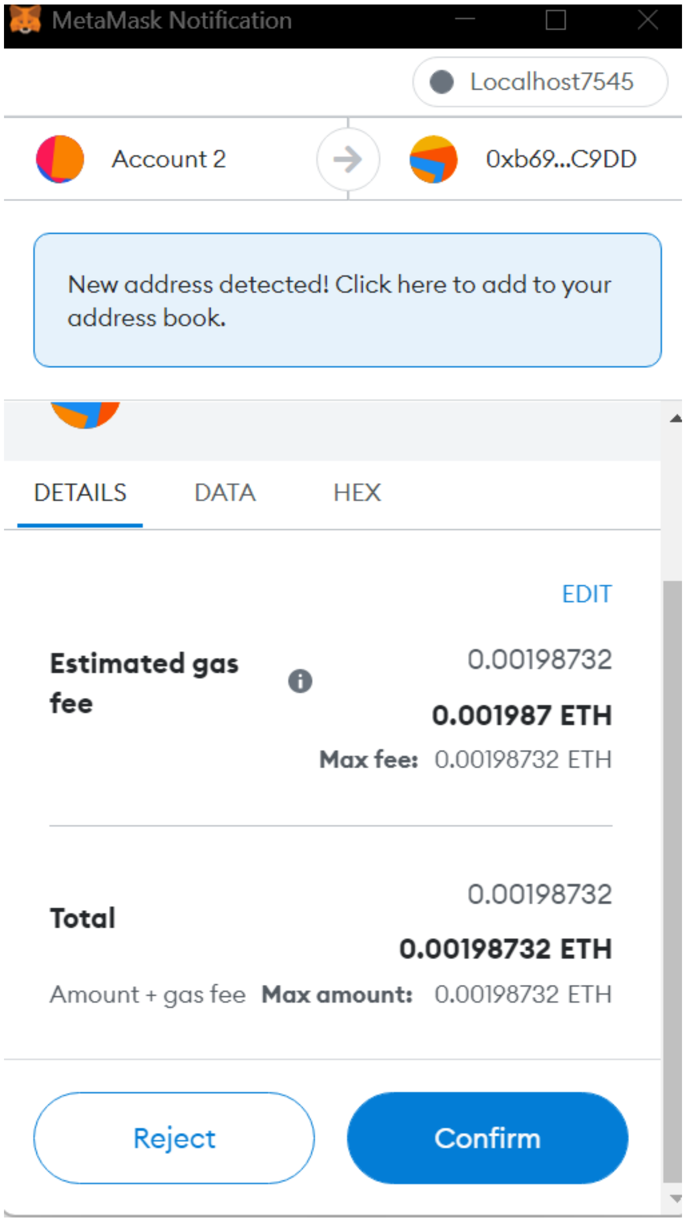
Select Type: Private Key

Enter your private key string here:

••

Figure 5.7: Snapshot of Importing Account.

The electorates choose candidate of their choice, the metamask pop-up gets open when clicked on vote to confirm the transaction


MetaMask Notification

Localhost7545

Account 2 → 0xb69...C9DD

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

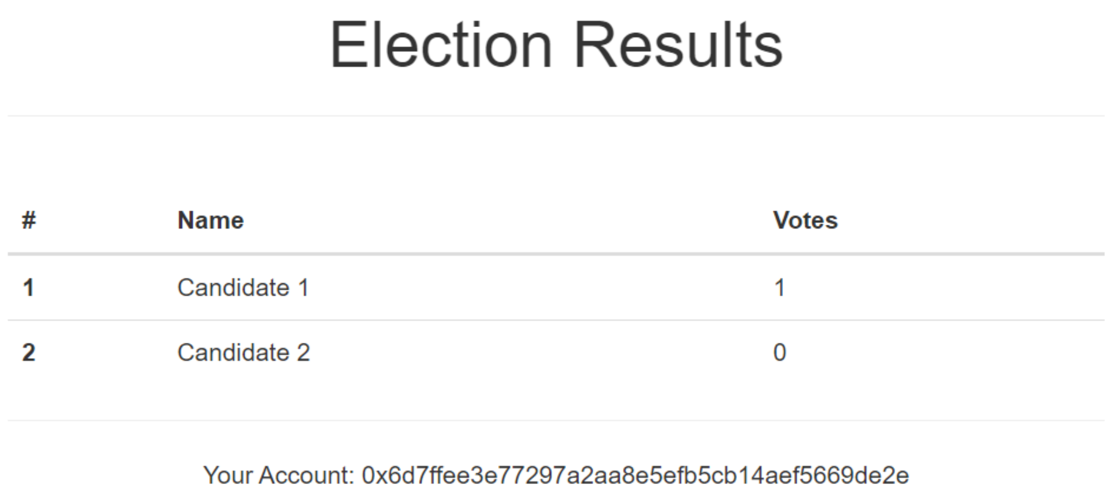
Estimated gas fee 0.00198732 **0.001987 ETH** **Max fee:** 0.00198732 ETH

Total 0.00198732 **0.00198732 ETH**

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

Figure 5.8: Snapshot of Confirming Transaction.

Once confirmation is done, the voter gets redirected to the main page where only results are visible but now you can't vote. In the similar manner, others can also vote by importing their account.



Election Results

#	Name	Votes
1	Candidate 1	1
2	Candidate 2	0

Your Account: 0x6d7ffee3e77297a2aa8e5efb5cb14aef5669de2e

Figure 5.9: Snapshot of Result on Main Page.

CHAPTER 6: CONCLUSION

6.1 Conclusion

The recent development in the area of the voting system includes Blockchain technology, which not only proved to be time and cost-efficient but is also safe and secure, hence is more reliable and precise than the earlier approaches. In this project, we will use blockchain-based e-voting using a smart contract that includes rules governing the communication and decision on the contract between parties. Various tools like Ganache, Truffle framework, NPM and Metamask were used for implementation as blockchain technology is decentralized. Tempering and alteration in such a system are quite attainable. Our proposed method provides convenience to the voters by allowing them to connect to an easy-to-use user interface system. They can cast their vote by importing their account and easily reviewing their vote. It creates a sense of trust among voters that their vote is being computed and kept in safe custody.

6.2 Limitation

Though this project has many advantages, it has some limitation as well, and this is listed Below-

- NID validator not used in the decentralize system.

6.3 Future Works

Anyone can develop the below mentioned features in future with their project-

- Validate the certificate-chain present in the NID and verify the eligibility of the voting citizen using a decentralized network of oracles.
- Extend support in the Dapp to Metamask alternatives such as Fortmatic.
- Store voting keys in the internal memory of the NID, in furtherance of convenience to the voting citizen.

REFERENCES:

Aranha DF, Ribeiro H, Paraense ALO. (2016) ‘Crowdsourced integrity verification of election results’, Annals of Telecommunications, 71(7-8), pp. 1–11.

Ayed, A.B. (2017) ‘A Conceptual Secure Blockchain Based Electronic Voting System’, International Journal of Network Security & Its Applications (IJNSA), 9(3), pp. 1-9.

Budurushi J, Renaud K, Volkamer M, Woide M. (2016) ‘An investigation into the usability of electronic voting systems for complex elections’, Annals of Telecommunications, 71(7-8), pp. 309-322.

E. Elewa, A. AlSammak, A. AbdElRahman, T.ElShishtawy. (2015) ‘Challenges of Electronic Voting-A Survey’, Advances in Computer Science: An International Journal, 4(6), pp. 98-108.

Gjøsteen K, Lund AS. (2016) ‘An experiment on the security of the norwegian electronic voting protocol’, Annals of Telecommunications, 71(7-8), pp. 299-307.

Hsiao JH, Tso R., Chen CM., Wu ME. (2018) ‘Decentralized E-Voting Systems Based on the Blockchain Technology’, Advances in Computer Science and Ubiquitous Computing. CUTE 2017, CSA 2017. Lecture Notes in Electrical Engineering, vol 474. Springer, Singapore.

Jonath Alexander, Steven Lander and Ben Howerton. (2018) Netvote. A Decentralized Voting Network Available at: https://netvote.io/wp content/uploads/2018/02/NetvoteWhite-Paper-v7.pdf

Neumann S, Volkamer M, Jurlind B, Prandrini M. (2016) ‘Secivo: a quantitative security assessment model for internet voting schemes’, Annals Telecommunication, 71(7-8), pp. 337-352.

APPENDIX:

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Election Results</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <div class="container" style="width: 650px;">
      <div class="row">
        <div class="col-lg-12">
          <h1 class="text-center">Election Results</h1>
          <hr/>
          <br/>
          <div id="loader">
            <p class="text-center">Loading...</p>
          </div>
          <div id="content" style="display: none;">
            <table class="table">
              <thead>
                <tr>
                  <th scope="col">#</th>
                  <th scope="col">Name</th>
                  <th scope="col">Votes</th>
                </tr>
              </thead>
              <tbody id="candidatesResults">
              </tbody>
            </table>
            <hr/>
            <form onSubmit="App.castVote()">
              <div class="form-group">
                <label for="candidatesSelect">Select Candidate</label>
                <select class="form-control" id="candidatesSelect">
                </select>
              </div>
              <button type="submit" class="btn btn-primary">Vote</button>
              <hr />
            </form>
            <p id="accountAddress" class="text-center"></p>
          </div>
        </div>
      </div>
    </div>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as
needed -->
    <script src="js/bootstrap.min.js"></script>
    <script src="js/web3.min.js"></script>
    <script src="js/truffle-contract.js"></script>
    <script src="js/app.js"></script>
  </body>
</html>
```

App.js

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',
  hasVoted: false,

  init: function() {
    return App.initWeb3();
  },

  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');
      web3 = new Web3(App.web3Provider);
    }
    return App.initContract();
  },

  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);

      // App.listenForEvents();
      return App.render();
    });
  },

  // Listen for events emitted from the contract
  listenForEvents: function() {
    console.log("intro to event.");
    App.contracts.Election.deployed().then(function(instance) {
      instance.votedEvent({}, {
        fromBlock: 0,
        toBlock: 'latest'
      }).watch(function(error, event) {
        console.log("event triggered", event)
        // Reload when a new vote is recorded
        console.log("end of event.");
        App.render();
      });
    });
  },

  render: function() {
    console.log("intro to render.");
    var electionInstance;
    var loader = $("#loader");
    var content = $("#content");

    loader.show();
    content.hide();

    // Load account data
    web3.eth.getCoinbase(function(err, account) {
      if (err === null) {
        App.account = account;
        $("#accountAddress").html("Your Account: " + account);
      }
    });

    // Load contract data
    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[2];

          // Render candidate Result
          var candidateTemplate = "<tr><th>" + id + "</th><td>" + name +
"</td><td>" + voteCount + "</td></tr>"
          candidatesResults.append(candidateTemplate);

          // Render candidate ballot option
          var candidateOption = "<option value='" + id + "' >" + name + "</
option>"
          candidatesSelect.append(candidateOption);
        });
      }
      return electionInstance.voters(App.account);
    }).then(function(hasVoted) {
      // Do not allow a user to vote
      if(hasVoted) {
        $('form').hide();
      }
      loader.hide();
      content.show();
    }).catch(function(error) {
      console.warn(error);
    });
    console.log("end of render.");
  },

  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();
    }).catch(function(err) {
      console.error(err);
    });
  }
};

$(function() {
  $(window).load(function() {
    App.init();
  });
});
```

Election.sol

```
pragma solidity >=0.4.22 <0.8.0;

contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        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
    );
    constructor() public {
        addCandidate("Candidate 1");
        addCandidate("Candidate 2");
    }
    function addCandidate (string memory _name) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name, 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);
    }
}
```

deploy_contracts.js

```
var Election = artifacts.require("./Election.sol");

module.exports = function(deployer) {
  deployer.deploy(Election);
};
```

Text/election.js

```
var Election = artifacts.require("./Election.sol");

contract("Election", function(accounts) {
  var electionInstance;

  it("initializes with two candidates", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
    }).then(function(count) {
      assert.equal(count, 2);
    });
  });

  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], "Candidate 1", "contains the correct name");
      assert.equal(candidate[2], 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], "Candidate 2", "contains the correct name");
      assert.equal(candidate[2], 0, "contains the correct votes count");
    });
  });

  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[2];
      assert.equal(voteCount, 1, "increments the candidate's vote count");
    })
  });

  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[2];
      assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
      return electionInstance.candidates(2);
    }).then(function(candidate2) {
      var voteCount = candidate2[2];
      assert.equal(voteCount, 0, "candidate 2 did not receive any votes");
    });
  });

  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[2];
      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[2];
      assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
      return electionInstance.candidates(2);
    }).then(function(candidate2) {
      var voteCount = candidate2[2];
      assert.equal(voteCount, 1, "candidate 2 did not receive any votes");
    });
  });
});
```