

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/351871006>

Decentralized Voting Application Using Ethereum Smart Contract

Technical Report · March 2019

DOI: 10.13140/RG.2.2.22372.09605

CITATION

1

READS

139

5 authors, including:



Gananjay Thanekar

Somaiya Vidyavihar

8 PUBLICATIONS 10 CITATIONS

[SEE PROFILE](#)



Sulochana Madachane

South Indian Education Society's Graduate School of Technology

4 PUBLICATIONS 1 CITATION

[SEE PROFILE](#)

Decentralized Voting Application Using Ethereum Smart Contract

Submitted in partial fulfillment of the requirements of the degree of

“Bachelor of Computer Engineering”

by

Gananjay Sandeep Thanekar (Roll No.: 45)

Aveesh Ashok Shetty (Roll No.: 37)

Sagar Gayadhar Sethi (Roll No.:36)

Supervisors:

Prof. Sulochana Madachane

Prof. Hasib Shaikh



DEPARTMENT OF COMPUTER ENGINEERING

**K.C.COLLEGE OF ENGINEERING & MANAGEMENT STUDIES &
RESEARCH, THANE (E)**

University of Mumbai

2018-19

CERTIFICATE

This is to certify that the project entitled “**Decentralized Voting Application Using Ethereum Smart Contract**” is a bonafide work of “(Roll No.: 37) Aveesh Shetty (Roll No.: 45) Gananjay Thanekar (RollNo.:36) Sagar Sethi” submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of “**Bachelor of Computer Engineering**”

Mr. Hasib Shaikh
Supervisor/Guide

Mrs. Sulochana Madachane
Supervisor/Guide

Mr. Mandar Ganjapurkar
Head of Department

Dr. Vijaykumar N. Pawar
Principal

Project Report Approval for B.E.

This project report entitled **Decentralized Voting Application Using Ethereum Smart Contract** by **Aveesh Shetty, Gananjay Thanekar, Sagar Sethi** is approved for the degree of Bachelor of Engineering in **Computer Engineering**.

Examiners

1.-----

2.-----

Date:

Place: Thane

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of student and Roll No.

Signature

1. (Roll No.: 37) Aveesh Shetty
2. (Roll No.: 45) Gananjay Thanekar
3. (Roll No.:36) Sagar Sethi

Date:

ACKNOWLEDGEMENT

No project is ever complete without the guidance of those expert who have already traded this past before and hence become master of it and as a result, our leader. So we would like to take this opportunity to take all those individuals how have helped us in visualizing this project.

We would like to express special thanks of gratitude to our guide Mrs. Sulochana Madachane and Mr. Hasib Shaikh as well as our Project Coordinator Mrs. Asmita Deshmukh and Mrs. Dhanashri Kanade who gave us the golden opportunity to do this wonderful project on the topic of Decentralized Voting Application Using Ethereum Smart Contract, which also helped us in doing a lot of research and we came to know about so many new things.

We are very grateful to our Head of the Department Mr. Mandar Ganjapurkar for extending his help directly and indirectly through various channels in our project work.

We extend our sincerity appreciation to our Principal Dr. V. N. Pawar for providing us the opportunity to implement our project.

We are really thankful to all our Professors from K.C. College of Engineering and Management Studies and Research for their valuable inside and tip during the designing of the project. Their contributions have been valuable in so many ways that we find it difficult to acknowledge of them individual.

Thanking You.

TABLE OF CONTENT

Sr. No.	Topic	Page No.
	Certificate.....	i
	Approval Sheet.....	ii
	Declaration.....	iii
	Acknowledgement.....	iv
	Table of Content.....	v
	List of Figures.....	vi
	List of Tables.....	vii
	Abstract.....	viii
1.	Introduction.....	1
2.	Review of Literature.....	2
	2.1 Existing System	
3.	Report on Present Investigation.....	4
	3.1 Requirement Analysis.....	4
	3.1.1 Scope.....	4
	3.1.2 Feasibility Study.....	4
	3.1.2.1 Operational Feasibility.....	4
	3.1.2.2 Technical Feasibility.....	4
	3.1.2.3 Economic Feasibility	4
	3.1.3 Hardware & Software Requirement.....	5
	3.2 Problem Statement.....	5
	3.3 Project Design	6
	3.3.1 Data Flow Diagram.....	6
	3.4 Methodology.....	8
	3.5 Implementation Plan.....	15
	3.5.1 Semester VI.....	15
	3.5.2 Semester VIII.....	15
4.	Results and Discussion.....	16
5.	Conclusion.....	22
6.	References.....	23
7.	Publications by the candidate.....	24

List of Figures

Sr. No	Name of figure	Page No.
1	Centralized System Vs Decentralized System	1
2	DFD level-0	6
3	DFD level-1	6
4	DFD level-2	7
5	System Flow	8
6	Setup IP Address	12
7	Creation Of Account	13
8	Migrating Application	16
9	Serving Application To The Browser	16
10	Election Homepage	17
11	Setting Up New IP Address	17
12	Ganache Dummy Accounts	18
13	Private Key Of Individual Account	18
14	Importing New Account	19
15	Account Homepage	19
16	Confirmation Page	20
17	Candidate Dashboard	20
18	Transaction Details	21

List of Tables

Sr. No	Name of figure	Page No.
1.	Review Literature	2
2.	Semester VII Activity Table	15
3.	Semester VIII Activity Table	15

ABSTRACT

The voting system in India as well as in some countries abroad is flawed and can be easily manipulated and hampered by those with power to suit their personal benefits. It allows people with money to buy the votes or tamper the machine that record it. A current example can be the recent elections in Uttar Pradesh where the ruling party manipulated the voting machines to gain unfair advantage

A voting system which is unhackable or which cannot be tampered. Using blockchain based distributed networks to store and record votes which allows voting to be almost 99.9% secure as it is distributed among a cluster of networks and requires computing power of over 500 supercomputers to even get hold of the network which is currently impossible to achieve. The system has a user friendly interface and can be used to run polls and elections with total security and zero percent chance of manipulation

Chapter 1

Introduction

Block chain is a peer-to-peer connection of nodes that talk to each other. On the block chain the data doesn't reside in a single server. It is distributed across every node. All of this data is contained in bundles of records called blocks which are chained together to create a public ledger and all of the nodes work together to ensure that data in the publication because it means that our account sent the transaction when we vote and our vote will go to the candidate and be recorded forever. Because all of the data is shared across devices in the block chain, it is database and as each node talk to each other it is a network so instead of traditional web model you can think of block chain as database and network all in one. The Ethereum allows us write code that get executed on the ethereum virtual machine with smart contracts. This is where our business logic of our application will lie. This is the code that'll responsible for reading and writing the data transfer the value and executing any business logic that we program.

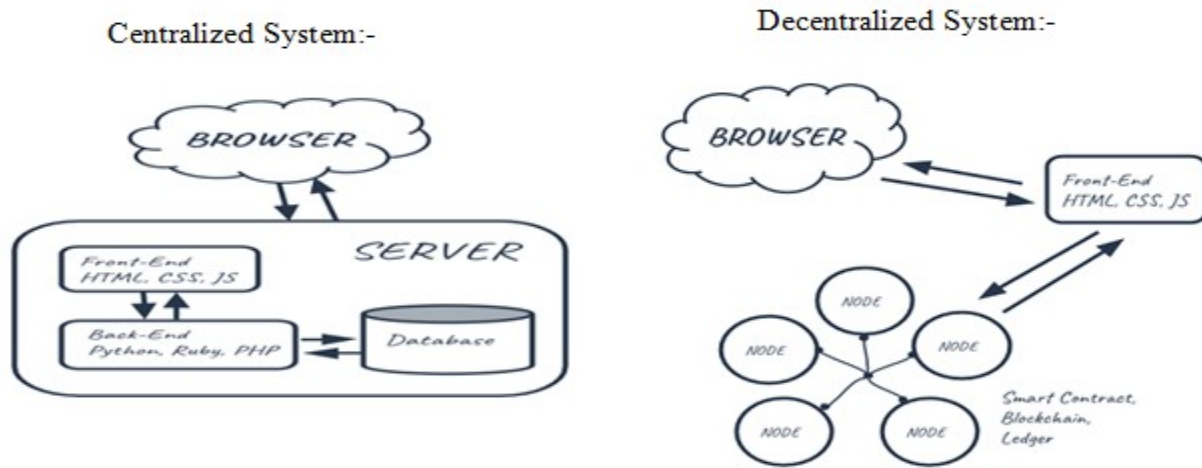


Figure 1.1- Centralized System Vs Decentralized System

Chapter 2

Review of Literature

Table of Literatures Referred

Sr.No	Year of Publishing	Author name	Paper name	Methodology
1	19-21 Dec. 2016	Angus K. Y. Wong ,Sai-Man Cheok , Su-Kit Tang	A user-friendly voting and quiz system for classroom use with connected devices	Design a voting system that is easy-to-use for teachers and students. Implement the backend server so that teachers can create and update the voting/quiz questions on the web. Implement a check-in system so that students can browse the voting/quiz page by scanning a QR code or entering a randomly generated code. Implement the statistic system so that teacher can view the voting/quiz result.
2	15-16 Dec. 2017	B Madhuri ,M G Adarsha , K R Pradhyumna , B M Prajwal	Secured Smart Voting System using Aadhar	Proposed system is App based Biometric online voting system, which determines whether a particular person is eligible for casting vote by authenticating his/her finger print. Voter details are retrieved from the Aadhar Database and verified. Detail about the casted vote is updated to the respective Database.
3	26-27 Oct. 2017	Rifa Hanifatunnisa , Budi Rahardjo	Blockchain based e-voting recording system design	Permission Blockchain inversely proportional to the previous type, operated by known entities such as consortium blockchains, where consortium members or stakeholders in a particular business context operate a Blockchain permission network. This Blockchain permission system has means to identify nodes that can control and update data together, and often has ways to control who can issue transactions.
4	27 May-3 June 2018	Péter Hegedus	Towards Analyzing the Complexity Landscape of Solidity Based Ethereum Smart	The primary concepts enabling such general use of the blockchain are the so-called smart contracts, which are special programs that run on the blockchain. One of the most popular blockchain platforms

			Contracts	that supports smart contracts are Ethereum. As smart contracts typically handle money, ensuring their low number of faults and vulnerabilities are essential. To aid smart contract developers and help maturing the technology, we need analysis tools and studies for smart contracts
5	07 March 2018	Jason Paul Cruz Yuichi Kaji Naoto Yanai	Role-Based Access Control Using Smart Contract	We present a role-based access control using smart contract (RBAC-SC), a platform that makes use of Ethereum's smart contract technology to realize a transorganizational utilization of roles. Ethereum is an open blockchain platform that is designed to be secure, adaptable, and flexible. It pioneered smart contracts, which are decentralized applications that serve as "autonomous agents" running exactly as programmed and are deployed on a blockchain.
6	iacr.org 2017	Patrick McCorry, Siamak F. Shahandashti and Feng Hao	A Smart Contract for Boardroom Voting with Maximum Voter Privacy	Ethereum's blockchain provides a natural platform for the Open Vote Network. It provides a public bulletin board and an authenticated broadcast channel which are necessary in decentralized internet voting protocols to support coordination amongst voters. As well, almost all computations in the Open Vote Network are public computations that can be written as a smart contract.

Chapter 3

Report on Present Investigation

3.1 Requirement Analysis

1. Scope

The Ethereum allows us write code that get executed on the ethereum virtual machine with smart contracts .This is where our business logic of our application will lie. This is the code that'll responsible for reading and writing the data transfer the value and executing any business logic that we program. We can think of smart contract as micro-service that lives on the web. If public ledger represents data layer, then smart contract represents the layer reads and writes the data , does stuff with it and talk to other services. Also its called a smart contract because it represents some kind of covenant or agreement.

2. Feasibility study

A voting system which is unhackable or which cannot be tampered. Using blockchain based distributed networks to store and record votes which allows voting to be almost 99.9% secure as it is distributed among a cluster of networks. The system has a user friendly interface and can be used to run polls and elections with total security and zero percent chance of manipulation.

3. Technical feasibility

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. Compact and offline system

4. Economic feasibility

The project is economically feasible as it's weighed against the current tangible and intangible benefits. Overall cost reduced.

5. Operational feasibility

Operational feasibility is a measure of how well a proposed system solves the problem. To ensure success, desired operational outcomes are imparted during design and development.

6. Hardware and Software requirements

Hardware requirements:

CPU: Intel Core i5 5th gen or above

Ram: 4GB or above

HDD: 10 GB or more

Software requirements:

OS: Windows 7, 8, 10, Linux

Web Browser: Microsoft Edge, Google Chrome

Ganache

Metamask Extension

3.2 Problem Statement

The voting system in India as well as in some countries abroad is flawed and can be easily manipulated and hampered by those with power to suit their personal benefits. A current example can be the recent elections in Uttar Pradesh where the ruling party manipulated the voting machines to gain unfair advantage. The Voting System that we are making is an Online Web-Application. Our System is more secure as, once the votes are given to the candidates it cannot be tempered. The System which we have made is decentralized in nature, which adds more advantage to our system than the traditional centralized system which is currently present.

3.3 Project Design

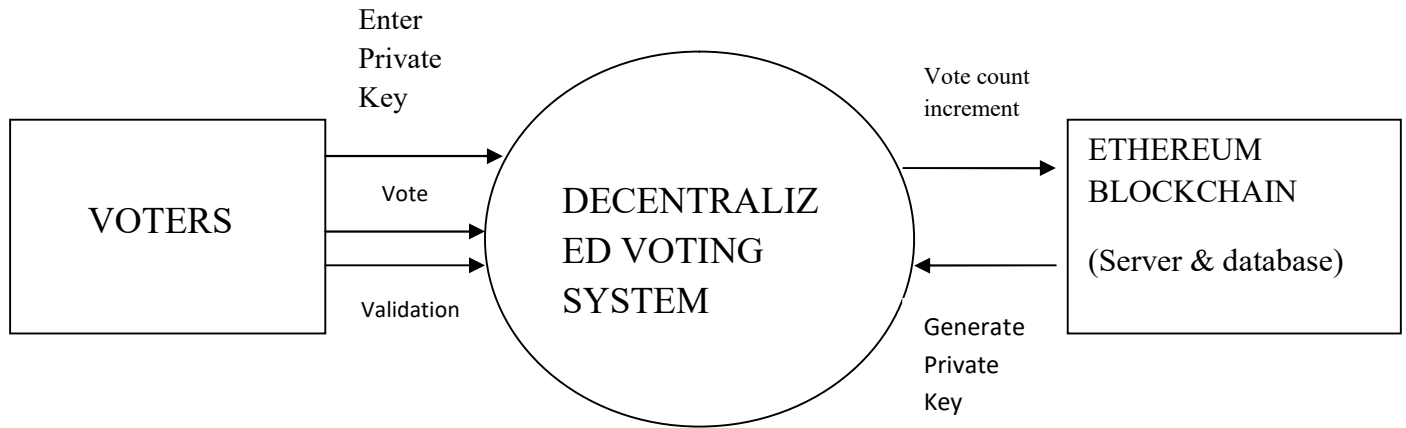


Figure 3.1 - DFD Level 0

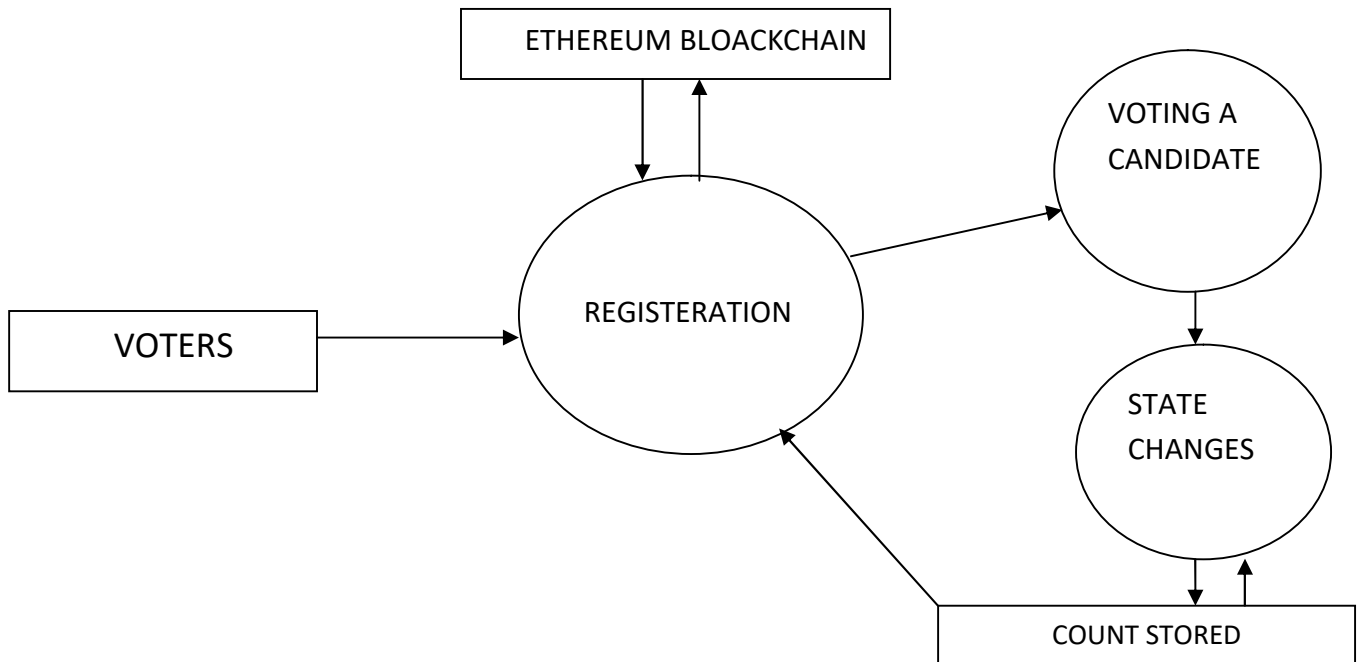


Figure 3.2 – DFD Level 1

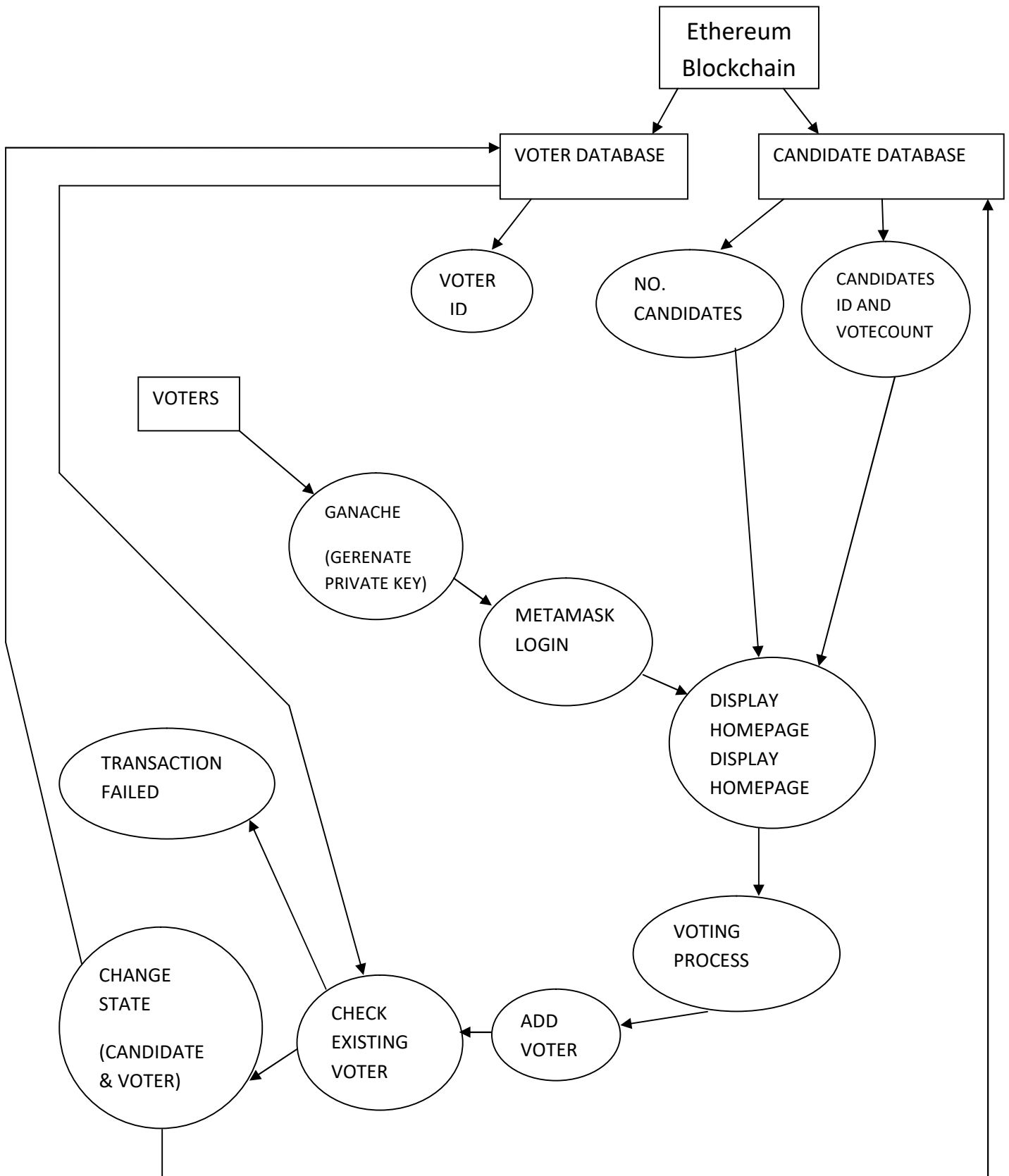


Figure 3.3 – DFD Level 2

3.4 Methodology

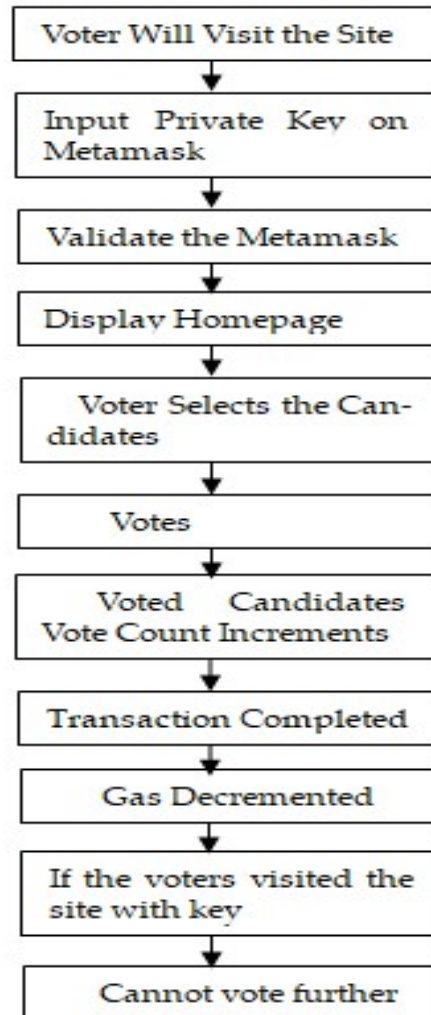


Figure 3.4 – System Flow

The Ethereum allows us write code that get executed on the ethereum virtual machine with smart contracts .This is where our business logic of our application will lie. This is the code that'll responsible for reading and writing the data transfer the value and executing any business logic that we program. We can think of smart contract as micro-service that lives on the web. If public ledger represents data layer, then smart contract represents the layer reads and writes the data, does stuff with it and talk to other services. Also it's called a smart contract because it represents some kind of covenant or agreement.

A voting system which is unhackable or which cannot be tampered. Using blockchain based distributed networks to store and record votes which allows voting to be almost 99.9% secure as it is distributed among a cluster of networks. The system has a user friendly interface and can be used to run polls and elections with total security and zero percent chance of manipulation.

3.4.1 Setting up the development environment

- Instead of developing the app against the live blockchain, we will use an in- memory blockchain (think of it as a blockchain simulator) called ganache.
- We will interact with the real blockchain. Below are the steps to install ganache, web3js and start the test blockchain on a linux operating system.
- Notice that the ganache-cli creates 10 test accounts to play with automatically. These accounts come preloaded with 100 (fake) ethers.

3.4.2 Create a project using truffle

- To use most Truffle commands, you need to run them against an existing Truffle project. So the first step is to create a Truffle project.
- You can create a bare project template, but for those just getting started, you can use Truffle Boxes, which are example applications and project templates. We'll use the pet-shop box, which creates a token that can be transferred between accounts.

- Create a new directory for your truffle project:

```
mkdir pet-shop
```

```
cd pet-shop
```

- Download (“unbox”) the pet-shop box:

```
truffle unbox pet-shop
```

- Once this operation is completed, you'll now have a project structure with the following items:

1. Contracts/: Directory for Solidity Contracts
2. Migrations/: Directory for scriptable deployment files
3. Test/: Directory for test files for testing your application and contracts
4. Truffle.js/: Truffle configuration file

3.4.3 Compiling Contracts

- Location: All of your contracts are located in your project's contracts/ directory. As contracts are written in Solidity, all files containing contracts will have a file extension of .sol. Associated Solidity libraries will also have a .sol extension.
- Command: To compile a Truffle project, change to the root of the directory where the project is located and then type the following into a terminal “truffle compile”
- Build artifacts: Artifacts of your compilation will be placed in the build/contracts/ directory, relative to your project root. (This directory will be created if it does not exist.). These artifacts are integral to the inner workings of Truffle, and they play an important part in the successful deployment of your application.
- Dependencies: You can declare contract dependencies using Solidity's import command. Truffle will compile contracts in the correct order and ensure all dependencies are sent to the compiler.
- Dependencies can be specified in two ways:
 1. Importing dependencies via file name
 2. Importing contracts from an external package

3.4.4 Running Migrations

- Migrations are JavaScript files that help you deploy contracts to the Ethereum network.
- These files are responsible for staging your deployment tasks, and they're written under the assumption that your deployment needs will change over time.
- As your project evolves, you'll create new migration scripts to further this evolution on the blockchain.
- A history of previously run migrations is recorded on-chain through a special Migrations contract
- To run your migrations, write the command:

\$ truffle migrate

3.4.5 Testing the contracts

- Framework: Truffle comes standard with an automated testing framework to make testing your contracts a breeze. This framework lets you write simple and manageable tests in two different ways:
 1. In Javascript and TypeScript, for exercising your contracts from the outside world, just like your application.
 2. In Solidity, for exercising your contracts in advanced, bare-to-the-metal scenarios.
- Location: All test files should be located in the ./test directory. Truffle will only run test files with the following file extensions: .js, .ts, .es, .es6, and .jsx, and .sol. All other files are ignored.
- Command: To run all tests, simply run “\$ truffle test”
- Clean-room environment: Truffle provides a clean room environment when running your test files. When running your tests against Ganache or Truffle Develop, Truffle will use advanced snapshotting features to ensure your test files don't share state with each other. When running against other Ethereum clients like go-ethereum, Truffle will re-deploy all of your migrations at the beginning of every test file to ensure you have a fresh set of contracts to test against.
- Speed and reliability considerations: Both Ganache and Truffle Develop are significantly faster than other clients when running automated tests. Moreover, they contain special features which Truffle takes advantage of to speed up test runtime by almost 90%. As a general workflow, we recommend you use Ganache or Truffle Develop during normal development and testing, and then run your tests once against go-ethereum or another official Ethereum client when you're gearing up to deploy to live or production networks.

3.4.6 Setup Ganache (Ethereum client)

- Ganache is a personal blockchain for Ethereum development you can use to deploy contracts, develop your applications, and run tests.
- When you launch Ganache, the screen will show some details about the server, and also list out a number of accounts. Each account is given 100 ether. Having ether automatically in all accounts allows you to focus on developing your application.
- Ganache is a personal blockchain for Ethereum development you can use 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). Ganache is available for Windows, Mac, and Linux.
- Ganache CLI: If you are interested in the command-line version (formerly known as the TestRPC), you can get it through npm

```
npm install -g ganache-cli
```

- *Ganache Accounts*, There are four pages available:
 1. The Accounts page shows the accounts generated and their balances. This is the default view.
 2. The Blocks page shows each block as mined on the blockchain, along with gas used and transactions.
 3. The Transactions page lists all transactions run against the blockchain.
 4. The Logs page shows the logs for the server, which is useful for debugging.

Also note that you can search for block numbers or transaction hashes from a search box at the top.

- *Ganache Settings*
 1. The Server page shows details about the network connection, including hostname, port, network ID, and whether to automatically mine each transaction into a block.
 2. The Accounts & Keys page sets details about the number of accounts created, and whether to use a specific mnemonic or let Ganache generate its own.
 3. The Chain page sets details about the actual workings of the generated blockchain, including gas limit and gas price.
 4. The Advanced page toggles Google Analytics, which is useful for the Ganache team to track usage of the application.

After making changes, you will have to click Restart on the application for the changes to take effect.

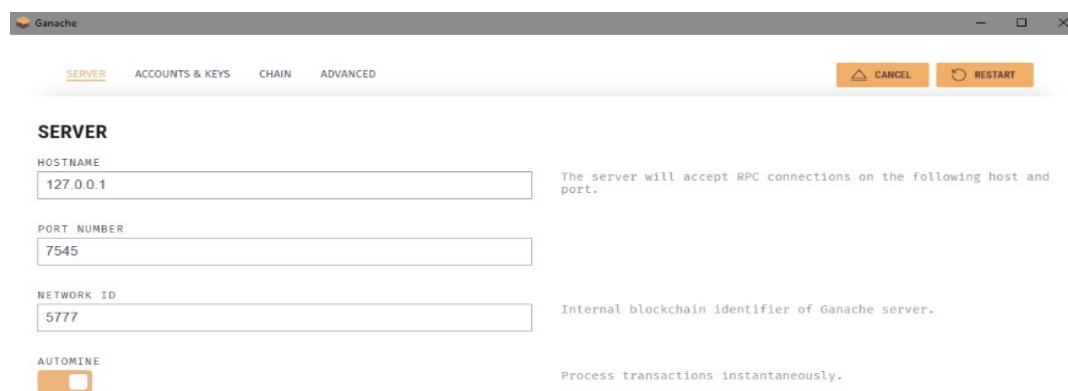


Figure 3.5 – Setup IP Address

3.4.7 Setup Metamask

- MetaMask is the easiest way to interact with dapps in a browser. It is an extension for Chrome or Firefox that connects to an Ethereum network without running a full node on the browser's machine.
- It can connect to the main Ethereum network, any of the test nets (Ropsten, Kovan, and Rinkeby), or a local blockchain such as the one created by Ganache or Truffle Develop.
- Setting up MetaMask
 1. Click Import with seed phrase. In the box marked Wallet Seed, enter the mnemonic that was displayed when launching Ganache.
 2. Enter a password below that and click OK
 3. Now we need to connect MetaMask to the blockchain created by Ganache. Click the menu that shows "Main Network" and select Custom RPC.
- *MetaMask network menu:* In the box titled "New RPC URL" (to the right of "New Network") enter `http://127.0.0.1:7545` and click **Save**. The network name at the top will switch to say "Private Network". Click the cross in the top-right of the current window close out of the page and return to the Accounts page. Now that we've connected MetaMask to Ganache, you'll be taken to the accounts screen. Each account created by Ganache is given 100 ether. The first account should have less than the others because that account supplies the gas for smart contract deployment. Since you've deployed your smart contract to the network, this account paid for it. Click the account icon in the upper-right to create new accounts, the first 10 of which will correspond to the 10 accounts displayed when you launched Ganache.

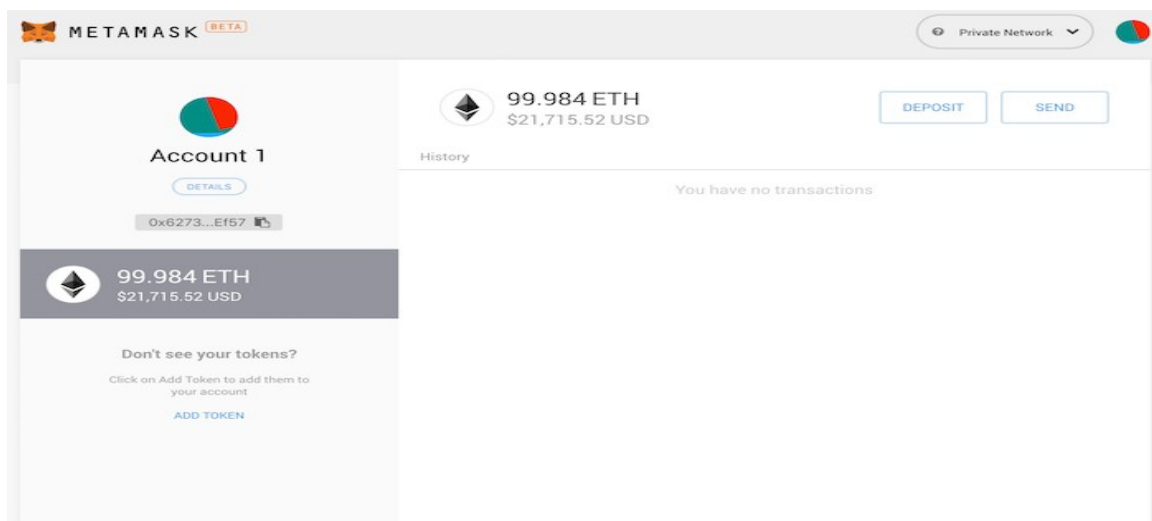


Figure 3.6 – Creation Of Account

3.4.8 Pay Eth And Vote

- One of the friction points for Ethereum dapp adoption is that the users have to pay gas(txn fee) to get their transactions recorded on the blockchain.
- A user who wants to record her vote on the blockchain has to pay a transaction/gas fee.
- This is not ideal because as a dapp owner, you are expecting your application users to have Ether to pay for gas when all they want to do is perform a simple action which has nothing to do with transferring money.
- But if the transaction needs to be executed on the blockchain, there is no other option but to pay the fee.

3.5 Implementation plan

3.5.1 Semester VII Activity Table

Activity	Days	Start Date	End date
Guide meeting	1	08/24/18	08/24/18
Research on Dapp	10	08/25/18	09/06/18
Testing Solidity Js	11	09/07/18	09/21/18
Testing GAS	11	09/22/18	10/05/18

3.5.2 Semester VIII Activity Table

Activity	Days	Start Date	End date
Setting up the development environment	8	01/05/19	01/12/18
Create a project using truffle	8	01/14/19	01/21/19
Compiling Contracts	6	01/26/19	01/31/19
Running Migrations	8	02/02/19	02/09/19
Testing The contracts		02/11/2019	02/17/19
Setup ganache	8	02/19/2019	02/26/2019
Setup metamask	5	02/28/2019	03/04/2019
Voting Demo	4	03/07/2019	03/10/2019

Chapter 4

Results

Output1:

```

lite-server
C:\Users\Aveesh shetty\election-2019_update>truffle migrate --reset
Important
If you're using an HDWalletProvider, it must be Web3 1.0 enabled or your migration will hang.

Starting migrations...
> Network name: 'development'
> Network id: 5777
> Block gas limit: 6721975

1_initial_migration.js
=====
Replacing 'Migrations'
> transaction hash: 0xac832b16c559d5fda728774ae0deab6ea921543f1c49365fdd9e1a0174d8f379
> Blocks: 0 Seconds: 0
> contract address: 0x4085CddB5360e1E84536b379F44E118da3bd194D
> account: 0x4150b8181E3997acB20177D3E77F91b0e1a5ab78
> balance: 99.7810194
> gas used: 277462
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.00554924 ETH

> Saving migration to chain.
> Saving artifacts
> Total cost: 0.00554924 ETH

2_deploy_contracts.js
=====
Replacing 'Election'
> transaction hash: 0x93032809fb4f0f68728c7f7d4f9408830b50ca5548735618f50342cb0ad37e54
> Blocks: 0 Seconds: 0
> contract address: 0xd9A7CD03B923b6b0dbe6f78c0a04b308bd609c13
> account: 0x4150b8181E3997acB20177D3E77F91b0e1a5ab78
> balance: 99.77078962
> gas used: 469481
> gas price: 20 gwei

```

Figure 4.1 – Migrating Application

```

lite-server

> Saving migration to chain.
> Saving artifacts
> Total cost: 0.00554924 ETH

2_deploy_contracts.js
=====
Replacing 'Election'
> transaction hash: 0x93032809fb4f0f68728c7f7d4f9408830b50ca5548735618f50342cb0ad37e54
> Blocks: 0 Seconds: 0
> contract address: 0xd9A7CD03B923b6b0dbe6f78c0a04b308bd609c13
> account: 0x4150b8181E3997acB20177D3E77F91b0e1a5ab78
> balance: 99.77078962
> gas used: 469481
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.00938962 ETH

> Saving migration to chain.
> Saving artifacts
> Total cost: 0.00938962 ETH

Summary
=====
> Total deployments: 2
> Final cost: 0.01493886 ETH

C:\Users\Aveesh shetty\election-2019_update>npm run dev

> election@1.0.0 dev C:\Users\Aveesh shetty\election-2019_update
> lite-server

** browser-sync config **
{ injectChanges: false,
  files: [ './**/*.html,css,js' ] },

```

Figure 4.2 – Serving Application To The Browser

Output2:

Election Results

#	Name	Votes
1	Candidate 1	0
2	Candidate 2	0

Select Candidate


Candidate 1

Vote


Your Account: 0x2786848b3f0ba7deaba39b72c3ba77f4dda0e55d

Figure 4.3 – Election Homepage

Output3:

 METAMASK

http://127.0.0.1:7545



Settings

Info

Currency Conversion

Updated Mon Mar 11 2019 20:54:47 GMT+0530 (India Standard Time)

USD - United States Dollar

Primary Currency

Select native to prioritize displaying values in the native currency of the chain (e.g. ETH). Select Fiat to prioritize displaying values in your selected fiat currency.

☒ ETH ☐ Fiat

Show Conversion in Testnets

Select this to show fiat conversion in when you are on Testnets

☐

Current Language

English

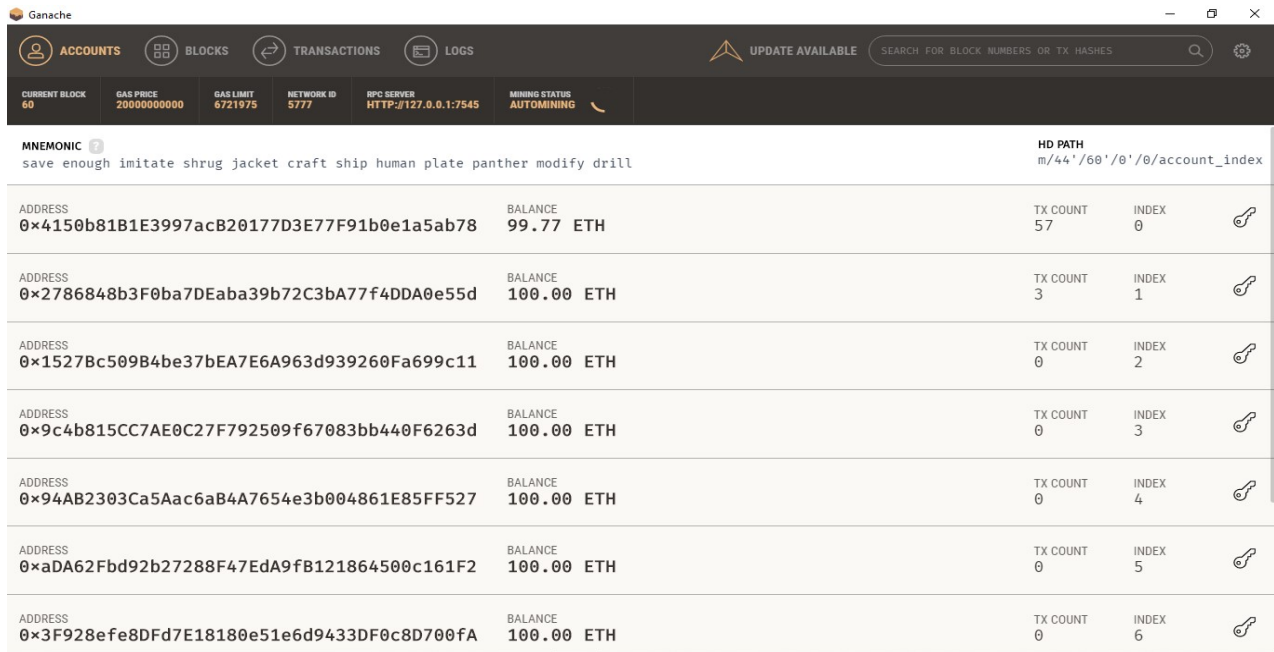
English

New Network

New RPC URL

Figure 4.4 – Setting Up New IP Address

Output4:



ACCOUNTS	BLOCKS	TRANSACTIONS	LOGS
<p>ACCOUNTS</p> <p>CURRENT BLOCK: 60 GAS PRICE: 20000000000 GAS LIMIT: 6721975 NETWORK ID: 5777 RPC SERVER: HTTP://127.0.0.1:7545 MINING STATUS: AUTOMINING</p>			
<p>MNEMONIC</p> <p>save enough imitate shrug jacket craft ship human plate panther modify drill</p>		<p>HD PATH</p> <p>m/44'/60'/0'/0'/account_index</p>	
<p>ADDRESS</p> <p>0x4150b81B1E3997acB20177D3E77F91b0e1a5ab78</p>	<p>BALANCE</p> <p>99.77 ETH</p>	<p>TX COUNT</p> <p>57</p>	<p>INDEX</p> <p>0</p>
<p>ADDRESS</p> <p>0x2786848b3F0ba7DEaba39b72C3bA77f4DDA0e55d</p>	<p>BALANCE</p> <p>100.00 ETH</p>	<p>TX COUNT</p> <p>3</p>	<p>INDEX</p> <p>1</p>
<p>ADDRESS</p> <p>0x1527Bc509B4be37bEA7E6A963d939260Fa699c11</p>	<p>BALANCE</p> <p>100.00 ETH</p>	<p>TX COUNT</p> <p>0</p>	<p>INDEX</p> <p>2</p>
<p>ADDRESS</p> <p>0x9c4b815CC7AE0C27F792509f67083bb440F6263d</p>	<p>BALANCE</p> <p>100.00 ETH</p>	<p>TX COUNT</p> <p>0</p>	<p>INDEX</p> <p>3</p>
<p>ADDRESS</p> <p>0x94AB2303Ca5Aac6aB4A7654e3b004861E85FF527</p>	<p>BALANCE</p> <p>100.00 ETH</p>	<p>TX COUNT</p> <p>0</p>	<p>INDEX</p> <p>4</p>
<p>ADDRESS</p> <p>0xaDA62Fbd92b27288F47EdA9fB121864500c161F2</p>	<p>BALANCE</p> <p>100.00 ETH</p>	<p>TX COUNT</p> <p>0</p>	<p>INDEX</p> <p>5</p>
<p>ADDRESS</p> <p>0x3F928efe8DFd7E18180e51e6d9433DF0c8D700fA</p>	<p>BALANCE</p> <p>100.00 ETH</p>	<p>TX COUNT</p> <p>0</p>	<p>INDEX</p> <p>6</p>

Figure 4.5 – Ganache Dummy Accounts

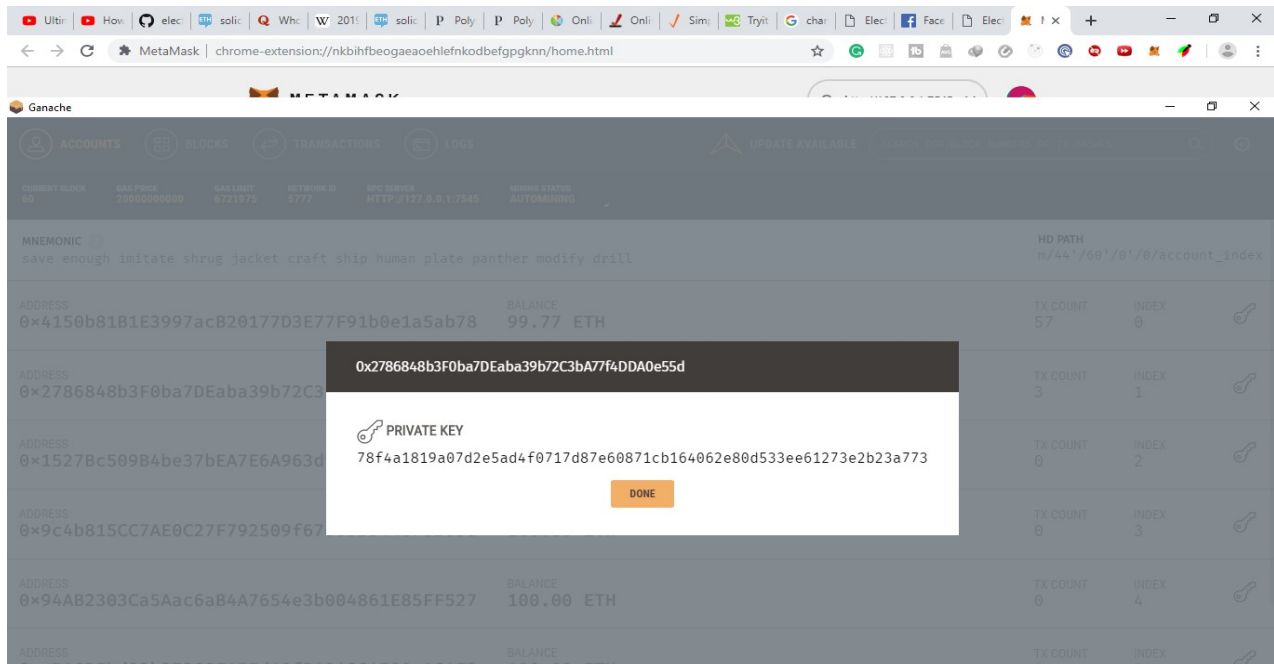


Figure 4.6 – Private Key Of Individual Account

Output5:

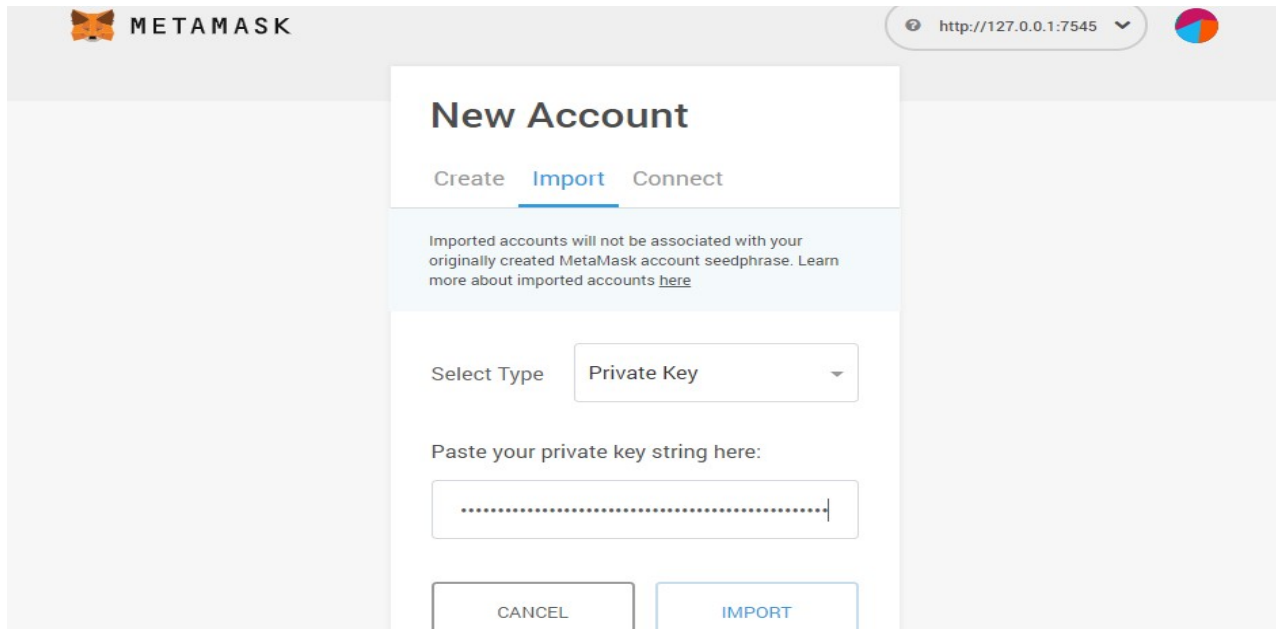


Figure 4.7 – Importing New Account

Output6:

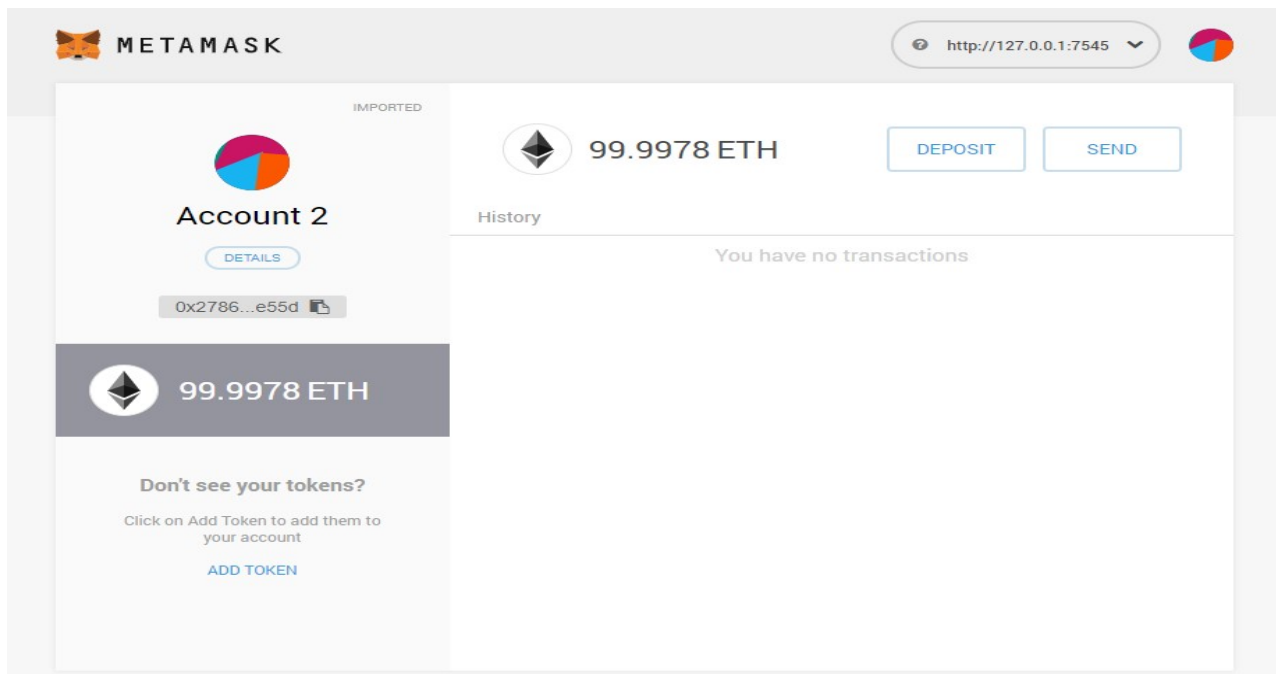


Figure 4.8 – Account Homepage

Output7:

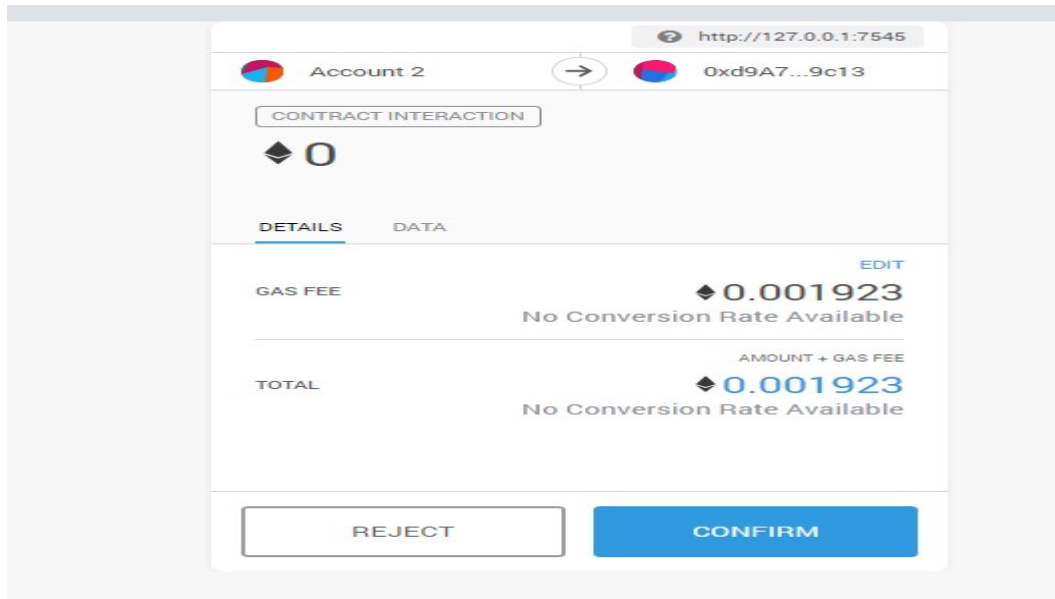


Figure 4.9 – Confirmation Page

Output8:

The screenshot shows a web browser displaying the 'Election Results' dashboard. The browser's address bar shows 'localhost:3000'. The dashboard has a title 'Election Results' and a table with the following data:

#	Name	Votes
1	Candidate 1	1
2	Candidate 2	0

Below the table, it says 'Your Account: 0x2786848b3f0ba7deaba39b72c3ba77f4dda0e55d'. At the bottom right, there is a dark grey notification box that says 'Confirmed transaction' and 'Transaction 3 confirmed! View on'.

Figure 4.10 – Candidate Dashboard

Output9:

The screenshot displays the Metamask web application interface. At the top, the Metamask logo and the text 'METAMASK' are visible. The address bar shows 'http://127.0.0.1:7545'. The main interface is divided into several sections:

- Account 2:** Located on the left, it shows the account name 'Account 2', a 'DETAILS' button, and the address '0x2786...e55d'.
- Balance:** Below the account name, it shows '99.9966 ETH'.
- Don't see your tokens?:** A section with the text 'Click on Add Token to add them to your account' and an 'ADD TOKEN' button.
- Transaction History:** A section titled 'History' showing a 'Contract Interaction #3 - 3/11/2019 at 20:56' with a 'CONFIRMED' status and a value of '-0 ETH'.
- Transaction Details:** A section titled 'Details' showing the transaction from '0x2786848b3F0ba7DEaba39b72C3bA77f4DDA...' to '0xd9A7CD03B923b6b0dbe6f7Bc0a04b308bd609c...'. It includes a table with transaction details and an 'Activity Log'.

Transaction	
Amount	0 ETH
Gas Limit (Units)	96151
Gas Used (Units)	64101
Gas Price (GWEI)	20
Total	0.001282 ETH

Activity Log

- Transaction created with a value of 0 ETH at 20:56 on 3/11/2019.
- Transaction submitted with gas fee of 0 WEI at 20:56 on 3/11/2019.
- Transaction confirmed at 20:56 on 3/11/2019.

Figure 4.11 – Transaction Details

Chapter 5

Conclusion

In a traditional web application, a hacker can hack into code that is residing in the central server or the database. Voting has been always offline due to the fact that making the voting process online can result in tampering of votes leading to victory of wrong candidate. DAPP stands for Decentralized web application solves the problem of a traditional web application. Blockchain is a peer-to-peer connection of nodes that talk to each other. If you have a device and it's connected to the blockchain network then you are a node and you talk to all the other nodes. Ethereum blockchain technology makes the web application secure and decentralized by applying the concept of cryptography and decentralization. Each user has to connect to Ethereum blockchain so that it can be a node of the ethereum network and can contribute to keeping the public ledger and contribute in voting process. The voting count that is stored on the public ledger available to all the nodes but is encrypted so no one can edit it neither the rules that are defined in the code or the data in public ledger that act as the database. Thus we conclude that we believe that the traditional web application should be replaced by decentralized web application so that developer of the website shouldn't have all of the user's data They may use them to manipulate the user for their own benefits. The rules defined in the application are applicable to all users whether it is normal user or admin.

Chapter 6

References

1. Secured Smart Voting System using Aadhar e-ISSN: 2395-0056 Volume: 05 Issue: 01 | Jan-2018 www.irjet.net p-ISSN: 2395-0072 Blockchain based e-voting recording system design (26-27 Oct. 2017) Kanchan Avhad, Kalyani Avhad, Gayatri Bhosale, Kamini Kamale
2. Towards Analyzing the Complexity Landscape of Solidity Based Ethereum Smart Contracts (27 May-3 June 2018) IEEE Péter Hegedus Conference Location: Gothenburg, Sweden, Sweden
3. Role-Based Access Control Using Smart Contract (07 March 2018) IEEE Jason Paul Cruz ; Yuichi Kaji ; Naoto Yanai Electronic ISSN: 2169-3536 INSPEC Accession Number: 17649204 DOI: 10.1109/ACCESS.2018.2812844
4. A Smart Contract for Boardroom Voting with Maximum Voter Privacy (iacr.org 2017) Patrick McCorry, Siamak F. Shahandashti and Feng Hao School of Computing Science, Newcastle University UK (patrick.mccorry, siamak.shahandashti, feng.hao)@ncl.ac.uk
5. Blockchain based Smart Contract for Bidding System (13-17 April 2018) Yi-Hui Chen ; Shih-Hsin Chen ; Iuon-Chang Lin Date Added to IEEE Xplore: 25 June 2018 INSPEC Accession Number: 17862005 DOI: 10.1109/ICASI.2018.8394569

Online References:

1. <http://truffleframework.com>
2. <https://github.com/trufflesuite/ganache-cli>
3. <https://github.com/ethereum/go-ethereum>
4. <https://github.com/ethereum/solidity>
5. https://github.com/yeasy/blockchain_guide
6. <https://github.com/ConsenSys/smart-contract-best-practices>

Chapter 7

Publications of the Candidate

[1] Gananjay Thanekar, Aveesh Shetty, Sagar Sethi, Sulochana Madachane, Hasib Shaikh “Decentralized Voting Application Using Ethereum Blockchain”, International Journal of Emerging Technologies and Innovative Research (www.jetir.com), ISSN:2349-5162, Vol.6, Issue 3, Page No. 72-76, March-2019