A Decentralized E-Voting Application Based on the Ethereum Blockchain

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Sonal Saxena
(Roll No. 801732051)

Under the supervision of:
Dr. Seema Bawa
Professor (CSED)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
PATIALA – 147004

July 2019

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"***A Decentralized E-Voting Application Based on the Ethereum Blockchain***"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of *Prof. Seema Bawa* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Sonal Saxena

Signature:

Sonal Saxena

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Dr. Seema Bawa

Professor,

Computer Science and Engineering Department

ACKNOWLEDGEMENT

Due to the holy blessings of Babuji Maharaj (founder of Sri Ram Chandra Mission, Shahjahanpur, Uttar Pradesh, India), it is a moment filled with honor and humility for me to express sincere thanks to the various quarters from where I have received unflinching support during my M. E. at Thapar Institute of Engineering and Technology (TIET), Patiala, Punjab.

First and the foremost, I am elated to express my sincere thanks to my research supervisor, Prof. Seema Bawa, Department of Computer Science and Engineering, T.I.E.T, for all her efforts to raise me up from several spiral sinks of despondency. The critical comments, rendered by her during the discussions are deeply appreciated. I am elated to state that Prof. Bawa's vast spectrum of knowledge and wisdom in the field of computer science always attracts me and gives the appropriate way to think positively.

I am thankful to all my friends in TIET and around with special mention of Reaya Grewal, Sumedha Satija, Supreet Kaur Bajwa for helping me from the various prospective during my stay at TIET.

I express my profound respect to Prof. Maninder Singh, Head of Department, Department of Computer Science and Engineering, T.I.E.T. for providing me the requisite infrastructure and a research friendly atmosphere. I would also like to thank all my teachers who taught me in the first year of my M.E. and enlightened the concept of blockchain.

Lastly, I would like to thank my parents Mr. Ravi Prakash Srivastava and Mrs. Neelima Srivastava who continuously supported me in all my pursuits. My sincere regards to my in-laws Mr. Ishwar Chandra Saxena and Mrs. Sunita Saxena, uncle Mr. Shripraksah Srivastava for their support and encouragement to pursue higher education. Heartfelt thanks to my brother Ayushdeep, sister-in-law Shruti and her husband Vivek, and to my lovable nephew Viraj, respectively, for moral and emotional supports. Great thanks to my loving, supportive, encouraging, and patient husband Dr. Sahaj whose support during the final stages of this M.E. is so appreciated and he always has a trust in me to do something.

Sonal Saxena

(SONAL SAXENA)

ABSTRACT

E-voting is a trending technology. Considerable amount of work based on complex cryptographic algorithm have been introduced to design and implement the e-voting. However, there is always a need of secure, confidential, authentic and easy technology enabled voting process.

In 2008, when the digital currency Bitcoin came into existence, since then the underlying technology blockchain became the hot topic. The design of blockchain is robust in such a way that it can bear tamper-resistant data because the voluminous amount of data is increasing day-by-day, which is cryptographically connected. The feature of blockchain technology to provide a database security as a distributed ledger for financial transaction is one of its application. In the blockchain, Ethereum is one of the most popular and emerging blockchain platforms. Ethereum offers protection against the malicious users by charging fees (referred to a gas) for every execution of transaction.

This thesis focuses on the design and implementation of decentralized e-voting application using blockchain technology. The proposed application is empowered by Ethereum platform where Smart Contract is written in solidity programming language and then deployed to the local blockchain which allows accounts to vote for their desired candidate. One can interact with the contract on the blockchain through a node.js console as well as through a web page to display the vote count and votes on the page. The advantage of the proposed system over the existing technology is that the blockchain serves as a single entity of having a network and a database all in one. This technology also enables the designer to avoid complex cryptographic algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

dapp	Decentralized application
DRE	Direct recording electronic
E-voting	Electronic voting
EVM	Ethereum voting machine
IDE	Integrated development environment
PKI	Public key infrastructure
PBC	Prepare Bitcoin Cards

CHAPTER 1

INTRODUCTION

1.1 Election process

Election is an official procedure to select an individual or a party by group decision making procedure to hold public office. The security and fairness of an election in every democracy is the matter of national security. The base of voting is democracy, even though they are not free from frauds and require severe security actions [1]. Election procedures can be carried out in many ways from traditional paper ballot voting to e-voting [2]. In a layman's language, the voting through electronic devices is called e-voting. The present scenario of e-voting is a challenging task which incorporates different disciplines of engineering rising from software engineering, cryptography to the electronic hardware implementation.

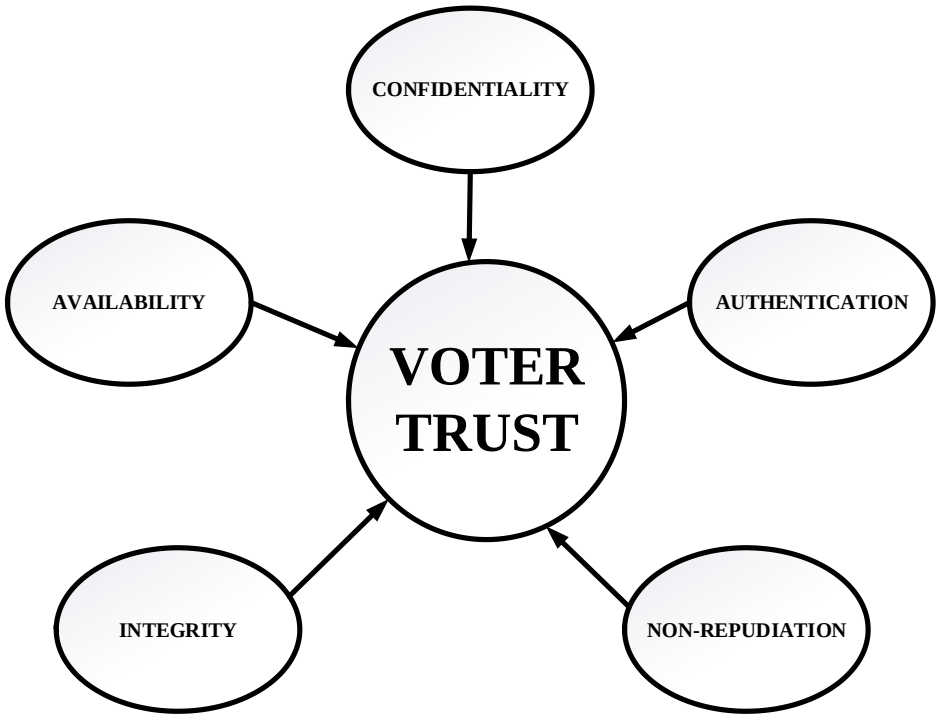

Fig. 1.1 Model representing voter's trust

While implementing the process of election, some factors are required to gain the trust of voter. As depicted in Fig. 1.1, these components are as follows:

(i) Confidentiality: It implies to protect the information against unauthorized person which means secrecy of ballot and mitigation against illegitimate users and attackers.

(ii) Authentication: It is basically a security service, which signifies the verification of the eligible voter.

(iii) Non-repudiation: It is the assertion that nobody cannot deny the validity of something. However in context of voting, it is an evidence of a voting transaction.

(iv) Integrity: Integrity is defined as the protection of information from misuse. In voting it intends to preserve data integrity of ballot information and audit records.

(v) Availability: It is defined as the accessibility of information by authorized parties, when it is required. Information only has the value if the right people have the right to access the information.

1.2 E-voting

E-voting denotes to the usage of electronic resources to carry out easy, secure, fast, economic and trustworthy voting procedure [3], [4]. Usually, e-voting system comprises of registration, authorization, authentication, casting of votes, counting of votes and verification of votes. E-voting has its own pros and cons. Some of its pros are as follows: prevention from frauds, less human interference, faster results processing, decrement of costs and remote voting. Some of the cons of e-voting which are not only technical but also methodological are as follows: paucity of transparency, threat of fraud manipulated by hackers or privileged insiders and rapid increase in the cost of voting infrastructure intend to communication technologies and power supply [5]. Table 1.1 illustrates the differences between traditional paper-ballot voting and electronic voting and reveals that the electronic voting is beneficial when compared with the traditional paper ballot voting.

Table 1.1 Difference between paper-ballot and e-voting

Characteristics	**Paper-ballot voting**	**E-voting**
Security	Easy to manipulate	Difficult to manipulate
Accuracy	Small alteration to results	No alteration, accurate results
Time consumption	Counting consumes huge amount of time	Less amount of time
Complexity	Counting is very tough (Manual)	Faster count and tabulation (Automatic)
Authenticity	Trust in poll workers	Trust in poll worker, system developers and admins
Device used	Paper, stamp and boxes	Electronic Voting Machine
Output	Impression on the paper	Single LED blinks beside each candidate's name or logo
Cost	Paper printing cost is high	Low hardware implementation and maintenance cost is low

1.3 Blockchain technology

Blockchain technology served as a good remedy to overcome the aforementioned difficulties in e-voting. It is a promising technology which is defined as a public, permanent, append-only distributed ledger to manage transaction, legal data, etc. [6]. Utilizing the characteristics of Blockchain in e-voting

(i) makes it secure and reliable,

(ii) provides better transparency because of its open and distributed ledger,

(iii) brings immutability (object cannot be modified once it is created) as well as intrinsic anonymity.

The interested readers can find the detailed study of the blockchain in [7].

1.4 Ethereum

Today, among the blockchain technologies, Ethereum is the most promising one. It was launched in 2015. It is a decentralized network of node, each node in the network executes some bytecode called Smart Contracts [8]. Ethereum is a digital currency as well as it provides a platform to develop blockchain-based application featuring the smart contracts and the Ethereum Virtual Machine (EVM). It further employs ether as a mode of currency for peer-to-peer contracts [9]. This Ethereum technology provides a base to develop decentralized applications (dapp).

1.5 Motivation of Thesis

The motivation of the work presented in the thesis is described in the following subsections.

1.5.1 Preference of blockchain application over web application

To interact with a web application over a network, the user uses a web browser to establish a connection with central server. This can be visualized in Fig. 1.2 in which the code of the web application lies in the central server and the data lies on the central database. Whenever a person performs transaction with the web application, he requires communication with the central server on the web. If a voting application is built on web, one can face the following problems.

(i) The data lies in the database may get changed or it can be completely removed or re-counted.

(ii) The relevant source code written on the web server may get altered several times.

In place of taking a network, a database and the centralized server, the blockchain serves as a single entity of having a network and a database all in one as shown in Fig 1.3. A blockchain is actually a peer-to-peer network of workstations namely nodes which shares data and relevant codes in the network [10]. Therefore, if a system wants to connect to the blockchain, it will act as a node in the network and thus can establish communication with all other nodes. At this time one can obtain a replica of the complete data and code on the blockchain. Therefore, the requirement of central servers vanishes [11]. Here, a group of computer system acts as a node talk to one another on the same network.

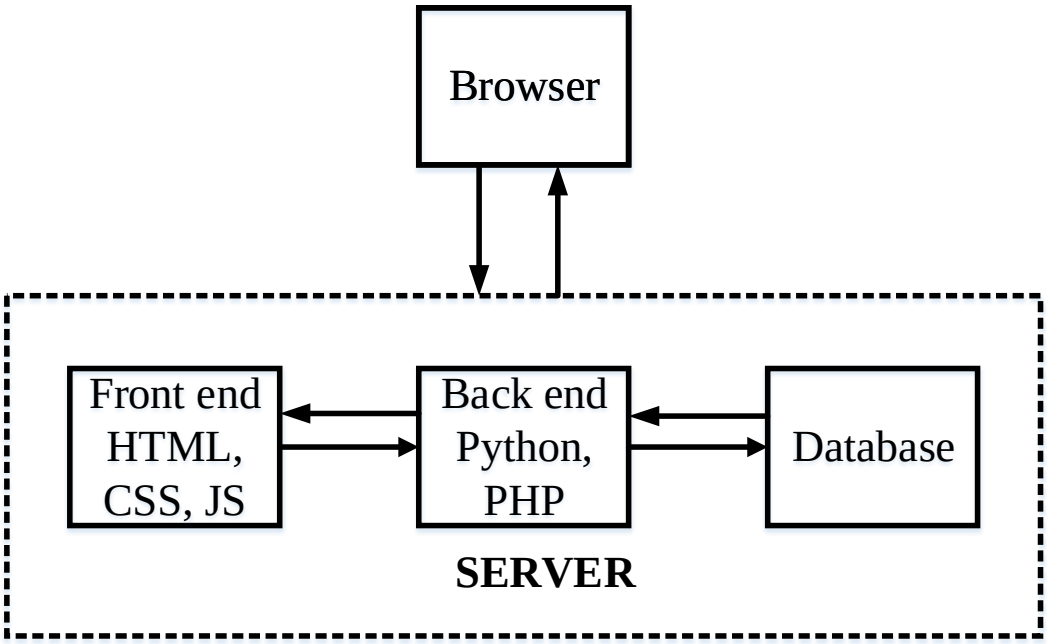

Fig. 1.2 Interaction with the web application

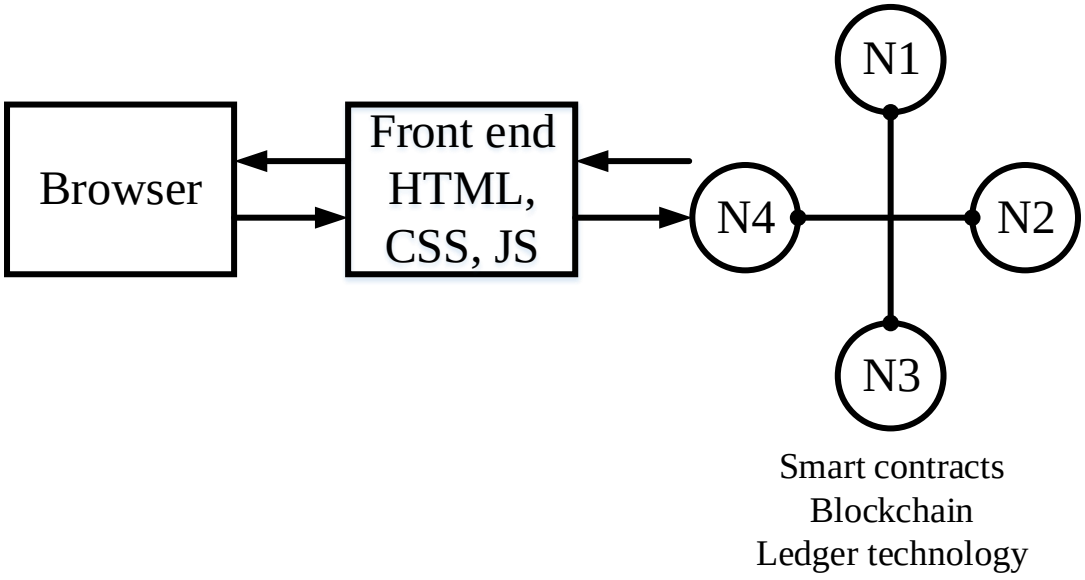

Fig 1.3 Interaction with the blockchain application

However from e-voting viewpoint, when the voter is connected to the network in blockchain, it implies that he is taking part in the process of election. Therefore, there must be surety in blockchain technology that their votes are counted and counted only once.

1.5.2 Preference of Ethereum over Bitcoin

The blockchain technology is based on the distributed network consists of numerous interconnected nodes. Each and every node of the chain has its own replica of the distributed ledger which comprises the complete history of the transaction handled by the network. The network is not controlled by the single authority. If majority of nodes approve on the particular transaction, then only they accept it. This network allows users to stay unacknowledged or anonymous [12]. The literature survey also states that the blockchain technology (including smart contracts) permits the suitability of designing e-voting. Even it possesses the power to create e-voting more acceptable and reliable. Extensive research is under progress. One can refer to [13], [14].

Earlier, Bicoin based blockchain technology has been utilized extensively for the efficient, secure, anonymous and transparent designing of e-voting system. However, there are some limitations in this technology which are as follows.

(i) E-voting based on Bitcoin technology is not a fully committed solution as it offers limited performance for designing a blockchain system [15].

(ii) It is not convenient for large number of voters. For instance as mentioned in [16], the issuance of Prepare Bitcoin Cards (PBC) is done physically in face to face communication which can be practically difficult.

(iii) There is a requirement to build a secure, transparent and complete anonymous e-voting structure without the involvement of complex cryptographic method [17].

Keeping the aforementioned things in mind, a decentralized e-voting application established on Ethereum blockchain is adapted in the present thesis.

1.6 Organization of Thesis

The organization of this thesis includes five chapters. Chapter 1 gives an overall introduction of the E-voting, Blockchain technology and Ethereum. Chapter 2 presents the related works and highlights their limitations concerning e-voting. Chapter 3 describes the problem statement for the e-voting. Chapter 4 discusses the implementation and testing of

the application. Finally, the thesis is concluded in the Chapter 5 followed by future prospects.

CHAPTER 2
LITERATURE SURVEY

This chapter presents the literature review and associated works pertaining to the application and implementation of e-voting using blockchain approach. The essential exploration of the blockchain technology recommends that it is most suitable technology to design e-voting application thereby building e-voting system from blockchain technology is more suitable and consistent [18]. Considerable amount of research (for example, see [19], [20] and references therein) has been done to explore this idea. The obvious merits of employing blockchain on e-voting are as follows:

(i) Open and distributed ledger provides greater transparency
(ii) Inherent anonymity
(iii) Safety and trustworthiness (against false data injection and denial-of-service attacks)
(iv) Immutability (to maintain robust reliability in the voting system and individual votes)

2.1 Background of blockchain and Ethereum

Blockchain is defined as a public decentralized database with copies circulated to the numerous nodes at the same time [21]. This technology is free from the authority in charge that maintains and manages the ledger of the transactions in blockchain technology. Consensus mechanism is used for establishing the verification of the ledger's version among the nodes. Blockchain technology offers a safe authentication of transaction's data integrity [22]. Furthermore, blockchain has an advantage of high availability, verifiability, and integrity makes it a powerful and secure alternative to distributed databases.

Ethereum, a new version of Blockchain, is actually an open-source, distributed and decentralized computing structure which runs the programs namely smart contracts [23]. Ethereum not only serves as digital currency but also enables decentralization for application. It is achieved by using an Ethereum Virtual Machine to perform a turing-complete scripting language. To change the state of contract in the blockchain, a charge

called the transaction fee is required. This transaction fees is valued in Ether. Moreover, Ether is treated as an energy source for working the distributed application.

2.2 Existence of e-voting

The conception of e-voting is considerably older than blockchain. Estonia is the first country whose government gave the permission to implement fully online election process [24]. Since 2001, e-voting has become the topic of debate in this country which later on implemented officially by the concerned authorities in the 2003 [25]. Their scheme is still operational with the alterations in the original one. Electronic devices such as smart digital identification cards and personal card readers are used for person-wise authentication [26]. Switzerland is also one of the few countries relying on e-voting trend. This country also started working officially on voting system called remote voting [27].

Prior to the introduction of Bitcoin, few network security based technologies were employed to implement the e-voting. Some of the major used implemented schemes are illustrated in Table 2.1.

Table 2.1 Related works on e-voting

Author	Year	Remark
Gritzalis [28]	2002	Described rational unified process for an adequate secure e-voting system
Besselaar et al. [29]	2002	Addressed the democracy oriented legal and constitutional requirement for e-voting
Lauer [30]	2004	Developed a secure public key infrastructure (PKI) based e-voting system
Mitrou et al. [31]	2008	Designed an optical scan voting system for verification that permits each voter to verify his recorded vote
Chaum et al. [32]	2014	Illustrated the comparison between the internet voting system and the direct recording electronic (DRE) voting system

2.3 E-voting based on Bitcoin

A privacy-preserving, decentralized and functional Bitcoin e-voting protocol is discussed in [33] that uses shuffling mechanism for preserving the privacy, threshold signatures to allocate voting rights, uses bitcoin transaction for transparency but number of transaction carried out by the mechanisms are minimum. A SHARVOT protocol is presented in paper [34] that practices blockchain technology to publicize and build an election. The use of multi signatures in P2SH scripts is also discussed for the efficient storage of votes. E-voting solution centered on Bitcoin protocol and blind signatures are proposed in [35]. According to Cruz and Kaji [16], Bitcoin protocol will be efficiently used in other different security application rather than e-voting. Ethereum based protocol could be investigated for the practical implementation of e-voting system [36].

2.4 E-voting based on Ethereum

Various solutions and researches are carried out to incorporate to decentralize the voting system while implementing e-voting using blockchain. Yavuz et al. [37] discussed a decentralized voting solution using Ethereum blockchain. The authors also suggested to deploy the application of e-voting as a smart contract and allows user with effective EOAs to cast vote on the contract but this type of solution has limitation of true automated address verification protocol as described by [38].

Tarasov and Tewari [39] present the potential use of blockchain in e-voting and proposes registration phase to confirm the user's characteristics. The process of verification is completed by utilizing a Challenge-Handshake Authentication Protocol (CHAP).

Messer [40] depicts two kinds of current problems related to e-voting approaches. The foremost one is the ability of any person to note the results before process of casting of vote completed and the second one is the privacy of the votes subsequently the public keys can be connected to the documented votes.

McCorry et al [14] claims that anyone can check the reliability of the election by ignoring the trust specialists and by preserving the voter's privacy. They used Open Vote network (OV-net), Direct Recording Electronic with integrity (DRE-i) to achieve the objective of verifying the integrity of election [41].

Linh et al [2] suggested Votereum which is an Ethereum based e-voting system which employed blockchain technology and smart contract to allow an open, safe election. Apart from this, the privacy of voters is also preserved by using the approach proposed by [42]. Primary necessities of ballot confidentiality, distinctiveness, verifiability and robustness are also discussed in this paper.

2.5 Research Gaps

There are some research gaps which are prevalent in voting process whether it is implemented by paper-ballot technique or e-voting technique. Researches gaps exist in the prevailing technologies are given below:

(i) The electronic voting system used nowadays are not perfect, likely to face issues such as authentication, secrecy or data integrity [1].

(ii) An e-voting has to be done in an easy and democratic way [2].

(iii) There is a requirement of transparency and verifiability in the existing electronic-voting machine [59].

(iv) Centralized database is vulnerable to attackers and hackers.

2.6 Prerequisites

This section presents some definitions which are useful for designing and implementing the decentralized e-voting application based on Ethereum.

2.6.1 Ethereum

Ethereum is a decentralized network of node, each node in the network executes some bytecode called smart contracts [43]. Ethereum is not only a digital currency but it also provides a platform to develop blockchain-based application featuring the smart contracts, the Ethereum Virtual Machine and practices ether (currency) for peer-to-peer contracts [44].

2.6.2 Smart contracts

It is basically a computer code that executes on the top of a blockchain having a set of rules under which the parties to that smart contract agree to interact with each other. It provides some properties to the platform on which it executed such as immutability, deterministic operation and distributed and verifiable state, etc. Smart contract is the micro-service on the web which can be accessible by everyone on the blockchain. In blockchain, the public ledger denotes the database layer and the smart contract is a place for all the business logic that manages with data [44], [45].

In the proposed e-voting dapp, smart contract is an agreement in which the vote is counted once at a time and then the candidate having maximum votes emerge victorious in the election.

2.6.3 Ethereum Virtual Machine

EVM is the Ethereum smart contracts bytecode execution environment. Each node present in the network executes EVM. The EVM executing on the Ethereum blockchain network is considered as a global decentralized computer having many executable objects containing its own permanent data storage. The EVM confirms that programs do not require any access to other's state by establishing communication without other possible intervention [46].

2.6.4 Gas

It is a unit with in itself not equal to Ethereum that charges money to communicate with the blockchain. Also, this currency drives some of the nodes on the network that performs all the required task to incorporate the code in the blockchain. The user has to mention the money which he is eager to wage to obtain his code incorporated in the blockchain. In order to get the Ethereum amount the user must have to multiply the gas usage by gas price [47].

2.6.5 Gas Price

Every unit of gas has its value related with it [48].

2.7 Tools Used

Following tools or dependencies are required to built Ethereum based decentralized e-voting application.

2.7.1 Node Package Manager (NPM)

Node Package Manager (NPM) is the dependency which is to be installed first. It comes with Node.js. Node.js enables NPM to compose applications in Javascript on the server [49]. One can easily verify that node has already been obtained by entering into the terminal and type: > node – v.

2.7.2 Truffle Framework

Truffle Framework permits to construct decentralized applications using the technology of Ethereum. Further, this offers a set of instruments that allows the user to compose the smart contacts in the Solidity programming language. Moreover, this language permits the user to check the smart contracts and organize them to the blockchain. Moreover, it helps the designer to provide a space to advance the client-side application [50].

One can easily mount the Truffle Framework with NPM in the command line by typing: > npm install –g truffle

2.7.3 Ganache

Ganache is basically a local in-memory blockchain. It can be downloaded from the Truffle Framework website. Ganache provides external accounts (upto 10) having addresses on the local Ethereum blockchain. Every account has 100 fake ethers in advance or preloaded. Basically it is a personal blockchain that can be run for development purpose locally [51].

2.7.4 MetaMask

MetaMask extension is the next dependency for Google Chrome. Therefore, to employ the blockchain, the user should link to it (Blockchain **is** a network). To utilize Ethereum blockchain, a special browser extension is required, i.e., where the metamask comes in. One can set connection to his local Ethereum blockchain with his personal account, and intermingle with the smart contract. Now, when it is successfully installed on the system the fox icon will appear on the top right hand side of the Chrome browser [52], [53].

2.7.5 Solidity (Programming language)

Solidity is defined as a contract-oriented high level language for executing smart contracts that get applied on the EVM. Motivated by C++, Python, JS, Gavin Wood and his team developed Solidity to design the target EVM. Many text editors and integrated development environments lack syntax highlighting. However, the user needs a package to support for Solidity. Here, Sublime Text is used. Further, the "Ethereum" package is downloaded which offers a nice syntax highlighting for Solidity [54]-[56].

CHAPTER 3

PROBLEM DESCRIPTION

This chapter discusses the problem description and the proposed approach using Ethereum blockchain and preliminary components to design and implement the e-voting application.

3.1 Problem statement

The novelty of the blockchain as a database technology gives rise to the stipulation for easy understandable and illustrative information which bridges the gap between technical source code and developers and the outside ecosystems and users. Even now the e-voting is relatively an unexplored field in spite of number of publications [57]. The aim of the thesis is to solve the following given issues.

(i) The electronic voting systems employed these days are not best which may results into various problems such as confirmation, confidentiality or data reliability [58].

(ii) Necessities of e-voting like transparency and verifiability has yet to be achieved [59].

(iii) The equipment like electronic voting machine can easily be destroyed physically thereby disturbing the ballots just like the case of traditional voting [60].

(iv) The data lies in on the central database may be changed or it can be completely removed or counted more than once.

3.2 Research approach

In order to solve the problem discussed in aforementioned section for designing voting application, the Ethereum based blockchain technology is proposed here. Our research approach is grounded on the smart contracts, a micro service on the web accessible to everyone on blockchain. The proposed scheme is motivated by the work done in [61]. The proposed scheme is described as follows:

(i) *Setting the development environment:* For developing the decentralized application (dapp), an in memory blockchain is used which is named as Ganache. Ganache comes up with 10 test accounts with preloaded 100 fake ethers.

(ii) *Writing the contract, compiling and deploying it on the development environment:* Smart contract is written by using Solidity programming language (an object-oriented programming). Note that all the logic of the dapp is contained in this smart contract. The code or logic written in contract allows listing the candidates to be executed in the election process, keeping the record of all the votes and voters. It directs the rule of election to vote only once. The name of contract file created for the application is Election.sol and web3 function is employed to deploy the contract on the blockchain.

(iii) *Interacting with contract on blockchain via node.js console:* Some commands will be written in node console to observe the vote count addition. To vote for a candidate, every time a transaction id is back on the screen as a result. This transaction id is the verification which states that the transaction happened successfully. This id is immutable. It is one of the benefits of using Ethereum blockchain.

(iv) *Interacting with contract through a web page to exhibit vote counts and votes for the candidates on the page:* In this part, an html file is created mentioning candidates' name, vote count and invoking the voting commands (which is verified in the node.js console) in a JavaScript (.js) file. Both html and JavaScript file has been placed in the election directory.

Based on the aforementioned points, the overall schematic diagram of the aforementioned research approach is presented in Fig. 3.1.

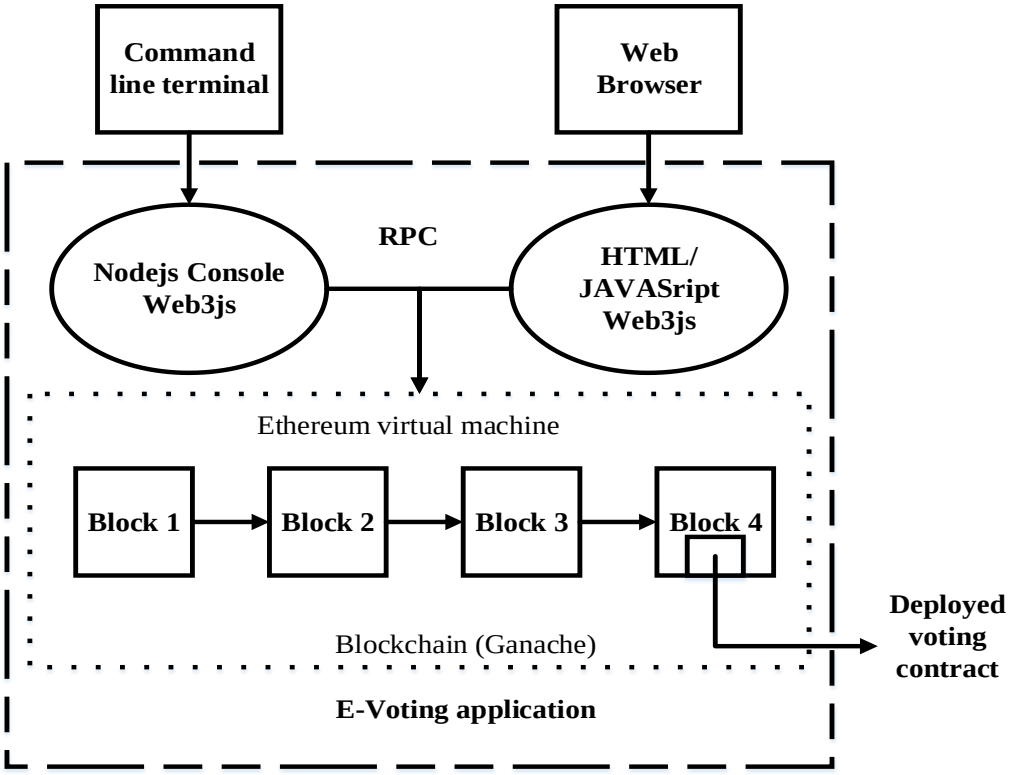

Fig 3.1 Schematic diagram to represent the designing of e-voting application

CHAPTER 4

IMPLEMENTATION AND TESTING

This chapter presents designing, implementing and testing procedure of the proposed e-voting dapp. The working model of proposed e-voting application based on Ethereum blockchain is employed and run on the network of blockchain. The complete step-by-step procedure has been explained in the following sections.

4.1 Step 1: Installation of dependencies

First and the foremost step is to install the following dependencies:

(i) Node Package Manager (NPM)
(ii) Truffle Framework
(iii) Ganache
(iv) Metamask
(v) Syntax Highlighting (optional)

4.2 Step 2: Ganache blockchain

Open the downloaded Ganache blockchain. Now the Ganache has booted and the local blockchain start running. Fig 4.1 shows the booted blockchain containing accounts with ethers. Ganache contains 10 accounts, each preloaded with 100 fake ethers that do not have any value in the Ethereum network. Note that every account contains a unique address and its private key. The address of every account will act as a distinctive identifier for every voter in the election. Afterwards, the project directory of the dapp is created line by typing in the command:

> mkdir election
>cd election

After entering inside the project, run a Truffle box. Truffle Pet Shop box, as shown in Fig. 4.2, is used to proposed this dapp. Table 4.1 illustrates the directories and their functionalities contains in the framework. A pet shop box is installed within the project from the command line by typing the given below command:

> truffle unbox pet-shop

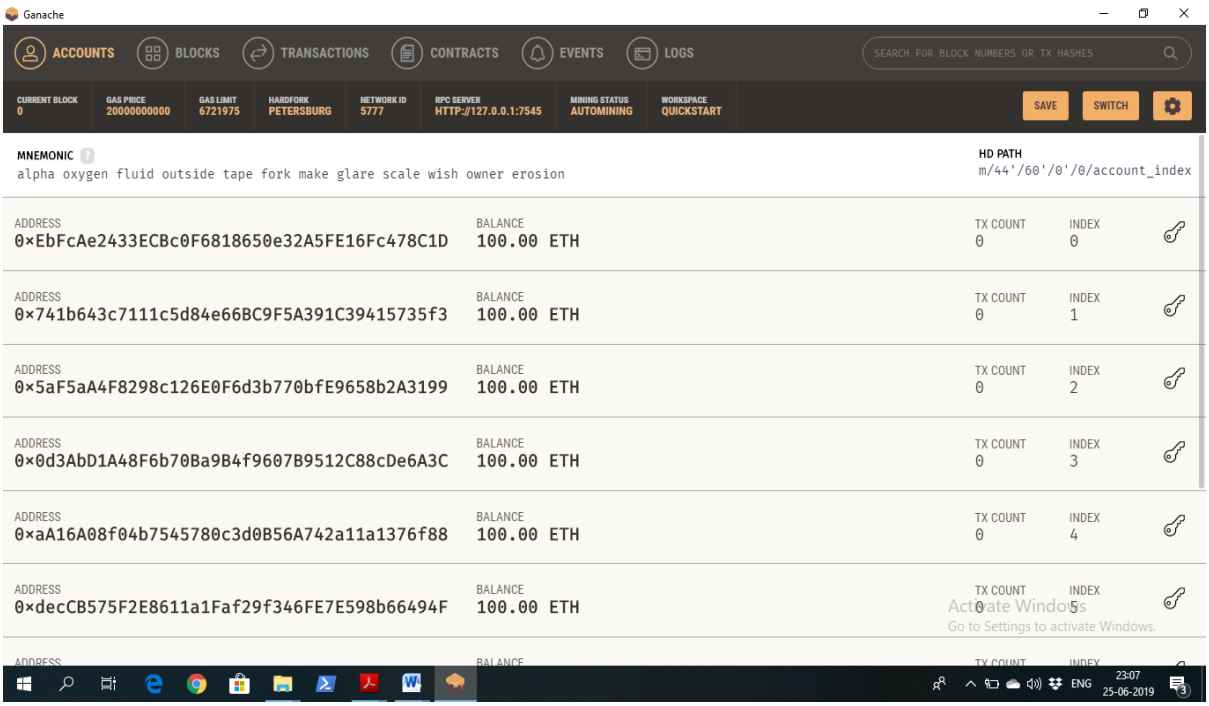

Fig. 4.1 Blockchain with accounts

Table 4.1 Illustration of the functionalities of the directories in the Truffle framework

Directories/Files	Functionalities
Contracts directory	Contains all the live smart contracts
Migrations directory	Comprises of live migration files which is required to set up the smart contracts to the blockchain
Node modules directory	Acts as a home for all the node dependencies
Test directory	Tests for smart contracts are written here
Truffle.js file	Main configurations file for the Truffle project

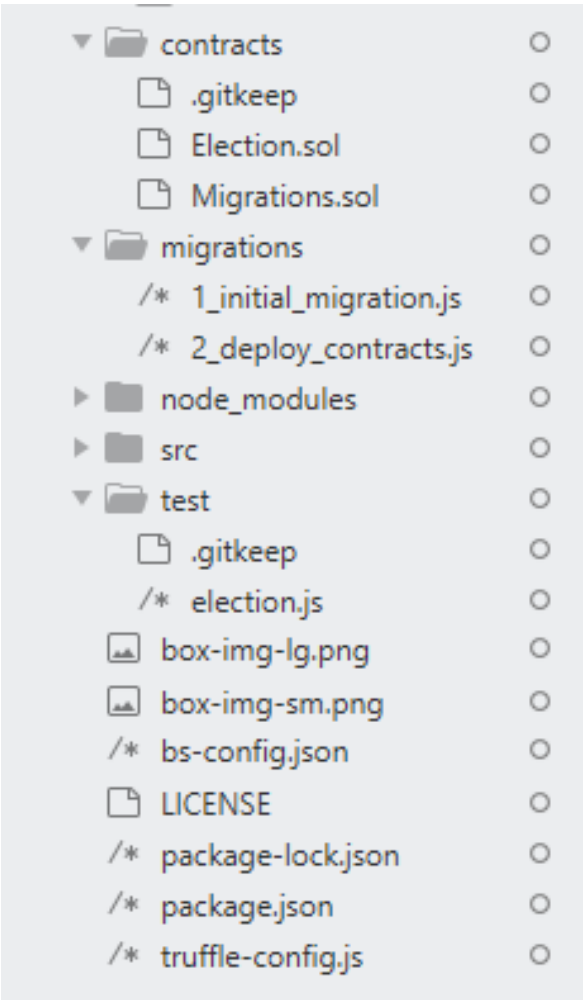

Fig. 4.2 Entities of Pet shop box

Then the smart contract is written. Smart contract will comprise all the logics of the decentralized application which allows the listing of the candidates required to execute in the election process, keeping the record of the votes and voters. Smart contract direct the guidelines of election such as imposing accounts to cast single vote. First of all, a new contract file in the contracts directory has been created in following way:

>touch contracts/Election.sol

Then a test has been done to ensure that the set-up of the project is working properly, after that only the contract will deploy on the blockchain successfully. First the Election.sol file is opened and the code is written. This code declares the version of the solidity with the pragma solidity statement. Then the smart contract is declared by the “contract” keyword and the contract name (i.e., Election) in the following manner:

```
Pragma solidity 0.5.0
Contract Election{
}
```

Next, a state variable is declared which stores the value of the candidate name. This state variable allows writing data on the blockchain. It is defined as a string and its visibility is fixed to public. Because it is public, it allows accessing this value outside of the contract.

Then a constructor function is created to deploy the contract on the blockchain. It is very important to have the same name of constructor function as the smart contract. In this way the solidity recognizes the function is a constructor. To install the contract to the blockchain, a new file is generated in the migrations directory through the command line by typing:

```
> touch migration/2_deploy_contracts.js
```

Files inside the migration directory have to be numbered so that Truffle identifies the order to implement them. The code is needed to build a new migration to deploy the contract. Now the migrations are executed from the command line by typing:

```
> truffle migrate
```

In this way, the smart contract gets successfully migrated to the local Ethereum blockchain. To interact with the smart contract a truffle console is opened from command line by typing:

```
> truffle console
```

Now inside the console, getting an instance of the deployed contract, one can recite the name of candidate from the contract by executing the code from the console:

```
Election.deployed(), then(function(instance) { app = instance })
```

Now, in migration file, the variable name is generated under the name of election. A deployed instance of the contract is retrieved with the deployed() function, then it is assign to an app variable within the callback function. Now one can recite the value of the candidate variable in following manner:

```
app.candidate()
```

Till now the smart contract is written and successfully installed to the blockchain and some of its data is retrieved.

4.3 Step 3: Listing of the candidates

After setting everything properly, continue creating the smart contract by listing out the candidate which will get executed in the election. The modeling of the candidates has been done by using struct keyword to maintain the record of the multiple candidates' id, name and vote count.

The candidates are modeled with a Solidity Struct. The Struct specified an id and vote count of unsigned integer type and name of string type. Next, a place is needed to store the candidates. It can be done by mapping. In Solidity, the mapping is an associative array that links key value pairs. A mapping has been done in the following way:
mapping(**uint** => Candidate) **public** candidates;

An unsigned integer is the key to the mapping and the value is a Candidate struct type which gives an id dependent look up of every candidate. The mapping is allocated to a state variable, anytime when data is written to the blockchain; a pair of new key-value is allocated to it. At last mapping's visibility is set to be public to get a getter function as done in the aforementioned test.

A code is required to keep the track of figure of candidates existing in the election with a counter cache state variable. A counter cache is used, because there is not any approach to define the size of mapping in Solidity and not any approach to iterate over it. A function is created to include candidates to the mapping in a way:

```
function addCandidate (string _name) private {

}
```

The function addCandidate_declares that it uses an argument of string type which denotes the name of candidate. In this function, when a new candidate is added, a candidate counter cache is incremented by 1. Then the mapping using a new Candidate struct is updated by employing the present candidate count as the key. The Candidate struct is initialized by the candidate id from the present candidate count. Next the forename from the function

argument and the initial vote count is set to zero. The visibility of the function has to be private because it is to be called within the contract.

Now the two candidates are added to the election by calling the “addCandidate” function two times within the constructor function. Next, the contract has been migrated like this:
> truffle migrate --reset

Some tests will be written to confirm that the smart contract is initialized correctly. Testing is very essential when the smart contracts are developing. There are following reasons to ensure that contracts are bug free.

(i) Ethereum blockchain code is immutable, i.e., unaltered. Even if the contract has bugs inside it, it has to be disabled and a new copy is deployed. The new copy does not contain the same state as the older contract having different address.

(ii) Deploying contracts comprises to cost a gas as it generates a transaction and writing data to the blockchain charges Ether and the amount of ether paid has to be minimized.

(iii) If any of the contracts contains bugs write to the blockchain, the account which calls the function got potentially waste the ether.

4.4 Testing

To write the test, run the Ganache in the background. A fresh test file is created in the command line from the root of the project in a way:
> touch test/election.js

All the tests have been coded in JavaScript to simulate the client-side interaction with the smart contract, as in the console. In this test, the contract is assigned to a variable as done in the migration file. Then the “contract” function is called and all the tests have been written inside the callback function. The callback function gives an “accounts” variable which denotes all the accounts in the Ganache. The First test is done to check that contract is set with the right number of candidates by testing that candidate count will be two. Then the test is done to inspect the value of each candidate in the election, confirming that every

candidate has the true id, correct name and then casted vote is count. The test is executed though the command line by typing:

> truffle test

4.5 Client-side application

Client-side application is built to talk to smart contract. It is built by altering the HTML and JavaScript files that comes up with the Truffle framework. The code for the client-side application is written in "index.html" file and in "app.js" file. The code in "index.html" file and in "app.js"file does the following:

(i) *Setting up web3*: It is defined as a JavaScript library that permits client-side application to chat with the blockchain. web3 is configured within the "initWeb3" function.

(ii) *Initialization of Contracts*: The installed instance of the smart contract is fetched from this function and allocates values to permit user to interact with it.

(iii) *Render function*: This function puts all the contents on the page including data from the smart contract. Then the candidates are listed and generated inside the contract by encompassing over each candidate in the mapping and render it to the table. Current account is fetched which is associated to the blockchain inside the function which is then exhibited on the web page.

Firstly, the contract is migrated to view the client-side application on the browser. It is migrated by typing the following in the command line:

> truffle migrate --reset

Fig. 4.3 shows the successful initialization of migration and deployment of contracts. Next, the development server is started from command line like this:

> npm run dev

```
C:\Users\user>cd election

C:\Users\user\election>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:    0xa93bda6376df04b8b2a26170df66ff6a726e51ebfcb9892e1bc3c2c387d1dd3e
   > Blocks: 0            Seconds: 0
   > contract address:    0xE4d50Ba558eadd64E0731f837a301E4826301C23
   > account:             0xC820a84E5e1d251353752CB7Cc01b0171Da2674D
   > balance:             99.61069766
   > gas used:            284908
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00569816 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00569816 ETH
```

(a)

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

   Replacing 'Election'
   --------------------
   > transaction hash:    0xba786b405f35752ae25bc017b34fc6739248b1ad2931433b22496193969c3546
   > Blocks: 0            Seconds: 0
   > contract address:    0x4617C5Ed89DFBD812E221F664eb4D7E45A855829
   > account:             0xC820a84E5e1d251353752CB7Cc01b0171Da2674D
   > balance:             99.60019108
   > gas used:            483295
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0096659 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0096659 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.01536406 ETH


C:\Users\user\election>
```

Fig. 4.3 (a) initialization of migration and (b) deployment of contracts

A new window with the client-side application is opened which is shown in Fig. 4.4. This figure further shows that application is loading because user does not projected in to the blockchain. One has to import the account from Ganache into the metamask to connect to

the blockchain from browser. After connected with the metamask, the entire contract and the account data is loaded as shown in the Fig. 4.5.

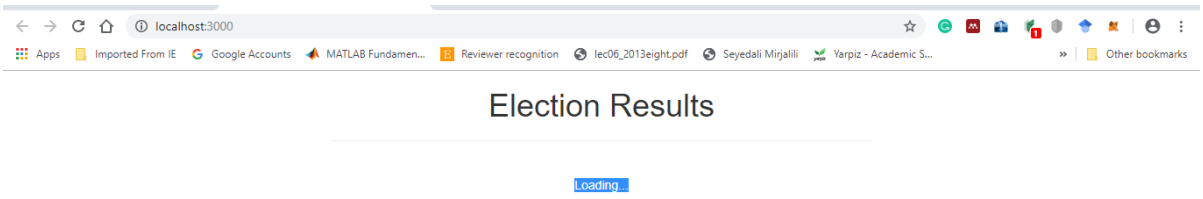

Fig. 4.4 Browser window with the client-side application

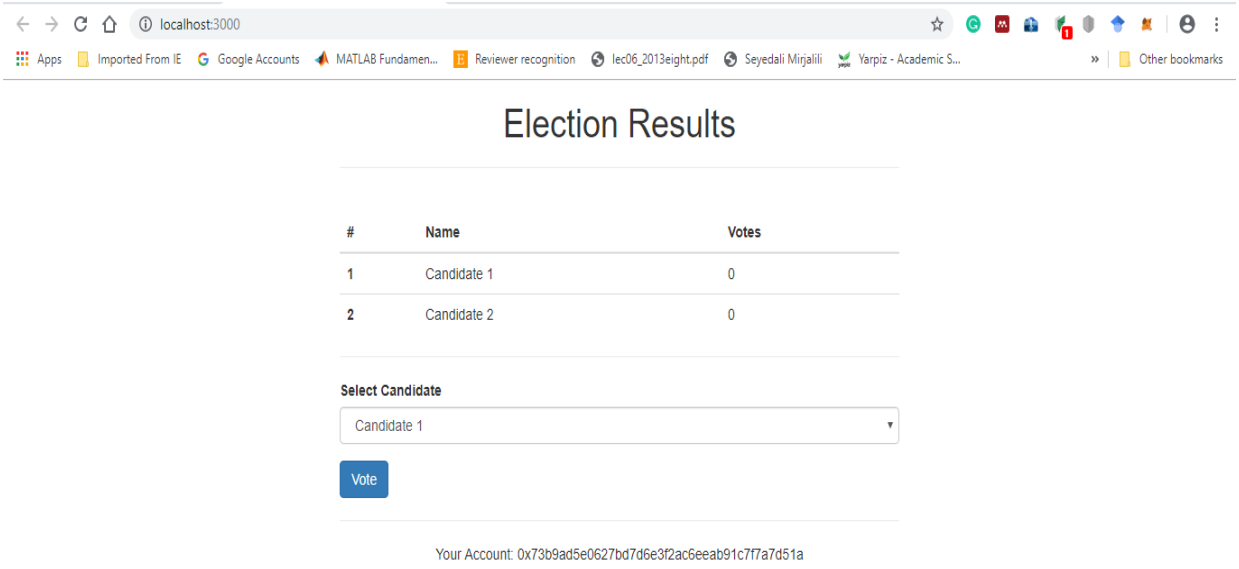

Fig. 4.5 Browser with loaded account data

4.6 Step 4: Casting of votes

To add the ability to cast a vote, a “voters” mapping is defined to maintain record of those accounts which have already voted in the election and a “vote” function is added by the code.

The functionality of the “vote” function is to increment the vote count by 1 on reading the Candidate Struct available on the “candidatures” mapping with the increment operator (++). The functionality illustrates the following points:

(i) It will accept only single argument which is an unsigned integer having candidate’s id.

(ii) The visibility is set to be public, so that external account calls it.

(iii) It allows keeping the track of the voter that he has voted in the election or not.

(iv) It also implements the require statements which stop the execution when the conditions are not satisfied. Main requirement is that the voter has not voted before. It can be done by analyzing the account address with “message.sender” variable from the mapping. In the case, if the account has previously voted, then it needs to check that candidate id is valid or not. The candidate id should be either greater than zero or less than or equal to the total candidate count.

The whole contract code is shown in Fig. 4.6 below.

```
pragma solidity 0.5.0;

contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }
    mapping(address => bool) public voters;
    mapping(uint => Candidate) public candidates;
     uint public candidatesCount;
    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(!voters[msg.sender]);
         require(_candidateId > 0 && _candidateId <= candidatesCount);
        voters[msg.sender] = true;
        candidates[_candidateId].voteCount ++;
         emit votedEvent(_candidateId);

    }

}
```

Fig. 4.6 Complete contract code

Now testing of the Voting function is done. The code has been written to test the functions. Therefore, supplement a test code to the “election.js” test file. Following two things has been tested in the test code:

(i) The function increases the vote counts for the candidate.

(ii) Whenever the voter is voted, he must be added to the mapping.

Next, the test has been written to confirm that vote function throws an exception for voting twice. The failed transaction and an error message can be asserted. The error message can be cracked to ensure that it contains the “revert” substring. Then in this way it is confirm that contract’s state was unchanged by confirming that the candidates has not accepted vote. The test code is written to prevent the double voting. Now for executing the test, type following in the command line:

> truffle test

This truffle test gives the following as shown in the Fig. 4.7.

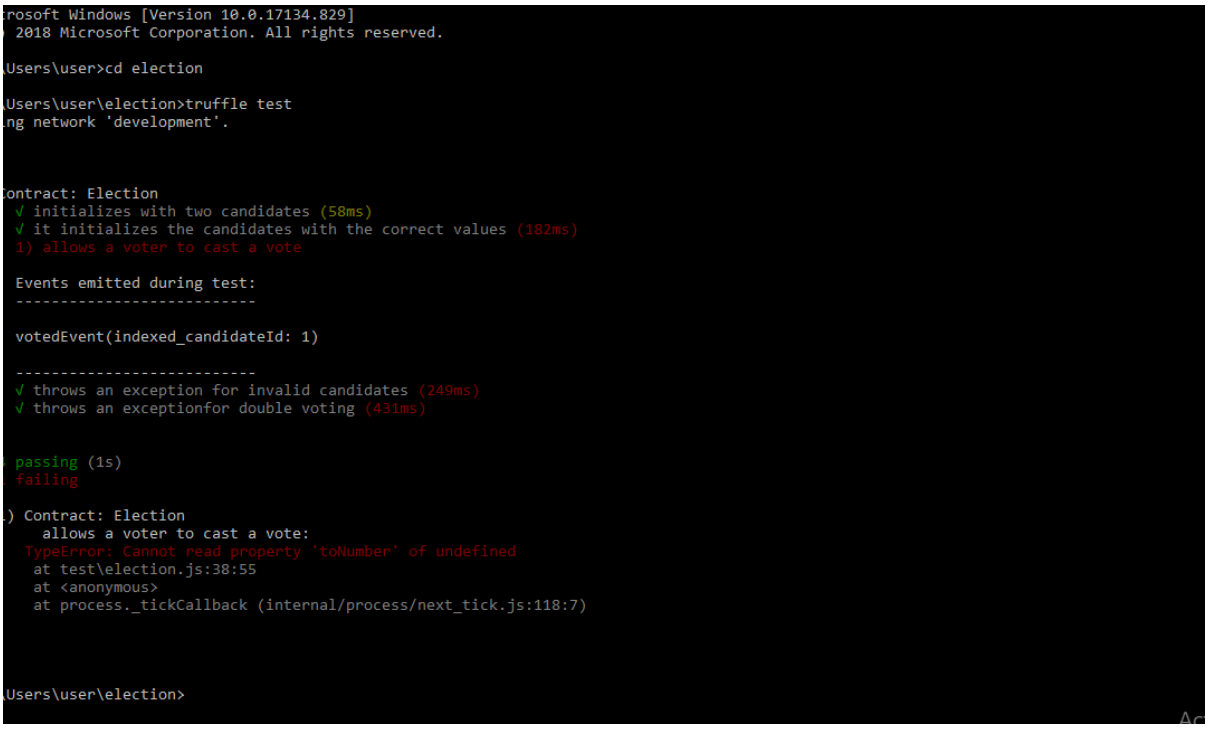

Fig. 4.7 Truffle test

4.7 Client-side voting

Insert a form which permits accounts to cast the vote. Now, the code has been updated to "index.html" file. An updated code in "index.html" file has following changes in the form:

(i) The form is generated by an empty select element. This select option is populate with the candidates given by the smart contract in the "app.js" file.

(ii) "onSubmit" handler is used to call the "castVote" function which will describe in the "app.js" file.

To list every candidate from the smart contract within the form's select option and hide the form from web the page when the account has voted, "app.js" file has to be updated. Render function has been updated by the code. Next, the function is written which is called when the form is submitted. Now, the front-end application will be generated as shown in Fig. 4.8.

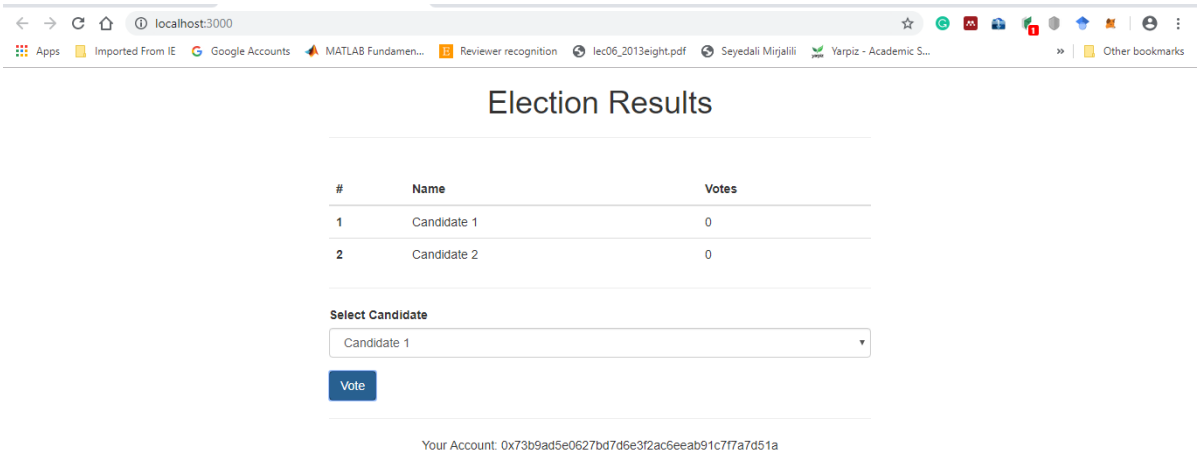

Fig. 4.8 Front-end application

When the voting function is run, a metamask confirmation pops up which is depicted in Fig. 4.9.

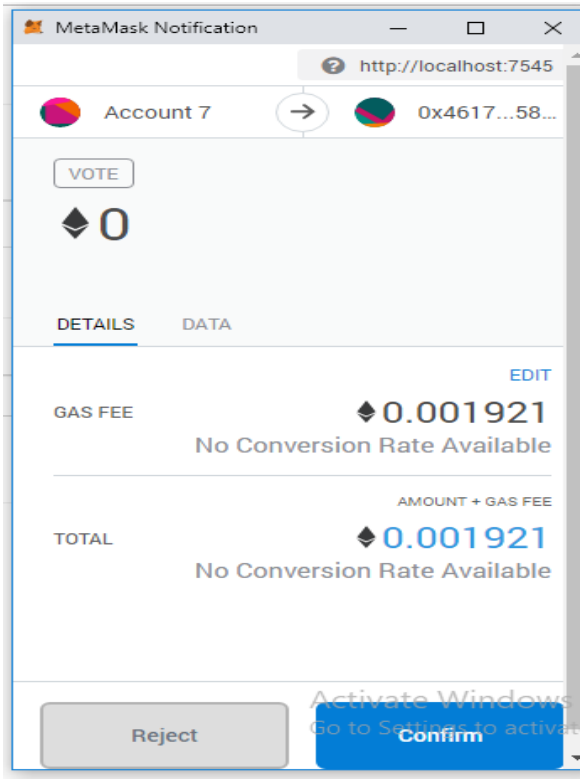

Fig. 4.9 MetaMask confirmation

When the voter clicks submit, a vote is casted successfully. If a loading screen is still appearing on the screen, he has to restore the page to see the recorded votes. The functionality to update the loader automatically is discussed in Step 5.

4.8 Step 5: Triggering of an event

The last step is to activate an event when a vote is cast, when account has voted it allows modifying the client-side application. It is done by declaring an event in the contract. The event is declared in a following way:

event votedEvent (
uint indexed _candidateId
);

Now, the "voted" event is triggered inside the "vote" function. After updating the contract, migrations has to be run from command line by typing:
> truffle migrate --reset

Tests are also updated to ensure the voting event. The screenshot of the updated test code is depicted in Fig. 4.10.

```
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");
    })
});
```

Fig. 4.10 Updated test code

The above test has been done to ensure that the transaction receipt return by the "vote" function has records. These records contain the triggered events. Now the client-side application is also updated and fires a refresh page anytime when it is triggered. The code uses "ListenForEvents" function for this. The above code does following things:

(i) The voted event is subscribed by calling the "votedEvent" function.

(ii) Then some metadata is passed which expresses to specify all the events on the blockchain and watch the event.

(iii) Log in to the console anytime, when a "votedEvent" is activated. All the contents of the page are re-render and after the vote has been recorded it results to get free of the loader. The updated vote count has been shown on the table.

Lastly, the function is called when we initialize the contract in the following way:

initContract: **function**() {

}

Now, one can successfully vote from client-side application and lookout the votes recorded in real time.

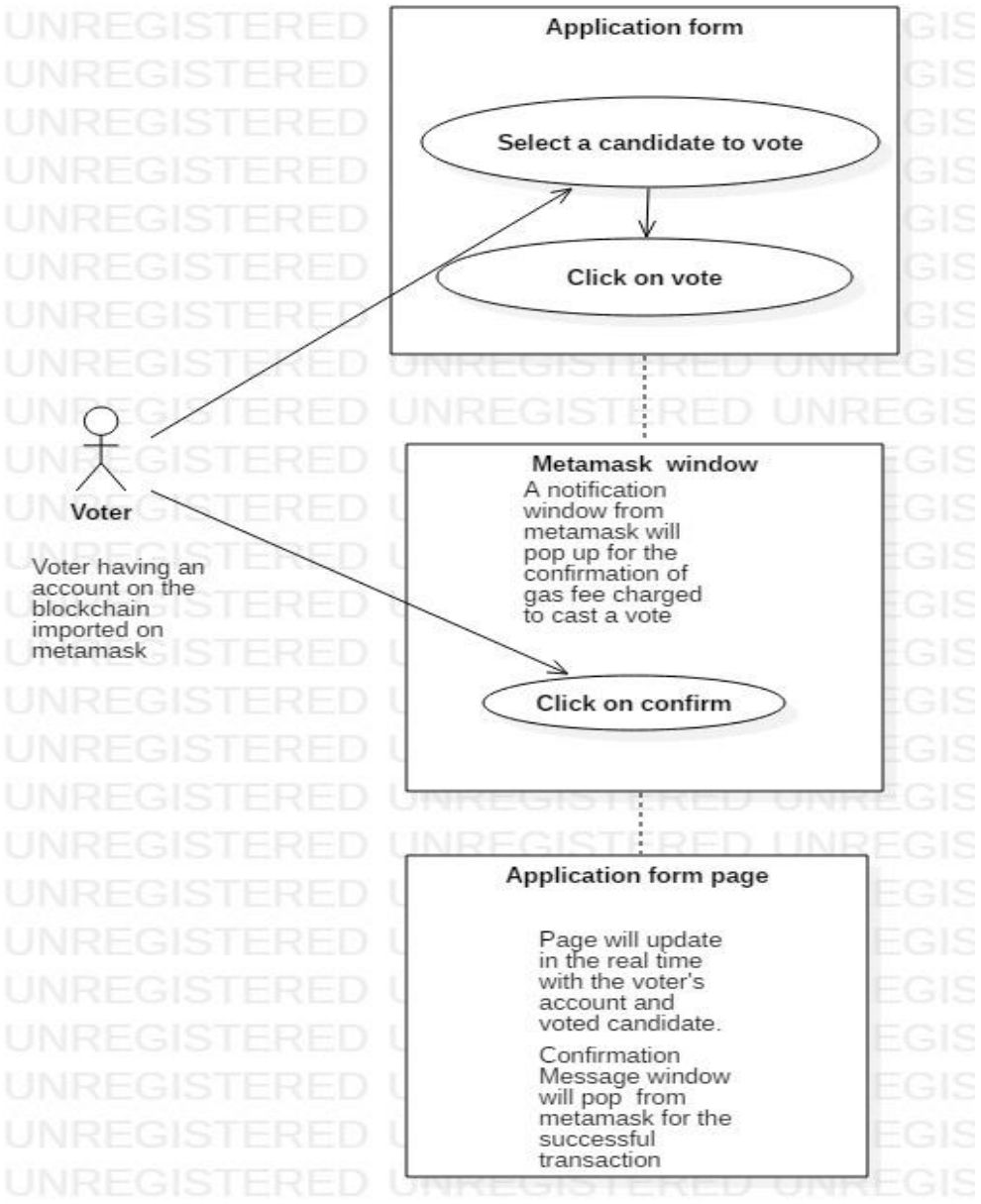

Fig. 4.11 Use-case diagram of the application

4.9 Use-Case description of the application

The user has an account containing a wallet with some ether in it. When the user connected to the network, he casts his vote and gives a transaction fee to write the transaction on the blockchain. The transaction fee referred to the “gas”. When the vote is cast, the few nodes of the network participate to accomplish the transaction. Then the node which completes its transaction is awarded with the Ether that the user paid fee to vote.

It is essential to note that to read data from the blockchain is free however write data to blockchain charges cost. When a user votes, he has to pay a gas price and when vote get recorded, one of the available nodes on the network get rewarded by user's Ether fee. Basically it's a confirmation that vote has recorded correctly. The use case diagram of the application from the voter or user's point of view is completely shown in Fig. 4.11.

CHAPTER 5
CONCLUSIONS

5.1 Conclusions

This thesis explores the potential and the efficacy of utilizing the technology of blockchain in designing and implementing the decentralized e-voting application. It is well evident that the e-voting is one of the fascinating applications for the current generation of technology experts. The goal of building an e-voting application using blockchain technology is to make the election procedure secure, transparent, faster, stress-free and safer. It also results in reduction of conducting a traditional election process. The main objective is to eliminate the prevailing threats concerning to the present traditional voting scheme. To establish the trust between voters and the election authority is the prime concern while developing e-voting concepts and technology.

5.2 Summary of contributions

The contributions of the thesis are summarized as below:

(i) A decentralized e-voting application using Ethereum blockchain tools is proposed to fulfill the important e-voting attributes. It delivers an amount of decentralization and places as much as possible in control of voter's hand.

(ii) Smart contracts has been written and deployed on the Ethereum network which addresses practically all the security concerns such as confidentiality of voters, reliability and authentication of votes and transparency of counting.

(iii) This e-voting application based on Ethereum blockchain leads to an open, faster, secure election along with voter's privacy.

5.3 Future research

There are some properties which are still not addressed solely by using blockchain technology such as:

(i) Verification of voters which requires further mechanisms to be integrated.

(ii) Besides e-voting, it will be interesting to investigate utilizations of the Ethereum for different security applications.

However, a portion of future work can be done to address the above mentioned issues. Moreover, there is a need of an intensive effort in the domain of blockchain research to develop the robust attributes and maintenance for difficult uses.

REFERENCES

[1] M. Pawlak, A. Poniszewska-Marańda, and N. Kryvinska, "Towards the intelligent agents for blockchain e-voting system," *Procedia Computer Science,* vol. 141, pp. 239-246, 2018.

[2] Vo-Cao-Thuy Linh, et al., "Votereum: an ethereum-based e-voting system," in Proc. *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF),* 2019.

[3] J. Willemson, "Bits or paper: Which should get to carry your vote?," *Journal of Information Security and Applications*, vol. 38, pp. 124–131, 2018.

[4] N. Braun Binder, A. Driza, R. Krimmer, U. Serdlt and P. Vinkel, "Focus on E-Voting", *ACE Electoral Knowledge Network,* 2014. [Online]. Available: http://aceproject.org/ace-en/focus/e-voting/about.

[5] R. M. Alvarez and T. E. Hall, *Electronic elections: The perils and promises of digital democracy,* Princeton University Press, 2010.

[6] M. Zhang and Y. Ji, "Blockchain for healthcare records: a data perspective," *PeerJ Preprints*, vol. 6, 2018.

[7] D. Drescher, *Blockchain basics*. Berkeley, CA: press, 2017.

[8] H. Diedrich, *Ethereum: blockchains, digital assets, smart contracts, decentralized autonomous organizations*, Sydney: Wildfire Publishing, 2016.

[9] A. M. Antonopoulos, and G. Wood, *Mastering ethereum: building smart contracts and dapps*, O'Reilly Media, 2018.

[10] Z. Zheng, et al, "An overview of blockchain technology: architecture, consensus, and future trends," in Proc. *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017.

[11] X. Xu et al., "A taxonomy of blockchain-based systems for architecture design," in Proc. *2017 IEEE International Conference on Software Architecture (ICSA)*, 2017.

[12] F. S. Hardwick, "E-voting with blockchain: an e-voting protocol with decentralisation and voter privacy," in Proc. *2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, 2018.

[13] T. Moura and A. Gomes, “Blockchain voting and its effects on election transparency and voter confidence,” in Proc. *18th Annual International Conference on Digital Government Research*, New York, USA, pp. 574–575, 2017.

[14] P. McCorry, S. F. Shahandashti, and F. Hao, “A smart contract for boardroom voting with maximum voter privacy,” in Proc. *International Conference on Financial Cryptography and Data Security*, pp. 357–375, 2017.

[15] Z. Bao, B. Wang, and W. Shi, “A privacy-preserving, decentralized and functional Bitcoin e-voting protocol,” in Proc. *2018 IEEE Smart World, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, 2018.*

[16] P. C. Jason and K. Yuichi, “E-voting system based on the Bitcoin protocol and blind signatures,” *IPSJ SIJ Technical Report*, vol. 2016-MPS-107, no. 7, 2017.

[17] S. Bartolucci, P. Bernat, and D. Joseph, “SHARVOT: secret SHARe-based voting on the blockchain,” in Proc. *1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, 2018.

[18] A. B. Ayed, “A conceptual secure blockchain-based electronic voting system,” *International Journal of Network Security & Its Applications*, vol. 9, no, 3, pp. 01-09, 2017.

[19] N. Kshetri, and J. Voas, “Blockchain-enabled e-voting, " *IEEE Software*, vol. 35, no, 4, pp. 95-99, 2018.

[20] R. Hanifatunnisa and B. Rahardjo, “Blockchain based e-voting recording system design,” in Proc. *11th International Conference on Telecommunication Systems Services and Applications*, 2017.

[21] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, “Where is current research on blockchain technology? a systematic review,” *PloS one*, vol. 11, no. 10, pp. e0163477, 2016.

[22] D. Khoury, “Decentralized Voting Platform Based on Ethereum Blockchain,” in Proc. *2018 IEEE International Multidisciplinary Conference on Engineering Technology*, 2018.

[23] V. Buterin et al., “A next-generation smart contract and decentralized application platform,” *white paper*, 2014.

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

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

[26] Estonian National Electoral Committee “E-voting System”, 2010. [Online]. Available:https://www.valimised.ee/sites/default/files/uploads/eng/General_Description_E-Voting_2010.pdf.

[27] S. Heiberg and J. Willemson, “Verifiable internet voting in Estonia,” in Proc. *2014 6th International Conference on Electronic Voting: Verifying the Vote*, 2014.

[28] A. D. Gritzalis, “Principles and requirements for a secure e-voting system,” *Computers & Security*, vol. 21, no. 6, pp. 539-556, 2002.

[29] P. Van Den Besselaar et al., “Experiments with e-voting technology: experiences and lessons,” 2003.

[30] T. W. Lauer, “The risk of e-voting,” *Electronic Journal of E-government*, vol. 2, no. 3, pp. 177-186, 2004.

[31] L. Mitrou, G. Dimitris, and S. Katsikas, “Revisiting legal and regulatory requirements for secure e-voting,” *Security in the Information Society*, Springer, pp. 469-480, 2002.

[32] D. Chaum et al., “Scantegrity: End-to-end voter-verifiable optical-scan voting,” *IEEE Security &* Privacy, vol. 6, no. 3, pp. 40-46, 2008.

[33] H. Tian, L. Fu, and J. He, “A simpler bitcoin voting protocol,” in Proc. *International Conference on Information Security and Cryptology*, 2017.

[34] Y. Wu, “An e-voting system based on blockchain and ring signature,” *Master Thesis.* University of Birmingham, 2017.

[35] S. Bistarelli et al, “An end-to-end voting-system based on Bitcoin,” in Proc. *Symposium on Applied Computing*, 2017.

[36] Ethereum Frontier. [online]. Available: https://www.ethereum.org/.

[37] E. Yavuz et al., “Towards secure e-voting using ethereum blockchain,” in Proc. *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018.

[38] R. Osgood, “The future of democracy: Blockchain voting,” *Information Security*, 2016, [Online]. Available: http://www.cs.tufts.edu/comp/116/archive/fall2016/rosgood.pdf.

[39] P. Tarasov and H. Tewari, “The future of e-voting.” in Proc. *International Journal on Computer Science & Information Systems*, vol. 12, no. 2, 2017.

[40] F. L. Meeser, “Decentralized, transparent, trustless voting on the ethereum blockchain,” 2017, [Online]. Available: http://aleph.com.mx/docs/blockchain_voting.pdf.

[41] M. Alharby and A. van Moorsel, “Blockchain-based smart contracts: A systematic mapping study,” *arXiv preprint arXiv:*1710.06372, 2017.

[42] F. Hjálmarsson et al., “Blockchain-Based E-Voting System,” in Proc. *IEEE 11th International Conference on Cloud Computing*, 2018.

[43] C. Dannen, *Introducing ethereum and solidity: foundations of cryptocurrency and blockchain programming for beginner, Brooklyn, New York, USA*, 2017.

[44] N. Prusty, *Building blockchain projects: building decentralized blockchain applications with ethereum and solidity*, Packt Publishing Limited, 2017.

[45] M. Adams, *Ethereum: the beginners guide to understanding ethereum, ether, smart contracts, ethereum mining, ICO, cryptocurrency, cryptocurrency investing*, CreateSpace Independent Publishing Platform, 2018.

[46] [Online]. Available: https://www.oreilly.com/library/view/mastering-ethereum/9781491971932/ch13.html.

[47] [Online]. Available: https://medium.com/@mvmurthy/full-stack-hello-world-voting-ethereum-dapp-tutorial-part-1-40d2d0d807c2

[48] [Online]. Available: https://medium.com/@itsromiljain/getting-started-with-ethereum-and-building-basic-dapp-ebb681fb3748

[49] [Online]. Available: https://www.sitepoint.com/beginners-guide-node-package-manager/

[50] [Online]. Available: https://www.trufflesuite.com/docs/truffle/reference/contact-the-developers

[51] [Online]. Available: https://www.trufflesuite.com/ganache

[52] [Online]. Available: https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en

[53] [Online]. Available: https://metamask.io/

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

[55] D. Mohanty, *Ethereum for architects and developers with case studies and code samples in solidity*, Apress, 2018.

[56] [Online]. Available: https://www.sublimetext.com/3

[57] N. Mpekoa and D. van Greunen, "E-voting experiences: A case of Namibia and Estonia," in Proc. *IST-Africa Week Conference*, Windhoek, 2017.

[58] R. Tso, Zi-Y. Liu, and J.-Ho Hsiao, "Distributed E-Voting and E-Bidding systems based on smart contract," *Electronics*, vol. 8, no. 4, pp. 422, 2019.

[59] S. Mello-Stark and E. A. Lamagna, "The need for audit-capable e-voting systems", in Proc. *31st International Conference on Advanced Information Networking and Applications Workshops*, Taipei, Taiwan, 2017.

[60] D. Roman and G. Schmid, "Beyond bitcoin: A critical look at Blockchain-based systems," *Cryptography*, vol. 1, no. 2, pp. 15, 2017.

[61] [Online]. Available: http://www.dappuniversity.com/articles/the-ultimate-ethereum -dapp-tutorial

PUBLICATIONS

Sonal Saxena and Seema Bawa, “Emergence of Blockchain in the Cloud Computing: A review and its applications” IETE Technical Review (Communicated).

PLAGIARISM REPORT

ME thesis

ORIGINALITY REPORT

8%	7%	1%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.dappuniversity.com Internet Source	5%
2	Submitted to University of West London Student Paper	1%
3	Submitted to Napier University Student Paper	1%
4	David Khoury, Elie F. Kfoury, Ali Kassem, Hamza Harb. "Decentralized Voting Platform Based on Ethereum Blockchain", 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), 2018 Publication	<1%
5	www.irjet.net Internet Source	<1%
6	medium.com Internet Source	<1%