

University of Southampton

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

Blockchain-based reputation system with Pagerank algorithm

by

Teeraphon Issaranuluk

October 2022

Supervisor: Professor George Konstantinidis

Second Examiner: Professor Nicholas Gibbins

A dissertation submitted in partial fulfilment of the degree
of MSC

University of Southampton

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
ELECTRONICS AND COMPUTER SCIENCE

Master of Science

by Teeraphon Issaranuluk

Blockchain technology is being studied in several fields for its capacity to deliver distributed, decentralised transactions with its timestamp. It is characterised by its fault-tolerance and downtime-less properties, as well as approaches to secure immutable data records such that their alteration is computationally impossible. An online interaction system's trust frameworks and reputation models are responsible for supplying sufficient information (e.g., a reputation score) to infer the trustworthiness of interacting entities. The computation is applied to the Pagerank algorithm to compute the rank of importance for each entity. Consequently, the reputation system must assign a precise, dependable, and unchangeable trust score since the score will store in the immutable chain of a transaction block. This master's thesis examines the proof-of-concept prototype of a blockchain-based reputation system using the Pagerank algorithm to compute the reputation score. It presents a system of smart contracts that establish interaction logic and model trust among the system's pseudo-anonymous identities. The contract is installed on a blockchain network that computes, stores, and updates the reputation score of entities. The suggested technique and trust metrics are tested by replicating an interaction network with a different degree of node number and transaction randomness. The research findings demonstrate that the suggested strategy is to perform efficiently in a small simulation network in performance and rank accuracy. ...

Acknowledgements

I would like to thank my thesis supervisor, professor George Konstantinidis for his direction and unwavering support during the thesis time and for introducing me to the proper tools and ideas that made my thesis feasible. And thank the second examiner, professor Nicholas Gibbins for the valuable opinion to contribute to the proposed research and useful sources.

In addition, I would like to thank my reviewer, Thanandon Imaromkul, for reviewing the report and giving me informative criticism, a contributive opinion on an aspect of the audience, and helpful ideas about the thesis project. Finally, I would like to thank my family and classmate for their inspiration and kind support.

Statement of Originality

- I have read and understood the [ECS Academic Integrity](#) information and the University's [Academic Integrity Guidance for Students](#).
- I am aware that failure to act in accordance with the [Regulations Governing Academic Integrity](#) may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Contents

Acknowledgements	v
1 Introduction	1
1.1 Project aim and objective	3
1.2 Report structure	3
2 Literature review	5
2.1 Background	5
2.1.1 Reputation System	5
2.1.2 Blockchain and smart contracts	7
2.1.3 Ethereum	9
2.1.4 Pagerank algorithm	10
2.2 Related works	11
2.2.1 Peer-Peer reputation system	11
2.2.2 Blockchain-based reputation system	12
3 Methodology	13
3.1 Problem statement	13
3.2 Blockchain platform comparison	14
3.3 Testing tools and frameworks exploration	15
4 Design and Implementation	17
4.1 System architecture overview	17
4.2 System requirements	18
4.2.1 Vouch machanism	19
4.2.2 Vouch detaching mechanism	20
4.2.3 Node deletion mechanism	22
4.2.4 Pagerank calculation mechanism	23
4.2.5 Penalties mechanism	24
4.3 Development dependencies	24
4.3.1 Node.js	25
4.3.2 Ganache	26
4.3.3 Remix	27
4.3.4 Truffle	29
4.3.5 Web3	30
4.4 Algorithm design	30
4.4.1 Make vouch algorithm	31
4.4.2 Vouch detaching	31

4.4.3	Node deletion	32
4.4.4	Pagerank calculation	33
4.4.5	Apply penalties	34
4.5	Implementation guide and techniques	34
5	Test and Evaluation	41
5.1	Unit testing	41
5.2	Performance	44
5.3	Results	45
6	Discussion	49
6.1	Limitation	50
6.2	Future Work	51
7	Conclusion	53
	Bibliography	55

List of Figures

2.1	Distributed ledger network	7
2.2	chain of transaction's block [11]	8
2.3	Overall process inside blockchain [11]	8
2.4	Directed graph problems	10
4.1	System architecture	17
4.2	Users in the system	18
4.3	Vouch mechanism	19
4.4	Vouch detaching mechanism	20
4.5	Node deletion mechanism	22
4.6	Penalties mechanism	24
4.7	Node JS official website	25
4.8	Node JS installation verification	26
4.9	ganache installation	27
4.10	Run Ganache	27
4.11	Remix installation	28
4.12	Remix plugin on vscode	28
4.13	Truffle installation	29
4.14	Truffle package verification	29
4.15	Web3 package installation	30
4.16	Web3 package connection	30
4.17	Structure of entities	35
4.18	Modifier of unvouch and token verification	36
4.19	Modifier of vouch verification	36
4.20	Pagerank calculation function code snapshot	37
4.21	Make vouch function code snapshot	38
4.22	Vouch detaching function code snapshot	39
4.23	Node deletion function code snapshot	40
4.24	Node penalties function code snapshot	40
5.1	Unit test on vouch function	41
5.2	Unit test on pagerank calculation function	42
5.3	Unit test on node deletion function	43
5.4	Unit testing result screenshot	43
5.5	Performance evaluation	44
5.6	Sample graph for testing	45

List of Tables

3.1	Blockchain platform comparison	14
3.2	Testing tools exploration	15
4.1	Requirements set for vouch mechanism	20
4.2	Requirements set for vouch detaching mechanism	21
4.3	Node deletion mechanism requirements	22
4.4	Requirements set for Pagerank calculation mechanism	23
4.5	Requirements set for penalties mechanism	25
5.1	Performance test environment configuration details	44
5.2	Result of rank and score of graph A	46
5.3	Result of rank and score of graph B	47
5.4	Result of rank and score of graph C	47

List of Algorithms

1	Make vouch	31
2	Vouch detaching	32
3	Node deletion	33
4	Pagerank Calculation	34
5	Apply penalties	34

Chapter 1

Introduction

The enormous rise of networking across the internet has contributed to a dramatic increase in the number of transactions between network nodes. Since the comprehensive varieties of transactions implemented in the recent decade, the transaction might represent a critical process from application, for instance, financial transactions among users and the approval process in an organisation. For this reason, the role of trustworthiness between the massive anonymous identity nodes might be considered to take into account and has received increased attention across several disciplines in recent years. Nevertheless, the trustworthiness or reliabilities of the transaction between untrusted or anonymous accounts might be fixed by using a trustable central third party. This methodology might be known as the centralisation system. Besides, the financial transaction is primarily conducted and processed through trusted financial institutions, which perform as a middle party to verify and authorise both transaction and peer in the specific financial network.

Moreover, there are several proposed applications for computing the trustworthiness of the node population inside the network, for example, the reputation system. The system might measure the numerical procedure's reliability, trustworthiness or centralities. On the other hand, although many reputation systems seem to perform as a rating system, the reputation system may be implemented in the theme of reliabilities calculation. This point will be emphasised in the section 2 to demonstrate a clear example of how the proposed application will perform. However, various reputation system applications are implemented in the traditional method, with the centralisation system methodology, which may manipulate by a trusted third party, the platform owner, a trusted middle person, etc. Although the centralisation method can offer significant benefits, such as system reliabilities, consistency, efficiency and simplicities of system implementation and maintenance [35, 46]. Regardless, some critical drawbacks must be considered. In order to accomplish high reliability, the major party must be manipulated and maintain the tremendous transaction and high complexity of the system. This might demonstrate the single point of the failer problem [47]; while the parent node deals with intensive

processing, there is a risk that the central node might crash, which may affect entry network availabilities. Furthermore, the limitation in scalability is also considered when using the centralisation paradigm to implement the network. For this reason, the decentralisation method seems robust for the reputation system network, which requires high consistency, transparency and reliabilities.

The decentralising system can be defined as the information network system in which no single party has authority power. Decentralised systems are often networked in computing, information technology, and applications in various domains, such as cryptocurrency. As mentioned earlier, decentralisation comes up with three main significant beneficial aspects. The first decentralisation system can serve the high availabilities of the application network. The system can accomplish the problem of a single point of failure to distribute the computing power, the data replication - such as the ledger that distributes across the network node [27, 30, 47]. Secondly, centralisation is the system in which the node is forced to trust the central party, which might risk data privacy. Due to the central authority performing as the signal right to manipulate transactions and data in the entry system; in other words, it has the owner privileges [35]. Finally, not only the personal data might be at risk for harm to privacy, but the transparency of transaction data is also restricted according to the highest privilege in the entire system. Since the decentralised system indicates the immutable system, data is validated and published to all peers in the network, which is transparent and ensures that data is not modified [27, 40].

Decentralisation is the methodology behind well-known technology, blockchain technology. Blockchain technology represents the decentralised ledger system which consists of the node of users in the networks, technically called peer-to-peer networks [37, 47]. However, as mentioned earlier, the enormous of nodes in the system might be critical to the reliabilities of the system, especially the network application that might be located in a required field such as financial. Then, the network's avoidance and prevention of fraud must be considered. This reason might motivate this research to develop a robust mechanism for the reputation system, which will be implemented on top of blockchain technology to utilise the highlighted benefits of blockchain-based applications as a decentralised system. Moreover, the appropriate algorithm for reputation is also purpose by integrated with the page ranking algorithm to manipulate and measure the centralities, in other words, the trustworthiness of nodes in the system. However, this research will try to integrate the Pagerank algorithm or another potential approach to accomplish the goals of this research.

1.1 Project aim and objective

The significant challenge with the dramatic growth of node size in networks nowadays is the reliabilities of the population interacting inside the network. Once the network is implemented through the centralisation approach, it might face a single point of failure, data transparency, and privacy violation. Thus, the decentralisation system, such as blockchain technology, is considered a high potential technology for developing the mechanism in this project. However, in order to utilise the considered beneficial aspect of blockchain technology implemented, follow along with the decentralisation paradigm. This research aims to deliver a high-quality prototype of a blockchain-based reputation system mechanism through a proper approach and research methodology.

The proposed mechanism demonstrates the transaction scenario between two network nodes, assuming Alice and Bob. Once Alice decides to make the transaction with Bob in a specific network, Bob's trustworthiness might be ambiguous from Alice's point of view. According to this situation, Alice may personally know Bob's reputation; in other words, a sense of Bob's reliabilities. This sense was developed as the reputation system that allows users in the network need to make the transaction of vouching to another person. Bob receives the vouch transaction from Alice, which gives him the reputation score; since there are two people in the example scenario, Bob illustrates the highest centralities or reliabilities in this case. Nevertheless, There is no guarantee that Bob, who got the highest trust from users in the system, will not behave destructively in the future. Therefore, the mechanism also included the penalties algorithm, which was used for penalising Bob by removing this reputation and Alice as a person who performs as a guarantor of Bob. In other words, not only Bob lost trust in the system, but also Alice's trustworthiness might have been reduced according to the bad behaviour of her vouchee. In addition, the research tries to answer the following questions;

- How to implement a blockchain-based reputation system?
- How well does the mechanism perform for computing ranking scores in smart contracts?

Nonetheless, there is significant research and study of the technology and robust algorithm to answer the research questions, which will be described in section 2.2, the methodology.

1.2 Report structure

In preparation for describing all processes of implementation and evaluation of entry research, the structure of the dissertation was illustrated as follows.

- Chapter 1: Demonstrate and introduce the project motivation and expected contribution, which includes the objective and aim of the research used to answer the stated research question based on the presented issue.
- Chapter 2: The next stage is to preserve the necessary background, which includes the literature review of related work, algorithms and technology. This chapter also provided the important definition of terminology used in this research.
- Chapter 3: The chapter illustrates the details of the methodology of this project, which included the problem statement and another comparative study to implement and research the proposed system.
- Chapter 4: The fourth section highlights the design and implantation process, which consists of the development dependencies, functional decomposition, algorithm design, and the important coding technique that must be noticed to understand this research effectively.
- Chapter 5: The next chapter depicts the evaluation and implementation results of the developed mechanism by performing the experiment performance and functional testing.
- Chapter 6: The section summarises the research result, discusses the result in several aspects, and defines the work limitation, and further scope, which reflects the possible improvement delivered project.
- Chapter 7: The last section summarises the research process and emphasis the research objective, limitation, and further scope.

Chapter 2

Literature review

2.1 Background

The reputation system can illustrate the application across various domains, with several definitions on different aspects. In addition, the reputation system is also implemented in a wide variety of technologies and methodologies. Therefore, it is crucial to consider the terminologies used as a background before illustrating the research in the next stage. In the same way, blockchain technology is considered the highlighted technology stack of this project and occurs in a wide range of application studies. However, blockchain technology and its application also researched and explored and analysed the potential of each blockchain type in a similar field of research. Thus, this section focuses on the necessary technical background, providing and discussing the definition of highlighted technical terms used in this research. Moreover, a similar study is introduced and discussed in this chapter to analyse and demonstrate the proposed work.

2.1.1 Reputation System

The words 'trust' and 'reputation' are typically used as synonyms, although these two words might represent completely different meanings [34]. Although these two terms have separate meanings, there might be some relation between these words in the sense of study in a similar field. According to Ferry Hendrikx et al.[26], reputation might be considered a tool for trust contribution. In other words, a higher reputation can increase trust, and vice versa.

The reputation system may refer to the system that can perform computationally, collecting and sharing the individual behaviour in the specific network [26, 28]. The system consists of three main properties identified in various previous studies. Firstly, trustworthy information of each participant in the network must be shared to deliver further information that might be used for consideration and decision-making of other entities[32].

Secondly, The system indicates the acceptable manner of people in the network. Since reputation information is shared across the network, reputation data is immutable; people tend to avoid penalising, which might reduce or remove their reputation. Finally, an efficient reputation system might consist of a robust punishment algorithm for the misbehaviour of the individual peer. Penalties used to prevent collusion leading to malicious activities across people in the network might be considered to apply both guarantee and guarantor.

However, the reputation system in the literature can be separated into two principal categories, distinguished from the structure of implemented approach, which are centralised and decentralised models. Initially, a centralised approach demonstrated the reputation model manipulated by the middle party. The central authority is responsible for storing and controlling the reputation information by computing and publishing it to the attending node. This model was initially found on "eBay" website [28]; the reputation scores of the merchant on the platform are calculated through the node that owns by eBay and distributed back to all merchant accounts in public. This model is also widely used in current e-commerce platforms, in which the number of merchants and buyers is noticed as untrusted entities from each other. On the other hand, decentralised model, reputation scores are published across the entry entities without a central authority. The decentralised method is highlighted and trendy for the peer-to-peer network, in which the securities and consistency of transaction ledger may be considered part of the implementation [32].

Regarding categorisation of the reputation system, not only divided by network architecture methodology, systems can be categorised in the aspect of "semantic trust". According to the study by Audun Jøsang et al. [29], reputation can be separated into two main types: "Subjective" and "Objective". The subjective reputation system relies on ratings supplied inside a local community of users. In other words, the user gives the reputation base on subjective evaluation [29]. The group has well-defined functions, such as selling products, sharing material, characterising users' experience, and sharing expertise or views. Both the community service provider and the reputation system have a reputation. Users provide "subjective" reputation ratings to other community members based on their interactions and the achievement of personal objectives and commitment, which may be modified over time [16]. In comparison, the objective reputation system is preserved based on the actual evidence to compute the reputation score [16, 29]. This data may be subjected to a personal study for the individual to arrive at his reputation evaluation (and share it with other users). Nonetheless, the essential "reputation measures" are based on objective, community-observable facts (e.g. reports, analysis-based scientific metrics and criteria, average bit rate of a video streaming server).

2.1.2 Blockchain and smart contracts

Initially, blockchain technology was introduced as the underlying pin behind "Bitcoin". In 2008, Satoshi Nakamoto proposed studying a new alternative digital cash system called "Bitcoin" [36]. The introduced currencies are implemented in the theme of a decentralised system; in other words, distributed ledger, the network's participants may evaluate distributed ledgers at each network node and be able to acquire an identical copy of the network-wide shared recordings. If the ledger is modified or supplemented, the modifications are duplicated and sent to the participants. As mentioned in figure 2.1, it is synced to ensure that the database is correct.

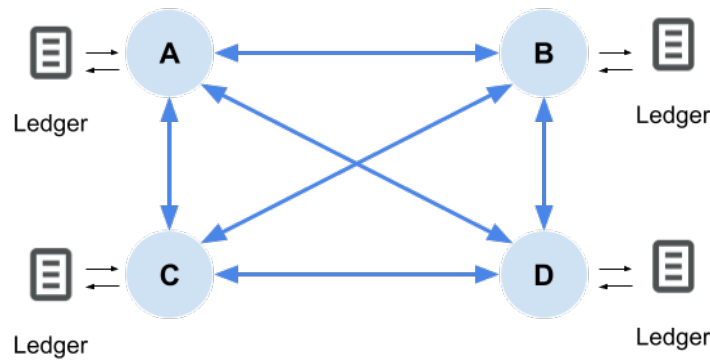


FIGURE 2.1: Distributed ledger network

The transaction data in the system implemented follows the mentioned approach seems to highlight the immutable and transparent. Since the transactions that occur inside the blockchain is stored in a chain of block [36, 49]. This chain expands continually as additional blocks are added. The essential properties of blockchain technology are decentralisation, assiduousness, namelessness, and audibility. Integrating numerous essential technologies, including cryptography, consensus protocol, and digital signature, allows blockchain to operate in a decentralised context. Using blockchain technology, a transaction of the implemented system might be conducted in a decentralised approach. Consequently, blockchain may significantly reduce costs and increase efficiency [49].

Figure 2.2 illustrates a typically characteristic instance of a blockchain. A blockchain consists of a chain of the set of transaction data (blocks). Additionally, each block includes a created timestamp, the hash value of the prior block, and a random number used to verify the hash. This approach ensures the integrity of the whole system, which might track back to the original block (the "genesis block"). Changes to a block in the chain would presently affect and change another unique hash value for that block [37, 49]. Therefore, the information on the blockchain cannot be modified.

The dramatical growth of blockchain technology in recent years lends credence to other hypotheses proposed in the literature. Szabo et al.'s study first established the notion of

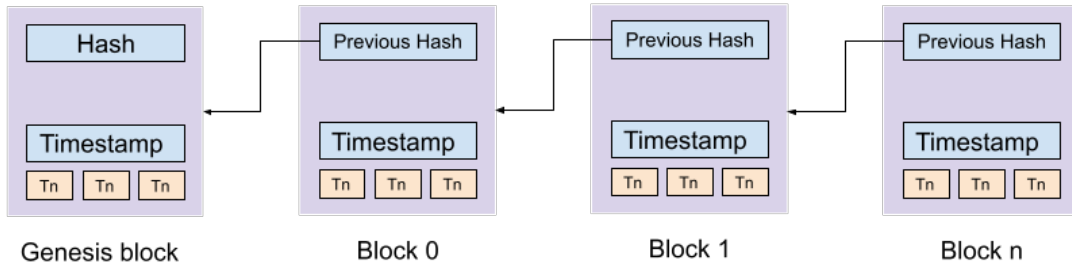


FIGURE 2.2: chain of transaction's block [11]

"Smart Contracts," which integrate computer protocols and human interfaces to execute the provisions of a contract [39]. In other words, "smart contracts" are blockchain-based applications that run automatically when certain conditions are satisfied. Typically, they are used to automate the execution of a contract so that all parties may inform the conclusion immediately and without central authority required. In addition, they may automate a process by launching the next activity when certain conditions are met. By using blockchains, the smart contract may be employed more effectively. Ethereum, a decentralised system first presented by Buterin et al., is a famous example of blockchain technology that addresses intelligent contracts [15]. Ethereum may be classified as an expanded version of the Bitcoin blockchain in order to accommodate a wider variety of applications. Thus, blockchain technology enables the creation of contracts utilising encryption and replacing third parties' authority to build confidence. Blockchain may disrupt the whole transaction procedure by automatically executing contracts transparent, secure, and cost-effectively [23].

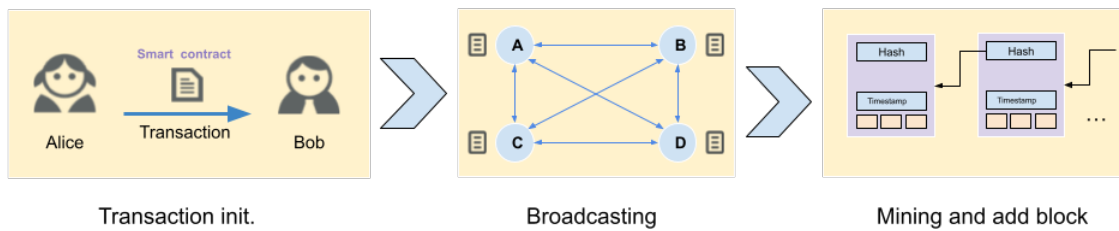


FIGURE 2.3: Overall process inside blockchain [11]

To illustrate the overall process of introduced components mentioned above. The process starts when users initiate the transaction; for example, transferring the digital coin or executing and requesting the smart contract operation. Once the transaction is started, the transaction will be validated and distributed to all network participants. In the next stage, typically, the blockchain environment will have a miner or transaction validator responsible for verifying and validating the initiated transaction and storing it in the block. At this stage, the validator will gather the verified transactions at a specific time

frame and finalise the block of the transactions. Before the add a new block to the chain in the network, the miner must accomplish the consensus mechanism applied in the system and add a new block to the chain.

2.1.3 Ethereum

Ethereum is the first platform that applies the concept of smart contracts concept, in order to maintain and enhance the transaction integrities across the network. In other words, this platform is proposed as the extended version of Bitcoin, which highlights programable blockchain platforms. Ethereum offers the state virtual machine, Ethereum Virtual Machine (EVM), in which contract code may be performed to produce a deterministic output given the same transaction context, and blockchain state [45]. The EVM, known as a single-world computer, operates on every Ethereum node and generates the same end state given the same beginning state. Multiple high-level programming languages may be used to create smart contracts for various blockchain systems, for example, solidity languages. The contract code remains in an immutable form on the blockchain. They are not autonomous, self-executing programmes but must be activated by a transaction or other contracts. Once the code is registered and deployed on the blockchain, no one, even the contract's owner, may modify it.

Nonetheless, the owner can provide a killable function that, when called, performs an EVM opcode called "SELF-DESTRUCT" and logically deletes the contract from the blockchain, i.e. it removes the contract's code and internal data from its address [2]. Once this contract has been deleted, sending any transaction to this address will no longer run any code. As the blockchain itself is unchangeable, doing so does not erase the history of transactions. As with other Turing-complete programming languages, it is influenced by the halting issue, i.e., given an input, there is no way to determine whether the programme will finish. In the event of a non-terminating software, a network-transmitted transaction may continue indefinitely, rendering the network worthless if the transaction cannot be executed. To prevent this scenario, Ethereum adds the idea of gas, a consumable resource on the network that serves as the primary cost unit for the network.

Gas refers to the unit that measures the cost of computational effort required to run specific operations on the Ethereum network. Since each transaction demands computational resources to execute, each transaction requires a fee. Gas refers to the transaction execution fee conducted when the transaction on Ethereum is successful. In other words, gas is used to reduce the danger of exploiting the network with excessive computational expenses and is purchased only using ether, the Ethereum unit of money. Wei is the minimum unit of money in Ethereum ($1 \text{ Wei} = 10^{18} \text{ ether}$) [45]. The gas must be provided to store or execute any state or function. Consequently, a programme with a fault or a non-terminating purpose will ultimately run out of fuel and terminate.

2.1.4 Pagerank algorithm

Google Search uses the PageRank (PR) algorithm to rank websites in their search engine results. PageRank was named after Larry Page, one of Google's founders [14]. PageRank calculates a website's importance by tallying the quantity and quality of inbound links to that page. The basic idea is that more significant sites will obtain more connections from other sites. It is not the only algorithm used by Google to organise search engine results, but it is the corporation's first and most well-known algorithm. The centrality metric described above is not implemented for multi-graphs. ethereum

$$PR(u) = \frac{1-d}{N} + d * \sum_{v \in B_u} \frac{PR(v)}{L(v)} \quad (2.1)$$

As mentioned in equation 2.1, the PageRank value of a page u depends on the PageRank values of each page v in the set containing all pages connecting to page u , divided by the number $L(v)$ of links from page v . The technique includes a damping component, denoted by d , and teleportation for the PageRank computation.

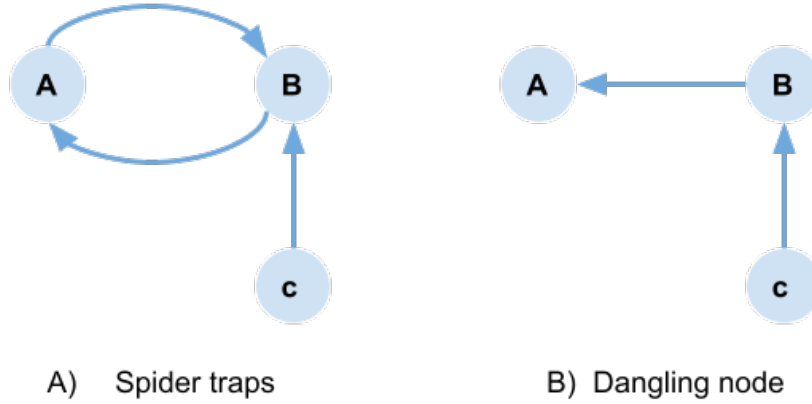


FIGURE 2.4: Directed graph problems

The mentioned technique is used to solve the prevalent directed graph problems, called spider traps and dangling nodes, as mentioned in figures Figure 2.4 - A and B, respectively. In case of spider traps issue, the random walk might trap on nodes A and B, which can affect the centralities score in the graph network. Since the probabilities distribution will cover only two linked nodes, this is unacceptable for the directed graph network's importance [12]. On the other hand, Figure 2.4 - B; the dangling node, called dead end problem, represents the problem that the node does not consist of an outbound link. Nadav Eiron et al. first propose the approach for dealing with this scenario [21]. The proposed approach contributed to the research from David Gleich et al. [25], which introduced the solution by applying the teleportation methodology. The teleportation

technique assumes that equally distributed probabilities are denoted by ϕ , aka—damping factors. The random walk will transverse to each node in graph P (transition matrix of node-link) with the probabilities $\frac{(1-\phi)}{N}$ by N representing the number of nodes in the graph, and move randomly to the specific node. With the introduced approach, a new altered transition matrix of node-link R can be defined as illustrated below equation [25];

$$R = \phi P + (1 - \phi)v\pi \quad (2.2)$$

By the vector of one denoted by v , and π are vectors of $1/n$. However, practically, the damping factor is typically set to 0.85 or 85 percent [13, 22, 24]. The recommended setting seems robust to the field of study and proposed application since it gives an accurate reputation score while applying this factor to the computation.

2.2 Related works

2.2.1 Peer-Peer reputation system

Before blockchain was founded and disturbed information technology, several researchers proposed reputation systems expected to satisfy the implementation of the peer-to-peer (P2P) network. The first study by E. Damiani et al. [18] introduces the reputation system across the peer-to-peer network. The mentioned studied work is a step toward a self-regulating P2P system that can also oblige in isolating from the network nodes deemed to be engaging in illegal or unethical behaviour. Although this research was accomplished in the aspect of the research result [18], the proposed study seems to have been no detailed investigation of the efficiency and highlighted algorithm for calculating the reputation score.

Nonetheless, reputation systems in P2P networks must deal with the identified problems of reputation systems in general and the added complexity of a peer-to-peer network. When implementing a reputation system over a P2P network, additional difficulties arise, for instance, data maintainabilities, consistency, and dynamically transmitted to numerous peers. These challenges were mentioned when the P2P network system was proposed [42]. On the other hand, a reputation system can be applied to various fields. Yao Wang and Julita Vassileva introduced the alternative reputation mechanism, which applied to the underpin of recommendation on the P2P network [18, 43]. The research also integrated the reputation model with the Bayesian model the introduced study might notice as the "Bayesian-based model trust system", which indicated the successful implementation result [43]. However, in the sense of reputation model design, this study

offers that a reputation system can be applied in various senses of the application field. The indicated study might separate the trustworthiness, and reputation [20, 43].

2.2.2 Blockchain-based reputation system

Once blockchain technology was widely used in literature and research, reputation systems study also became a part of this distribution. In 2018, Zhaojun Lu et al. [31] presented the mechanism of the blockchain-based reputation system, also known as "BARS", which is designed to destroy the linkability among real-life identities and public keys for enhancing the privacy of participants in the proposed network. Not only is privacy taken into account, but the proposed research design also improves the trustworthiness of transactions in the system based on both direct and indirect interactions of participants. In addition, the study consists of the penalties and reward mechanism called the "reputation evaluation algorithm", which might be applied to other work on reputation system development. In the same year, similar work was also published on the reputation model and trust framework theme. Sujata Tamang's research [40] indicated and developed the reputation model by applying the Ethereum and innovative contract platform as a base of the proposed work.

Another interesting contributed research was created in the following physical year of 2019. The study by M. Debe et al. [19] applied the reputation model, which is expected on the Ethereum platform, to the IoT technology. The system delivers the trustworthiness of transaction data and storage of the "Fog-cloud" system; this research also highlights the significant beneficial aspect of blockchain-based applications, such as the "single point of failure." through the comparative study in a similar centralised application. However, these mentioned previous works might be illustrated the usage of the Ethereum platform in the research field. In other words, the Ethereum blockchain is considered robust and reliable for implementing the reputation system [8, 17, 41]. Regardless, the reputation model in the existing literature might use a wide variety of mathematical methods in order to compute the reputation score. Several studies use the Pagerank algorithm to calculate a node's importance in the design environment [9, 33, 48]. However, the survey of platforms and related frameworks will be described and discussed in more detail in the next section.

Chapter 3

Methodology

The project may consist of four phases: research and feasibility tests on the technology under consideration, implementation, testing, and assessment, and conclusion and amended thesis documentation. These primary stages begin with research and feasibility testing, followed by a literature study and associated research activity, such as developing a solid framework, testing, and different performance assessment methods. In addition, the accompanying technologies and frameworks were studied and evaluated throughout this phase to determine the implementation's viability. The following phase is the implementation component which covers the process of development. This stage may be subdivided into a tiny subprocess (Sprint) in order to improve development and code quality. After implementation is complete, the prototype of the implemented mechanism is tested, along with an assessment of the development result based on the approach explored during the initiation phase. Even though the thesis documentation is generated and amended concurrently with other processes, the document must be rechecked and updated as necessary.

3.1 Problem statement

Due to the dramatic rise in online networks, the underlying reputation system must be as robust and transparent as feasible. The assurance that available information has not been tampered with and that the claimed identity is accurate must be provided to minimise the risk of fraud. Current online systems are centralised, making the dissemination of reputation information susceptible to external assaults and internal alterations. As a result, it cannot give the assurance of reliable and unchangeable data. In addition, the reputation systems do not account for the anonymity of participants, which is an essential characteristic for preventing retribution when offering considered and reliable vouching to other participants. This master's thesis research proposes using blockchain technology to store and administer reputation data in the sense of vouching system to

secure the integrity and verifiability of publicly accessible information. By modelling trust between entities in a pseudonymous way, the suggested system also considers the application of the PageRank algorithm to calculate the centralities of network participants.

3.2 Blockchain platform comparison

Blockchain platform				
Details	Bitcoin	Ethereum	Hyperledger Fabric	R3 Corda
Purposes	Cryptocurrencies	Dapps	Enterprise Dapps	Enterprise financial Dapps
Operation mode	Permissionless	Permissionless	Private	Private
Smart contract	No	Yes	Yes	Yes
Language	N/A	Solidity	Go, Java, Javascript	Kotlin, Java
Transaction fee	Rely on users	Rely on Gas usage	N/A	N/A
Currency	Bitcoin	Ether	N/A	N/A

TABLE 3.1: Blockchain platform comparison

The research method is started by exploring the considered robust blockchain platform to develop the blockchain-based reputation with the Pagerank algorithm. There are four leading well-known platforms that exist in the current field of study. Firstly, Bitcoin is the first existing blockchain-based platform. The platform is prominent in the first digital cash system and initiated blockchain technology. Nonetheless, as mentioned in the table 3.1, Bitcoin was introduced with the primary purpose of alternative digital currencies [1]. Moreover, the platform does not offer "smart contract" integration, which notices that the platform might not be appropriate for implementing the decentralised-based applications (also known as Dapps).

On the other hand, the remaining introduced platform - Ethereum, Hyperledger Fabric, and R3- illustrate the platform supporting decentralised application development [3]. However, some different details must be considered between the three platforms. The Ethereum and Hyperledger fabric indicate the supposed aspect of Dapps' development. Ethereum can implement on both public and private networks; in other words, the platform is denoted as an open source framework and allows various researchers to develop Dapp in focused research fields. Moreover, this platform initiated its currencies and "smart contracts" concept application.

In contrast, although Hyperledger Fabric was developed to implement decentralised applications, the platform lacks the operation model, offering only a private network or intra-organisation usage [4]. For these reasons, "Fabric" might not be appropriate

to implement the application that highlights uncertainty usage of the application in the future- which means the proposed mechanism may be used as either a general application or privately inside the organisation. Ethereum also provides embedded cryptocurrencies, enabling the platform to apply in various fields, such as financial applications, since it preserves the finance utility function. This benefits aspect also demonstrated the higher flexibilities of application development compared with the Hyperledger fabric platform.

Another platform included in the consideration is R3 Corda. With Corda, it is not intended to create digital currencies or tokens, even though the platform's purpose was to highlight the application in the financial domain. R3 Corda, similar to the Fabric, is not provided with built-in currencies and necessary financial functions [6]. Compared with Ethereum and Fabric, this platform also indicated the weakness of implementation's flexibilities and simplicities. In a nutshell, Ethereum has illustrated the prominent blockchain platform which will be applied in this research.

3.3 Testing tools and frameworks exploration

Testing tools and platform				
Name	Documentation	Support solidity version 8	Support web3 integration	Github
Solidity-Coverage	-	✓	-	✓
Waffle	✓	-	✓	✓
Remix Tests	✓	✓	-	✓
OpenZeppelin	✓	✓	-	✓
Truffle suite	✓	✓	✓	-
Brownie	✓	✓	✓	✓
Foundry Tests	✓	-	✓	✓
Etheno	-	✓	-	✓

TABLE 3.2: Testing tools exploration

Several tools and frameworks for testing the Ethereum smart contract might be widely known. Since the proposed mechanism intentionally expects various applications that might contribute to this research. The testing tools must support the front-end integration, such as web3, which presents one of the well-known front-end frameworks that communicate with the Ethereum smart contract. One of the tools that seem proper for this project is the remix. The tools also provided a contract editor, an automated compiler and simple deployment for implemented smart contracts. However, the remix unit test function also performs unit testing on developed contracts. Thus the remix is one potential candidate's tool for testing the developed contract. However, as mentioned in table 3.2, the remix does not support web3 integration, compared with truffle, which offers web3 support. Truffle might be considered as an essential framework to use in this research.

Chapter 4

Design and Implementation

4.1 System architecture overview

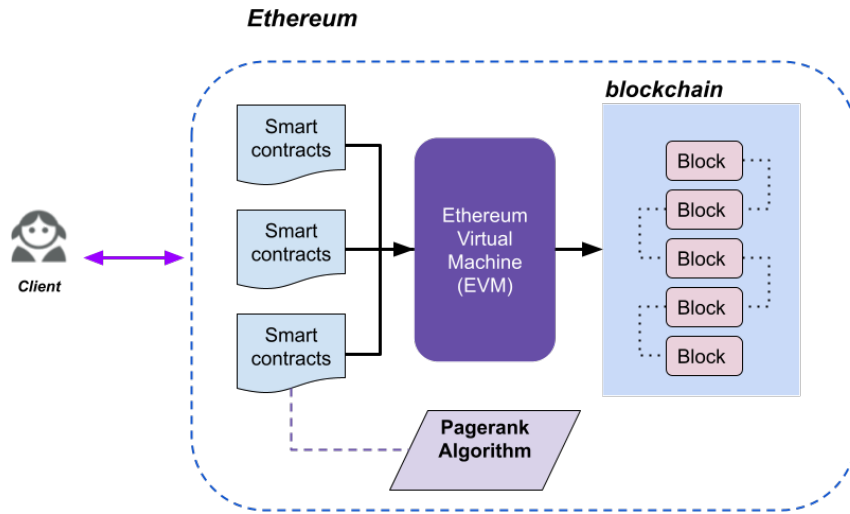


FIGURE 4.1: System architecture

According to the project's purpose, to develop the mechanism of a blockchain-based reputation system, the system might include only backend instruments in order to provide opportunities for extending the project in the future. The system architecture consists of two main components, the client side and the backend. In the introduced study, the user is allowed to interact directly with the developed mechanism of the vouching system. Once a user initiates the transaction, making vouch or rejecting other users, the transaction will directly interact with the blockchain system. Inside the blockchain environment, the transaction will execute the smart contract code. These contracts are responsible for ensuring that all transactions will align with the system requirements (more details on section the next section).

Moreover, the smart contract is also used to calculate the ranking score of each peer in the system. During the transaction and contract was executed, all contract codes complied through EVM and distributed to all system peers as mentioned in section 2.1.2 and 2.1.3. Finally, the transaction will be mined and stored in a block as mentioned in figure Figure 4.1.

4.2 System requirements

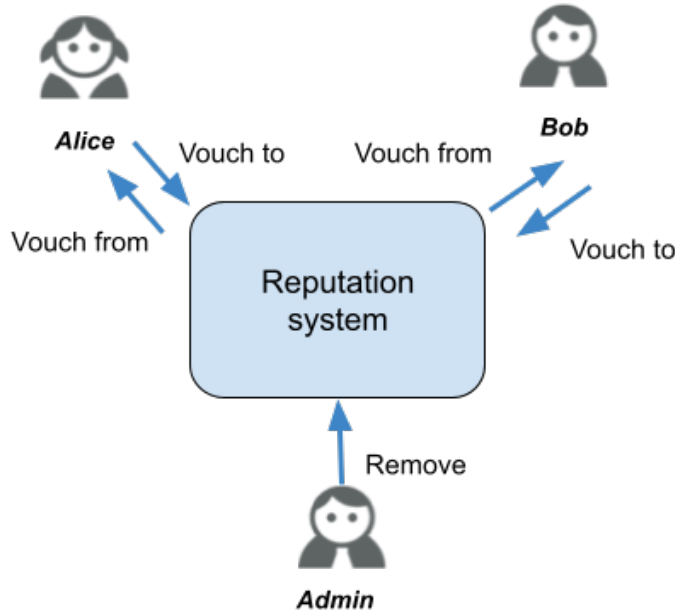


FIGURE 4.2: Users in the system

This research offers a decentralised vouching system to represent participant trust. As suggested by its name, the proposed system enables players to endorse one another to express their subjective judgement of other participants. The voucher (who provides the vouching) and the vouchsee (who gets a vouch or guarantee) are the two unique user roles in this system. The link between a voucher and a vouchsee is an endorsement relationship. One may want to build an endorsement relationship with other network organisations based on actual or digital acquaintance. However, the roles in the proposed system might overlap. In other words, voucher and vouchsee might represent the same person in the system; Alice and Bob can be both voucher and vouchsee simultaneously, as demonstrated in Figure 5.4. However, the admin role - in this case, the contract owner, has the right to remove the participant from the system. This scenario considers bad behaviour a threat to the system, and voucher(s) will get penalties automatically.

Based on her prior interactions with Bob, Alice is likely to vote with him on the reputation system. Therefore, the purpose of the endorsement connection is to illustrate

the direct personal/interpersonal trust between entities. The reputation system aims to provide a straightforward computing model for aggregating the reputation score of interactions and assigning a reputation value to infer trustworthiness through the Pagerank algorithm. If participant A vouch for participant B, A has confidence in B. Therefore, this system's domain of trust value is binary. In the network, B either vouch or does not vouch to entity A. Figure 5.4 depicts the endorsement system's interaction with its users (voucher and vouchee). In the software development paradigm, functional requirements may give a formal description and details of the system's features.

4.2.1 Vouch machanism

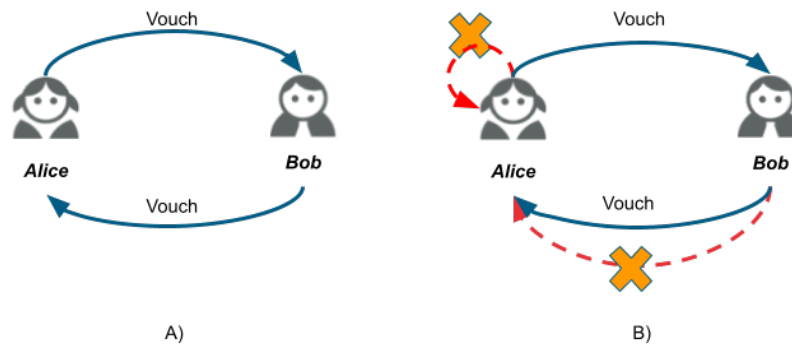


FIGURE 4.3: Vouch mechanism

The proposed system offers the vouch function to the user. This feature introduced the mechanism of vouch transactions. In other words, the transaction illustrates direct graph edges; graph edges have a direction. Thus the transaction between Alice and Bob is represented in Figure 4.3 - A, which means Alice vouch for Bob and vice versa, demonstrating two edges in the graph network. However, the suggested reputation system is proposed to perform as a trusted network; a few scenarios will restrict since there may conflict with usage requirements. First, as mentioned in Figure 4.3 - B, entities in the system not allow to make a transaction to themselves. It represents that user intends to vouch for themselves, which might be defined as collusion for reputation and not indicate the actual use case of the reputation system. Additionally, duplicate transactions are also not permitted in this project. The list of requirements set for the vouch function is described in the below table.

Functional requirements		
Function name	Requirement	Requirement ID
A. Vouch mechanism	[1] Transaction owner is allowed to make a vouch once to each account (another node)	A1
	[2] Transaction owners do not allow to vouch for themselves.	A2
	[3] Each transaction represents the edges of graph networks.	A3
	[4] For Every transaction, the smart contract will calculate and update the PageRank score on each node in the network.	A4
	[5] Vouch system represents the directed graph in which edges of inbound links do not equal outbound links.	A5

TABLE 4.1: Requirements set for vouch mechanism

4.2.2 Vouch detaching mechanism

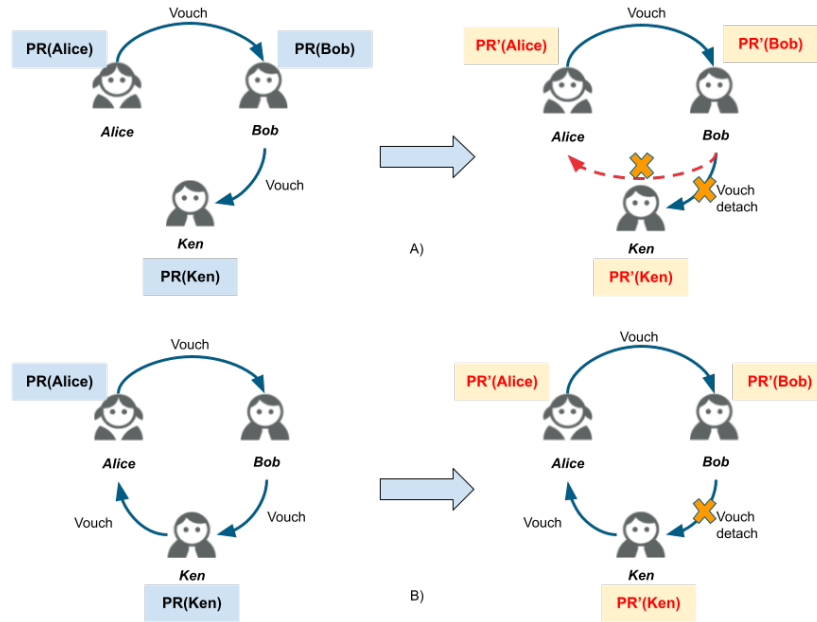


FIGURE 4.4: Vouch detaching mechanism

In contrast, the system also provided the vouch detaching mechanism in order to reject the vouch that performs previously. The action of vouch detaching in the proposed system is demonstrated the edge removal in a typical directed graph network. Nonetheless, the removal process might be customised to apply to the expected use case of introducing the mechanism. The condition can be categorised into two main scenarios - illustrated on Figure 4.4.

Functional requirements		
Function name	Requirement	Requirement ID
B. Vouch detach-ing mechanism	[1] Transaction owner is allowed to make a vouch detaching once to the account.	B1
	[2] Transaction owner not allowed to make a vouch detaching to account not vouched by transaction owner.	B2
	[3] Transaction owners do not allow to un-vouch themself.	B3
	[4] Each transaction represents edge removal for the graph network.	B4
	[5] Every time that transaction happens, the smart contract will calculate and update the Pagerank score on each node in the network.	B5
	[6] The detached node must be removed from the network if no inbound and outbound link is left.	B6
	[7] The smart contract must update each node's PageRank score if the node is removed from the network.	B7

TABLE 4.2: Requirements set for vouch detaching mechanism

As mentioned on Figure 4.4 - A, the user can unvouch only the entities to which the voucher proceeds transaction. To emphasise this scenario, the transaction owner- called a voucher, is allowed to make a vouch detaching once to the account, specifically, the account that the voucher made vouch in the past. Technically, edges occurred between two entities from the voucher's node to detach the link. In other words, the transaction owner is not allowed to make vouch for an account not vouched by the transaction owner or not have the edge between entities. This mentioned condition is also included as a part of the requirement mentioned in the table 4.2 - B1 to B2. In addition, comparable to the vouch procedure, detaching is not allowed to perform on the transaction owner node. This means the detachment not allow to make to themself and node(s) that do not vouch.

The next scenario demonstrated on Figure 4.4 - B. Once users reject their vouch back, the detached node might either remain in the network or be removed since node inbound and outbound link to that entity; the system will calculate and update the Pagerank score on each node in the network. The detached node must be removed from the system if one vouch or does not make vouch to anyone. Therefore, the system will recalculate the Pagerank score every time that edge or node is created, updated, and deleted. However, the Pagerank score must be sorted to represent the most important - the highest trust in the system.

4.2.3 Node deletion mechanism

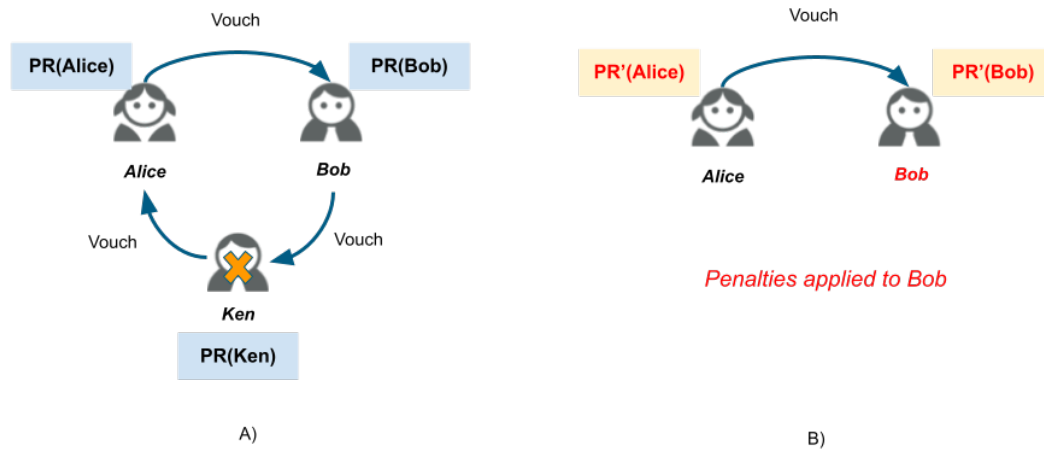


FIGURE 4.5: Node deletion mechanism

The entity deletion, in this case, represents the penalties or adverse action to the entity that misbehaves in the network. This intentional deletion is an absolute difference from node removal in case the node does not contain any link in the system. As mentioned above, this critical action will restrict to admin role in the introduced application. In this case, the admin role will depict the person who owns the contract - the contract owner. To simplify, only the contract owner has the right to delete entities from the system.

Functional requirements		
Function name	Requirement	Requirement ID
C. Node deletion mechanism	[1] Only the contract owner (Admin) has the right to delete entities from the system	C1
	[2] voucher of the deleted node will get a penalty (The penalty algorithm will be explained in detail in the next section.)	C2
	[3] Outbound link (edge) of the deleted node to another node will remove.	C3
	[4] After deleting the target node, the system will update the new PageRank score of all nodes.	C4
	[5] Deleting the node that does not have inbound/outbound does not apply the penalty.	C5

TABLE 4.3: Node deletion mechanism requirements

Nonetheless, as mentioned on Figure 4.5, the voucher which makes vouch for the deleted node will get a penalty (the penalty algorithm will be explained in detail in the next section) as the person who endorses deviant entities. Consequently, the outbound link (edge) of the deleted node to another node will be detached. After deleting the target node, the system will update the new PageRank score of all nodes. In such a case, the penalties will not be applied if deleting the node that does not have inbound/outbound does not apply the penalty. In order to clarify the composition of the node deletion feature, functional/ non-functional requirements are listed in the table 4.3.

4.2.4 Pagerank calculation mechanism

Behind the mechanism, the trustworthiness - the centralities aggregation of entities in the network was pushed by the Pagerank algorithm. The smart contracts are provided with the appropriate ranking calculation and store the ranking score for each peer interacted inside the network. The introduced algorithm was followed along with the equation 2.1, which fully describes in section 2.1.4. The algorithm will compute the Pagerank score of each account in the blockchain system; in other words, account and vouch transactions between entities demonstrated nodes and edges, respectively. However, as mentioned in the previous section, the Pagerank score must be recalculated when the edge is modified. Similarly, when the number of nodes in the network was modified (Create, update, and delete), the score and rank must be updated. These can be listed, and the requirements are mentioned in the table below.

Functional requirements		
Function name	Requirement	Requirement ID
D. Pager- ank calculation	[1] Pagerank score of each node in the network must be calculated correctly.	D1
	[2] Pagerank score must be recalculated when the edge is modified. (vouch and unvouch)	D2
	[3] Pagerank score must be recalculated when the number of nodes in the network changes. (node removed)	D3

TABLE 4.4: Requirements set for Pagerank calculation mechanism

4.2.5 Penalties mechanism

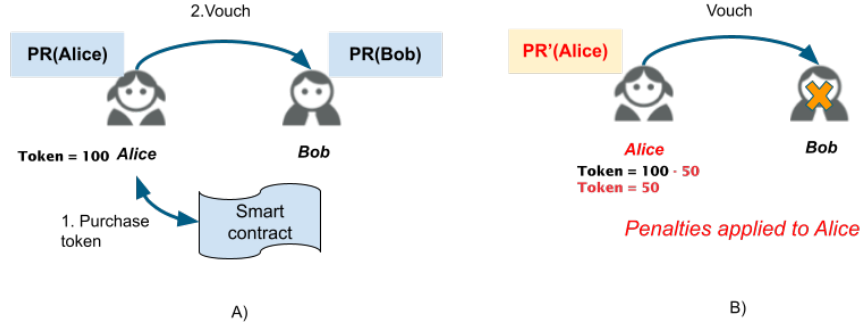


FIGURE 4.6: Penalties mechanism

The designed penalties function used in this project might be associated with the vouch function. In order to make vouch, the user must be validated with enough power to send the enforcement to other entities in the network. The process starts when the voucher deposits one ether to the smart contract to purchase the vouch token. Initially, the token represented the right to initiate the transaction of vouch in the system. To emphasise, the endorser is allowed to vouch only when the remains vouch token exceeds the fifty per cent threshold - in this case, fifty tokens. As part of the token deduction, the portion at each specific time will depend on the number of entities in the system as depicted in Figure 4.6. The following formula describes the cost of reduction;

$$Cost = 100/N \quad (4.1)$$

N is the total number of nodes in the network. However, if the user intends to reset the token, the voucher must deposit another ether to reset the vouch token back to one hundred per cent. The requirements all summarise in below table.

4.3 Development dependencies

The implemented project consists of several frameworks and tools that dependencies. The tools described in this section are part of the application development cycle - for example, implementation, testing, and evaluation- to implement the Ethereum blockchain-based reputation system with the Pagerank algorithm. This project is associated with various plugins and tools necessary to implement blockchain-based applications such as Node.js, Ganache, Remix, Truffle.

Functional requirements		
Function name	Requirement	Requirement ID
E. Penalties mechanism	[1] the voucher must be deposited 1 ETH to the smart contract to purchase a vouch token.	E1
	[2] voucher is allowed to vouch only vouch token remains more the threshold (50 per cent)	E2
	[3] The penalty is applied by reducing the vouch token on the node that vouches to the node deletion target.	E3
	[4] To reset the token, the voucher must deposit another 1 ETH.	E4

TABLE 4.5: Requirements set for penalties mechanism

4.3.1 Node.js

As a well-known Javascript library runtime, Node.js is an intended highly effective framework for creating scalable network applications. This technology is fundamental in this project since several integrated frameworks require node.js to run as a backend.

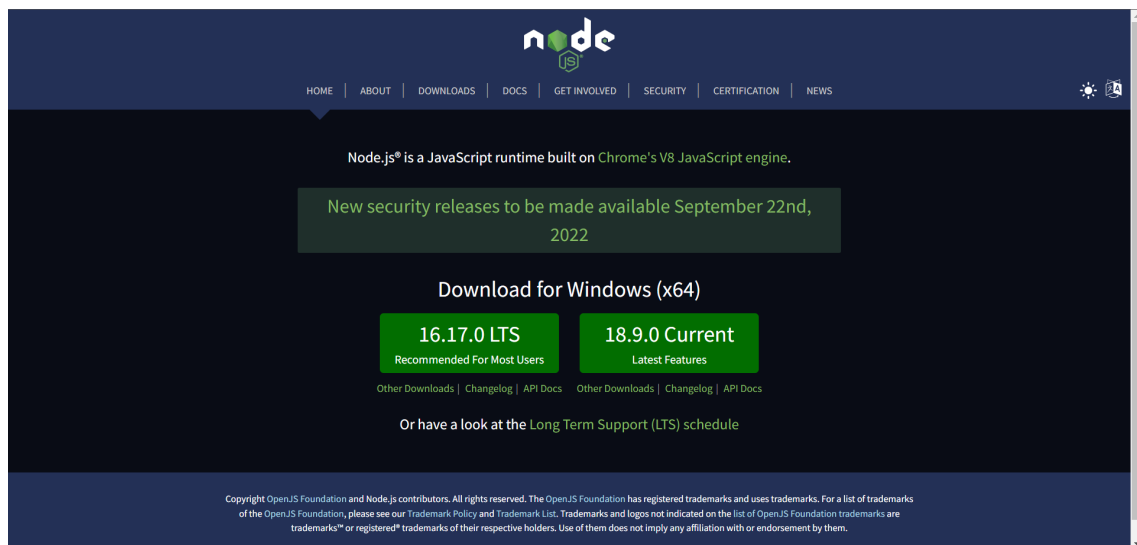
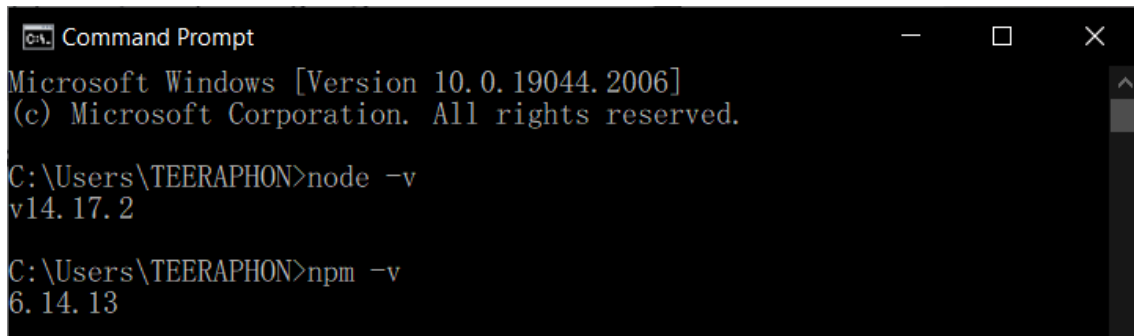


FIGURE 4.7: Node JS official website

There are several ways to install the "node.js" library locally. One noticeably simple way is to download directly from the library's official website (URL: <https://nodejs.org/en/>) as mentioned on Figure 4.7. Once the package was installed, the verification process proceeded through the version check command, as mentioned in the screenshot below.

A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". The text inside the window shows the Microsoft Windows version (10.0.19044.2006) and copyright information. The user has entered two commands: 'node -v' which returns 'v14.17.2', and 'npm -v' which returns '6.14.13'.

```
Microsoft Windows [Version 10.0.19044.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TEERAPHON>node -v
v14.17.2

C:\Users\TEERAPHON>npm -v
6.14.13
```

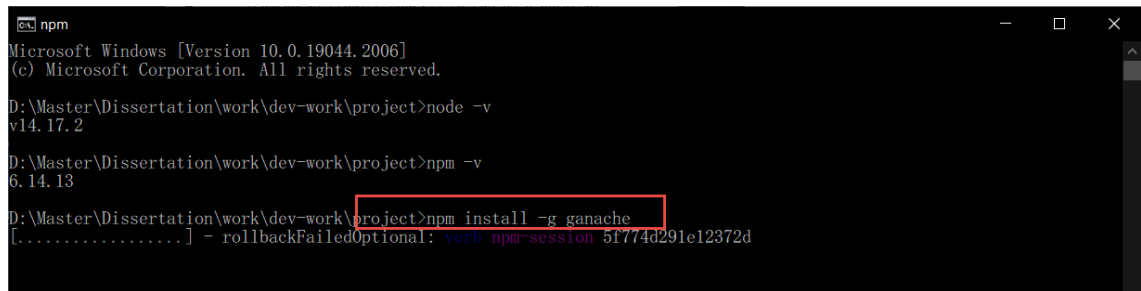
FIGURE 4.8: Node JS installation verification

4.3.2 Ganache

In the overall update and writing of transactions to the Ethereum, there are two separate stages:

- In the First step, a transaction is established and placed in a transaction pool
- The second phase occurs regularly and involves mining all transactions from a transaction pool. Mining entails writing transactions to the Ethereum database or ledger in this context.

If the same procedure were followed for development and testing, it would be a time-consuming undertaking. Ganache, formerly known as TestRPC, was intended to facilitate the creation and testing of smart contracts and solutions on Ethereum. The ganache-cli includes the Ethereum transaction processing and mining capabilities by itself. In addition, there is no default waiting time for the mining of transactions. This is overridable using the Ganache setup options. As the transactions are created, they are recorded. It implies that developers may use ganache-cli as their Ethereum node without requiring mining activity to publish transactions to a distributed ledger. Instead, transactions are recorded as they are made in a ledger. Ganache command-line interface (cli) installation requires Node.js, which must be present in the development environment before Ethereum's smart contract deployment.



```

C:\> npm
Microsoft Windows [Version 10.0.19044.2006]
(c) Microsoft Corporation. All rights reserved.

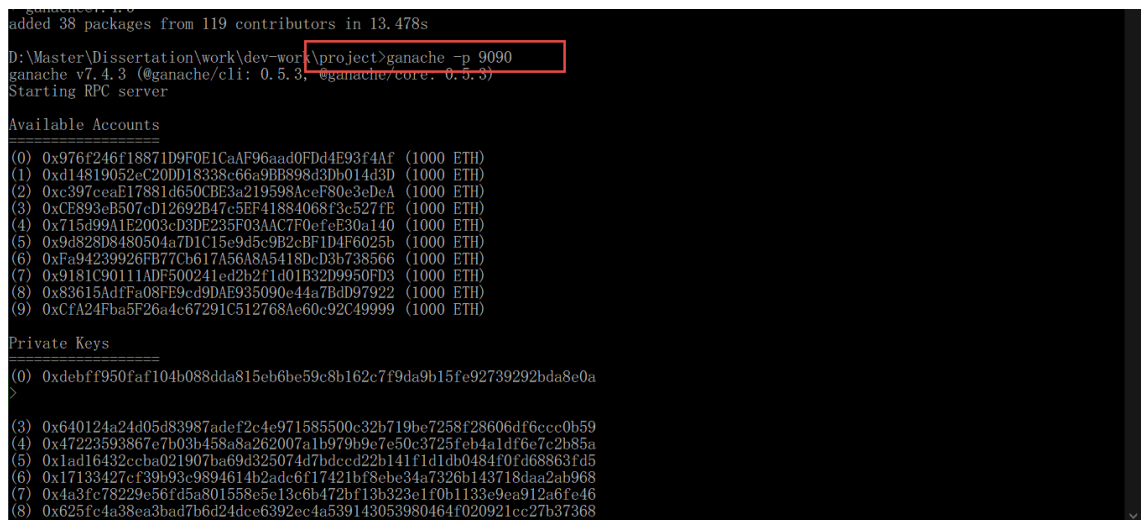
D:\Master\Disertation\work\dev-work\project> node -v
v14.17.2

D:\Master\Disertation\work\dev-work\project> npm -v
6.14.13

D:\Master\Disertation\work\dev-work\project> npm install -g ganache
[.....] - rollbackFailedOptional: verb npm-session 5f774d291e12372d

```

FIGURE 4.9: ganache installation



```

ganache-cli v6.0.0
added 38 packages from 119 contributors in 13.478s

D:\Master\Disertation\work\dev-work\project> ganache -p 9090
ganache v7.4.3 (@ganache/cli: 0.5.3, @ganache/core: 0.5.3)
Starting RPC server

Available Accounts
=====
(0) 0x976f246f18871D9F0E1CaAF96aad0FDd4E93f4Af (1000 ETH)
(1) 0xd14819052ec20DD18338c66a9EB898d3Db014d3D (1000 ETH)
(2) 0xc397ceaE17881d650CBE3a219598Acef80e3cDeA (1000 ETH)
(3) 0xCE893cB507cD12692B47c5EF41884068f3c527fE (1000 ETH)
(4) 0x715499A1E2003cD3DE235F03AAC7F0efeE30a140 (1000 ETH)
(5) 0x94828D8480504a7D1C15e9d5c9B2cBF1D4F6025b (1000 ETH)
(6) 0xFa94239926FB77Cb617A56A8A5418DcD3b738566 (1000 ETH)
(7) 0x9181C90111ADf500241ed2b2f1d01B32D9950FD3 (1000 ETH)
(8) 0x83615AdfFa08FE9cd9DAE935090e44a7BdD97922 (1000 ETH)
(9) 0xCfA24Fba5F26a4c67291C512768Ae60c92C49999 (1000 ETH)

Private Keys
=====
(0) 0xdebff950faf104b088dda815eb6be59c8b162c7f9da9b15fc92739292bda8e0a
>
(3) 0x640124a24d05d83987adef2c4e971585500c32b719be7258f28606df6ccc0b59
(4) 0x47223593867e7b03b458a8a262007a1b979b9e7e50c3725feb4a1df6e7c2b85a
(5) 0x1ad16432ccba021907ba69d325074d7bdcccd22b141f1d1db0484f0fd68863fd5
(6) 0x17133427cf39b93c9894614b2adc6f17421bf8ebe34a7326b143718daa2ab968
(7) 0x4a3fc78229e56fd5a801558e5e13c6b472bf13b323e1f0b1133e9ea912a6fe46
(8) 0x625fc4a38ea3bad7b6d24dce6392ec4a539143053980464f020921cc27b37368

```

FIGURE 4.10: Run Ganache

In order to install ganache-cli, the command "npm install -g ganache" was executed. The result of the installation was depicted on Figure 4.9. The next step, command "ganache -p 9090" was run in the next step for verification and application development purposes. The examples for ganache-cli is shown on Figure 4.10

4.3.3 Remix

The remix is an open-source integrated development environment (IDE) for developers' smart contracts. The platform provides full service of the blockchain development cycle, such as smart contract implementation, plugged-in compiler, deployment, and unit test. In addition, Remix IDE not only provides the entire function for blockchain development but also preserves the simplicity of installation and integration with other necessary tools.



FIGURE 4.11: Remix installation

However, the suggested platform provided several kinds of installation and usage. Firstly, through the browser (URL: <https://remix.ethereum.org/>) as mentioned in Figure 4.11, this option provides an automated development cycle to your smart contract code since it provides auto-compiled function and quick integration to the alternative testing blockchain network.

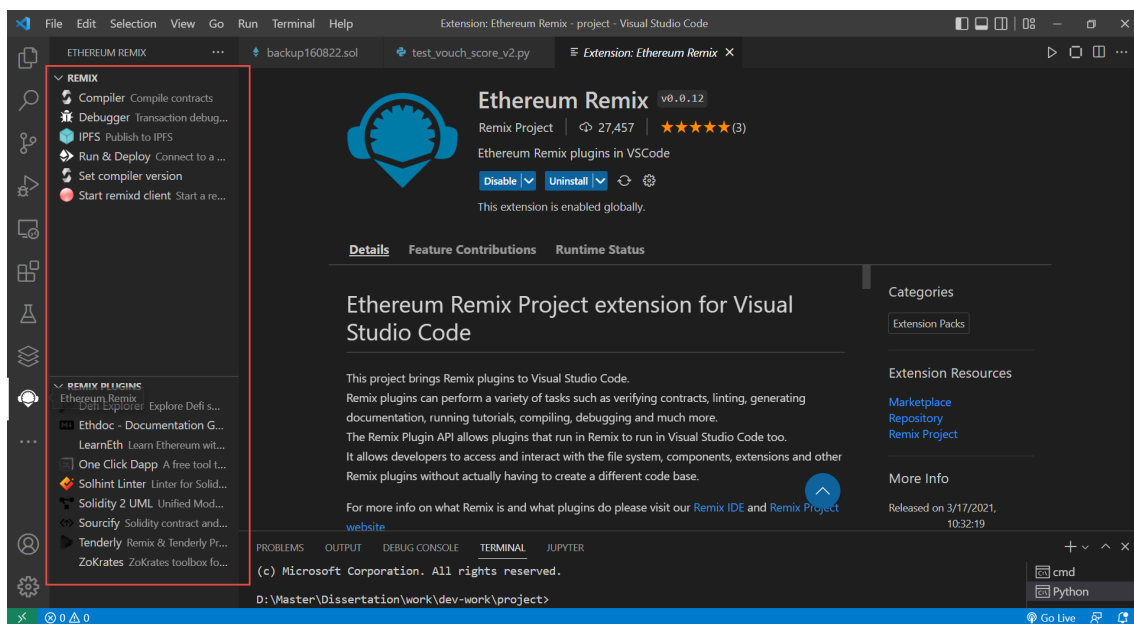


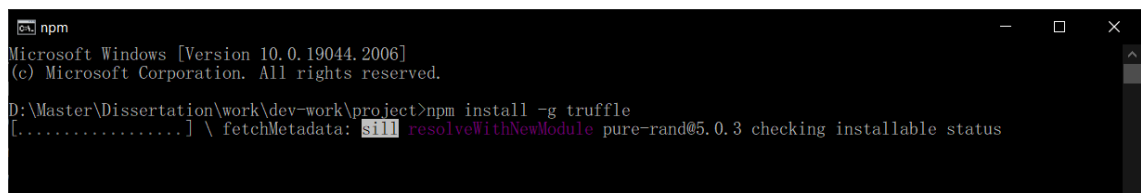
FIGURE 4.12: Remix plugin on vscode

Secondly, the remix is also provided in the plugin "vscode" plugin called "Ethereum Remix." This allows the project to be implemented on local IDE to implement the

smart contract in the sense of online IDE, as depicted in the first option (shown on Figure 4.12). Lastly, the remix also provided their IDE on local- called Remix desktop, which can be downloaded from the official GitHub repository (URL: <https://github.com/ethereum/remix-desktop/releases>). This project mainly uses the remix provided through the browser to develop a contract code.

4.3.4 Truffle

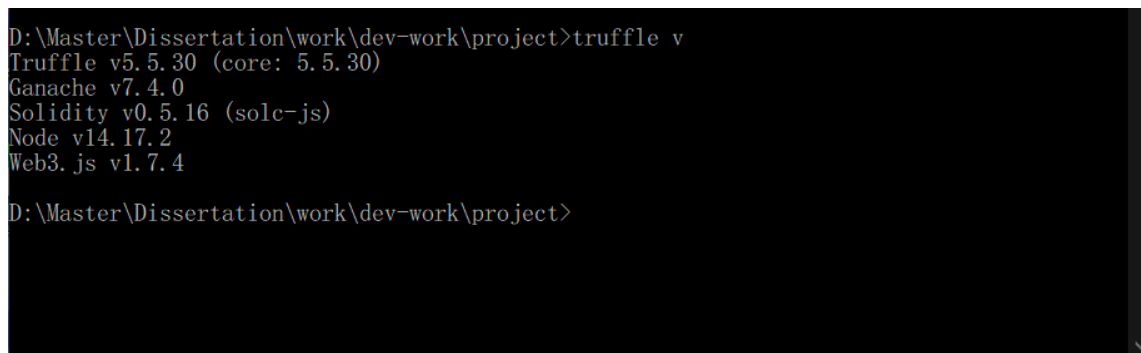
Truffle suite is a platform that provides a testing framework to run unit testing for smart contract development. The platform also provided several built-in functions, for example, a smart contract auditing suite, linking, deployment and binary management. However, automated testing for Ethereum smart contracts for rapid development was the main feature involved in this project.



```
npm
Microsoft Windows [Version 10.0.19044.2006]
(c) Microsoft Corporation. All rights reserved.

D:\Master\Disertation\work\dev-work\project>npm install -g truffle
[.....] \ fetchMetadata: sill resolveWithNewModule pure-rand@5.0.3 checking installable status
```

FIGURE 4.13: Truffle installation



```
D:\Master\Disertation\work\dev-work\project>truffle v
Truffle v5.5.30 (core: 5.5.30)
Ganache v7.4.0
Solidity v0.5.16 (solc-js)
Node v14.17.2
Web3.js v1.7.4

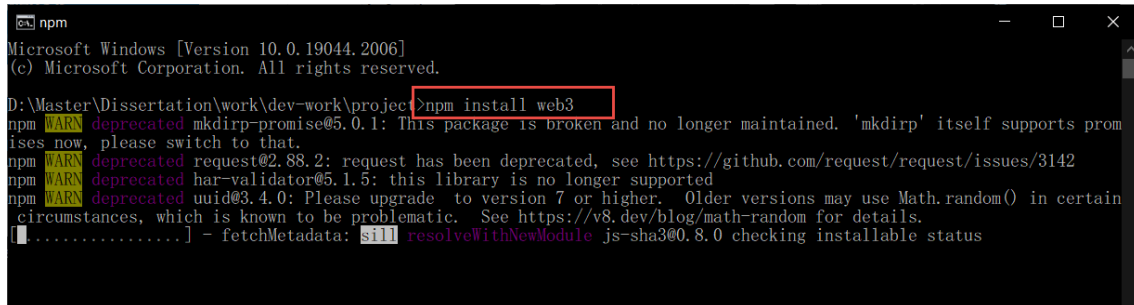
D:\Master\Disertation\work\dev-work\project>
```

FIGURE 4.14: Truffle package verification

As mentioned on Figure 4.13, the command "npm install -g truffle" was used to install the truffle suite to the local workspace and verified the installed package by command "truffle v" as mentioned in Figure 4.16.

4.3.5 Web3

In part of the smart contract interaction - in this case, this project's testing and evaluation process. The package installation using the command "npm install web3" is shown in the picture below.



```

npm
Microsoft Windows [Version 10.0.19044.2006]
(c) Microsoft Corporation. All rights reserved.

D:\Master\Dissertation\work\dev-work\project>npm install web3
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports prom
ises now, please switch to that.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
[.....] - fetchMetadata: sill resolveWithNewModule js-sha3@0.8.0 checking installable status

```

FIGURE 4.15: Web3 package installation

Once the package is installed in the workspace, the connection can perform through the embed code connection as mentioned in the figure of code snap below.



```

1 w3 = Web3(Web3.HTTPProvider("http://127.0.0.1:7545",
  request_kwarg={ "timeout": 1800}))

```

FIGURE 4.16: Web3 package connection

4.4 Algorithm design

The proposed reputation system consists of several components. This section discusses how each defined function on section 4.2 works to execute the contract code that may alter the blockchain's state. In other words, this designed algorithm will use the function's rough workflow in the reputation mechanism. The proposed algorithm consists of five primary functions: made vouch, transaction detaching, node deletion, Pagerank score calculation, and node penalties - which must be considered. However, these algorithms will represent pseudocode to clarify the procedure of the implemented contract code, which might have slightly different details from the developed contract code in terms of technical programming.

4.4.1 Make vouch algorithm

Algorithm 1 Make vouch

Input: $node_{Token} > 50, VouchTo$
Output: N/A
 $VouchTo.isVouched \leftarrow true$
 $VouchFrom \leftarrow msg.sender$
 $VouchTo.vouchBy[address] \leftarrow VouchFrom$
 $VouchTo.TotalInboundLink \leftarrow TotalInboundLink + 1$
 $VouchFrom.TotalOutboundLink \leftarrow TotalOutboundLink + 1$
if $Nodes$ is not contain address of $VouchFrom$ **then**

 ADD $VouchFrom$ to $Nodes$
end if
if $Nodes$ is not contain address of $VouchTo$ **then**

 ADD $VouchTo$ to $Nodes$
end if
while i in $Nodes$ **do**

 PagerankCalculation($Nodes[i]$)

 \triangleright Calculate Pagerank score of each node

end while

According to the requirement in section 4.2.1, a vouch transaction will establish the edges between entities, representing trustworthiness between two nodes. Initially, the contract code is used to verify the remaining token of the peer that intended to create a vouch transaction with the node's address of the transaction destination. The algorithm stores the destination address in a variable named "VouchTo", defined as vouchee entities. In addition, the workflow starts by storing a necessary initial value to verify transaction data, which must align with the rule defined in the requirement shown in table 4.1 such as transaction owners can not make vouch for themselves or someone that already vouch. The function also updated the variables for the proposed utilities, for instance, the total number of each entity's inbound and outbound links. Once all necessary static variables were updated correctly, the algorithm added the transaction entities to the "Nodes" variable, which collects all nodes in the current network if not added. In the final part, after all, the variable was updated reputation score- Pagerank score of nodes are calculated and stored for each entity.

4.4.2 Vouch detaching

Vouch detaching is noticed as the opposite mechanism of making vouch transactions. The algorithm stores the destination address in a variable named "UnvouchTo", defined as vouchee entities proposed to detach the trustworthiness from the voucher. Similarly, the workflow initiated by storing a necessary initial value to verify transaction data must align with the rule defined in the requirement shown in table 4.2 such as transaction owners can not detach the vouch from entities that never vouch before. The function also

updated the variables for the proposed utilities, for instance, the total number of each entity's inbound and outbound links, by decreasing one on each side of entities (voucher who intent to detach vouch and vouchee who lost another inbound link). Once all necessary static variables are updated correctly, the algorithm removes the entities from the "Nodes" variable because a node might lose all inbound and outbound links after the detaching transaction is complete. This mechanism is used to prevent unsatisfied ranking of entries network which cause the ranking calculation that includes node with no neither inbound nor outbound link. Lastly, after the detaching variable is updated, the nodes' reputation score is calculated and updated to the remaining entity.

Algorithm 2 Vouch detaching

Input: *UnvouchTo*
Output: *N/A*

```

UnvouchTo.isVouched  $\leftarrow$  false
UnvouchFrom  $\leftarrow$  msg.sender
UnvouchTo.TotalInboundLink  $\leftarrow$  TotalInboundLink - 1
VouchFrom.TotalOutboundLink  $\leftarrow$  TotalOutboundLink - 1
REMOVE UnvouchTo.vouchBy[UnvouchFrom]
                                 $\triangleright$  Pop vouched list on detaching node
if TotalOutboundLink and TotalInboundLink of UnvouchTo = 0 then
  REMOVE UnvouchTo from Nodes
end if
if TotalOutboundLink and TotalInboundLink of UnvouchFrom = 0 then
  REMOVE UnvouchFrom from Nodes
end if
while N in Nodes do
  PagerankCalculation(Nodes[i])            $\triangleright$  Calculate Pagerank score of each node
end while

```

4.4.3 Node deletion

The destruction of misbehaving entities can proceed through the node deletion mechanism. The function can proceed through the user with admin privilege - contract owner. This mechanism might be denoted as the extended version of the edges removal algorithm, which must remove the link between the outbound and inbound node of the deletion node. The algorithm starts by iterating through the list of an outbound node of the deletion target node; all related variable was updated, such as the count of an inbound link that which deleted node point to that entity. Similarly, a node that links to misbehaved peer must be updated, the edges that point to the target node of deletion must be modified, and penalties be applied to these nodes in the inbound node list.

Once all parameters were used, following the deletion mechanism, the Pagerank score was also recalculated and published to the system chain.

Algorithm 3 Node deletion

Input: *TargetNode*

Output: *N/A*

```

while Outbound node of TargetNode do
    TotalInboundLink  $\leftarrow$  TotalInboundLink - 1
    if TotalOutboundLink and TotalInboundLink of node = 0 then
        REMOVE node from Nodes
    end if
end while

while Inbound node of TargetNode do
    TotalOutboundLink  $\leftarrow$  TotalOutboundLink - 1
    if TotalOutboundLink and TotalInboundLink of node = 0 then
        REMOVE node from Nodes
    end if
    NodesPunishment(node)                                 $\triangleright$  call Apply penalties algorithm
end while
REMOVE TargetNode from Nodes
while N in Nodes do
    PagerankCalculation(Nodes[i])                         $\triangleright$  Calculate Pagerank score of each node
end while
  
```

4.4.4 Pagerank calculation

According to the mentioned equation, the algorithm receives the node as the input, which represents the mentioned equation 2.1, which represents the Pagerank computation formula. Initially, the damping factor is declared and set to 85; since the solidity does not support the decimal number variable type, 85 represents 85 per cent or 0.85. Once static parameters such as the damping factor are initiated, the algorithm will generate the initial value of the Pagerank score in case the node is newly added to the network. After all, nodes have an initial score; the contract will iterate the inbound node list of each entity and accumulate the fraction of individual ranking score divided by the number of outbound links as explained on 2.1. However, after getting the accumulated score, the contract will complete the computation by multiplying the damping factor and storing the calculated score in the chain in the final step, as shown in the below algorithm.

Algorithm 4 Pagerank Calculation**Input:** *Node***Output:** *N/A*

```

 $d \leftarrow 85$  ▷ Set dumping factor to 0.85 or 85
if Node not have rank score then
     $PagerankScore \leftarrow 1/N$  ▷ Assign init. value to each node
end if
while 0 to 99 do
    while Inbound node of Node do
         $PR_{temp} \leftarrow PR_{temp} + PagerankScore / NumberOfOutboundLink$ 
    end while
     $Pagerank_{node} \leftarrow (1 - d) + d * PR_{temp}$  ▷ Refer to eq. 2.1
end while

```

4.4.5 Apply penalties**Algorithm 5** Apply penalties**Input:** *node***Output:** *N/A*

```

 $cost \leftarrow 1/N$  ▷ Refer to eq. 4.1
 $node_{Token} \leftarrow node_{Token} - cost$ 

```

As mentioned in the previous section (4.2.5), the penalties algorithm might be associated with the vouch function. As part of the token deduction penalties, the portion at each specific time will depend on the number of entities in the system as depicted in algorithm 5. The cost of the token was also calculated based on the current number of entities in the current network, as mentioned on algorithm 5.

4.5 Implementation guide and techniques

Implementing a smart contract might denote slightly different details between the designed algorithm and the actual code. In other words, in terms of development, some technical technique might be applied in order to allow the contract to perform collectively and align with the declared requirements. In order to accomplish the system goal - to offer the reputation system to the user, there are several techniques in part of contract development which must be taken into account. In addition, this section also mentions the implementation guide on each function and provides a descriptive analysis of the implemented contract. However, this section might discuss some essential parts of the developed contract code that might be necessary for re-producing or contributing to this work in the future.



```

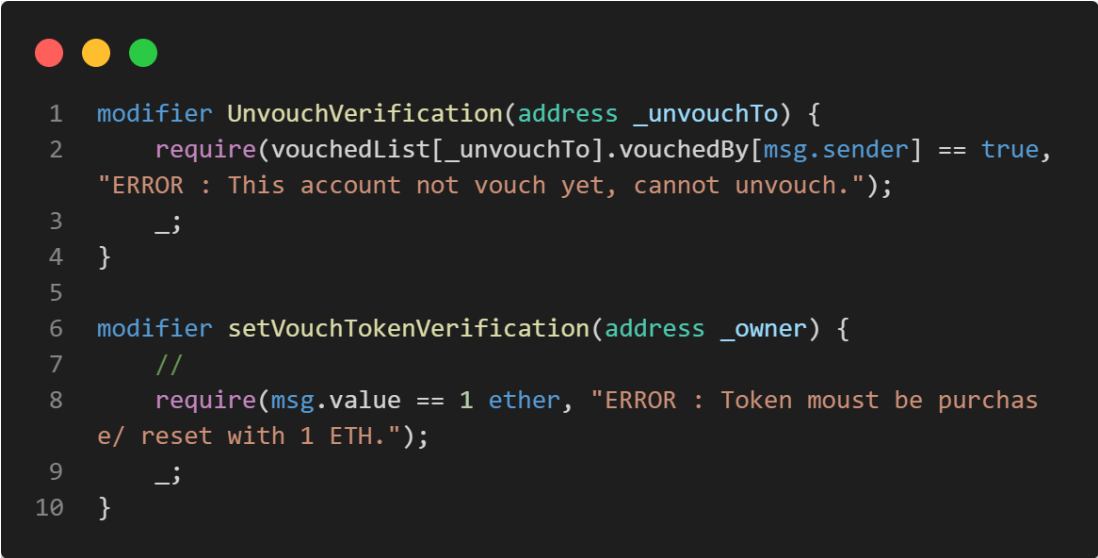
1  struct VouchDetails {
2      mapping(address => bool) vouchedBy;
3      mapping(address => bool) vouchedTo;
4      address[] vouchedByList;
5      address[] vouchedToList;
6      bool isVouched;
7      bool isAdded;
8      bool isinitVouchtoken;
9      uint total_inbound_link;
10     uint total_outbound_link;
11     uint prScore;
12     uint ranking;
13     uint vouch_token;
14 }
15
16 mapping(address => VouchDetails) public vouchedList;

```

FIGURE 4.17: Structure of entities


As mentioned on Figure 4.18, the primary variable for the contract is denoted as structure mapping. The mapping variable of each address in the system is a map to its details called the "VouchDeatils" structure. The structure collects the necessary value and parameter of each entity in the network; for instance, the mapping and array list of inbound and outbound entities necessary for related mechanisms such as ranking score calculation, node deletion, and edge removal. Additionally, another integer and boolean variable also store inside the structure. This parameter also provides the utilities to enhance some aggregation functions, for example, the count of each peer's total inbound and outbound links or the current Pagerank score. The structure also included entities' token values to represent that node's vouch power.

Furthermore, the modifier and require function was used to perform as the verification function instead of typical "if-else" verification. The "require" function is used to verify the contract transaction in case the condition of the transaction is true; the transaction was allowed. On the other hand, if the condition is returned as false, the requested transaction is also rejected and returns a specific error message. As mentioned below screenshot, the modifier gathers all validity statements of each functional such as vouch making, vouch detaching, and token purchasing. This verification and state parameter related to each function is verified to ensure that the contract will perform adequately.



```
1  modifier UnvouchVerification(address _unvouchTo) {
2      require(vouchedList[_unvouchTo].vouchedBy[msg.sender] == true,
3      "ERROR : This account not vouch yet, cannot unvouch.");
4  }
5
6  modifier setVouchTokenVerification(address _owner) {
7      //
8      require(msg.value == 1 ether, "ERROR : Token must be purchased/ reset with 1 ETH.");
9      _;
10 }
```

FIGURE 4.18: Modifier of unvouch and token verification



```
1  modifier VouchVerification(address _vouchTo) {
2      //Check the node can vouch once on each account
3      require(vouchedList[_vouchTo].vouchedBy[msg.sender] =
4      = false, "ERROR : This account already vouched by you.");
5      //Check the node can not vouch on themselves
6      require(_vouchTo != msg.sender, "ERROR : Not allow to
7      vouch yourself.");
8      //check the vouch token initialize
9      require(vouchedList[msg.sender].isinitVouchtoken == t
10     rue, "ERROR : Vouch token does not init or reset.");
11     //check right to make vouch
12     require(vouchedList[msg.sender].vouch_token > 50, "ER
13     ROR : Not enough vouch token.");
14     _;
15 }
```

FIGURE 4.19: Modifier of vouch verification

Pagerank calculation: In the part of the calculation, the contract development might require some specific arithmetic technique to generate the acceptable accumulated result since solidity might not support decimal number variable type. Although the solidity in the current version (greater than 8.0) provided a fixed point variable in the contract implementation, this variable is not fully supported. In other words, the fixed point variable might be declared by it not allowed to be assigned. As mentioned in the figure below, the ranking score calculation might be kept in integer form by scaling up the

factor and denominator to get the decimal result in integer format; for example, if the expected result is 0.75, the contract will keep the result as 75 instead. The screenshot below represents the function of Pagerank calculation, which is when the node and edges were created, modified, and deleted. The function receives the array list of a node in the network and the total number on a current node as an initial parameter and calculates the ranking score on each node.



```

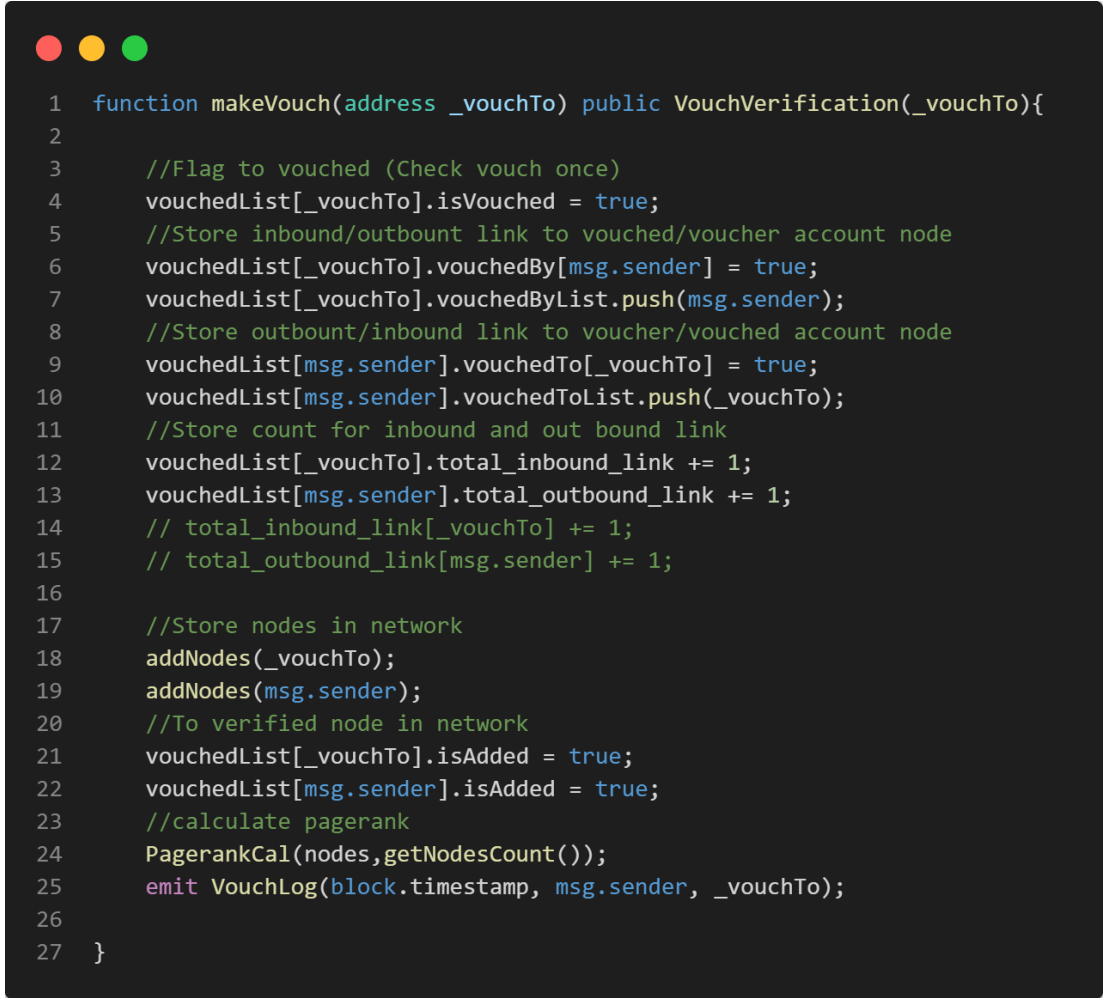
1  function PagerankCal(address[] memory _nodes, uint _n) internal {
2      //Damping factor
3      uint _alpha = 85;
4
5      Init_score(_nodes, _n);
6      //PageRank : PR(A) = 1 - d + d*(PR(Tn)/C(Tn)+....+...)
7      uint nodesLength = _nodes.length;
8
9      //Secure iteration for let score reach convergence
10     for(uint k = 0; k < 100; k++){
11
12         for(uint i = 0; i < nodesLength; i++) {
13             uint temp_score_sum;
14             if(vouchedList[_nodes[i]].total_inbound_link == 0) {
15                 // In case that node not have incoming link
16                 vouchedList[_nodes[i]].prScore = 100 - _alpha;
17             }else{
18                 for(uint j = 0; j < vouchedList[_nodes[i]].vouchedByList.l
19                     ength; j++){
20                     //Calculate sigma inbound node (PR(Tn)/C(Tn)+....+..)
21                     address temp_inbound_node = vouchedList[_nodes[i]].vou
22                     chedByList[j];
23                     require(vouchedList[temp_inbound_node].total_outbound_
24                     link != 0, "ERROR : Vouched account not have outbound link (Impossible) ==
25                     BUG");
26                     temp_score_sum += (vouchedList[temp_inbound_node].prSc
27                     ore) / vouchedList[temp_inbound_node].total_outbound_link;
28                 }
29
30                 // pr = (1 - d) + d(E PR(T)/C(T))
31                 temp_score_sum = ((100 - _alpha) * 100) + ((_alpha) * temp

```

FIGURE 4.20: Pagerank calculation function code snapshot

Make vouch: The next important part is the vouch function which extends the modifier "Vouchverification." which aims to change the make vouch function behaviour and perform as a verifier when entities initiate a vouch transaction. The vouch function starts by updating the flag and numeric parameter to complete the vouch transaction on both voucher and vouchee, as shown in the screenshot below. Once the parameter such as "vouchby" list, "vouchto" list, count of the inbound, and count of outbound on both peers are updated, the function will call add the node if it does not exist in

nodes list which represents the current node in the network. Lastly, the function calls the Pagerank calculation function to update all ranking scores of nodes in the system.




```

1  function makeVouch(address _vouchTo) public VouchVerification(_vouchTo){
2
3      //Flag to vouched (Check vouch once)
4      vouchedList[_vouchTo].isVouched = true;
5      //Store inbound/outbound link to vouched/voucher account node
6      vouchedList[_vouchTo].vouchedBy[msg.sender] = true;
7      vouchedList[_vouchTo].vouchedByList.push(msg.sender);
8      //Store outbound/inbound link to voucher/vouched account node
9      vouchedList[msg.sender].vouchedTo[_vouchTo] = true;
10     vouchedList[msg.sender].vouchedToList.push(_vouchTo);
11     //Store count for inbound and out bound link
12     vouchedList[_vouchTo].total_inbound_link += 1;
13     vouchedList[msg.sender].total_outbound_link += 1;
14     // total_inbound_link[_vouchTo] += 1;
15     // total_outbound_link[msg.sender] += 1;
16
17     //Store nodes in network
18     addNodes(_vouchTo);
19     addNodes(msg.sender);
20     //To verified node in network
21     vouchedList[_vouchTo].isAdded = true;
22     vouchedList[msg.sender].isAdded = true;
23     //calculate pagerank
24     PagerankCal(nodes, getNodesCount());
25     emit VouchLog(block.timestamp, msg.sender, _vouchTo);
26
27 }

```

FIGURE 4.21: Make vouch function code snapshot

Vouch detaching: Unvouch function or vouch detaching is noticed as the revert mechanism of the vouch-making function. As mentioned in the following screenshot, the unvouch verification extends the modifier "UnvouchVerification", which checks the vouch detaching transaction to align with the requirements. Firstly, the function also contemporise the related parameter on both entities, such as the "vouchBy" and "vouchTo" lists, a count of the inbound or outbound link which must be reduced when edges are removed. However, some further steps must be included in the removal function. The function also iterated on each entity of detaching; when both inbound and outbound like do not exist, the function also removes the node from the current system. After all the parameters are appropriately updated, the function will call the Pagerank calculation function to recalculate the ranking score on each entity in the system.



```

1  function unvouched(address _unvouchTo) public UnvouchVerification(_unvouch
   To){
2
3      //Flag to vouched (Check vouch once)
4      vouchedList[_unvouchTo].isVouched = false;
5      //Store inbound link to vouched account node
6      vouchedList[_unvouchTo].vouchedBy[msg.sender] = false;
7      //vouchedList[_unvouchTo].vouchedByList.pop(msg.sender);
8      //Update array list of inbound link of unvouched node and outbound of
   unvoucher node
9      uint len = vouchedList[_unvouchTo].vouchedByList.length;
10     for(uint i = 0; i < len ; i++){
11         if(vouchedList[_unvouchTo].vouchedByList[i] == msg.sender){
12             delete vouchedList[_unvouchTo].vouchedByList[i];
13         }
14     }
15     uint len_to = vouchedList[msg.sender].vouchedToList.length;
16     for(uint j = 0; j < len_to ; j++){
17         if(vouchedList[msg.sender].vouchedToList[j] == _unvouchTo){
18             delete vouchedList[msg.sender].vouchedByList[j];
19         }
20     }
21     //Store outbound link to voucher account node
22     vouchedList[msg.sender].vouchedTo[_unvouchTo] = false;
23     //Store count for inbound and out bound link
24     vouchedList[_unvouchTo].total_inbound_link -= 1;
25     vouchedList[msg.sender].total_outbound_link -= 1;

```

FIGURE 4.22: Vouch detaching function code snapshot

Node deletion: The deletion function separates the logic into two sections. First, the function will perform an iterate through the node that points from a target node of deletion. Each variable of an outbound node from the target node will be updated as mentioned below. The variable in the node which targets entities given vouch on previously is modified in the same approach of edges removal between the target node and outbound node - vouchee(s). Similarly, the inbound node(s) to the target node is also included in the execution scope of the deletion function. The function will update variables, for instance, count on an outbound link on each entity that points to the deleted entity.

Node penalties: Once the variable modification stage in the deletion function was completed. The inbound entities list - as a voucher of the deleted node, must apply the penalties as they depict the voucher of misbehaved entities. The penalties function will receive the number of the current node in the system and the voucher address as input. The function will calculate the token cost and deduce the cost to the input node, as depicted in the figure below (Node penalties function code snapshot).

```

1  function delete_node(address _targetnode) public {
2
3      //Outbound update from target node (Target_node --> Node A)
4      uint len = vouchedList[_targetnode].vouchedToList.length;
5      for(uint i = 0; i < len ; i++){
6          address out_targetnode = vouchedList[_targetnode].vouchedToList[i];
7          vouchedList[out_targetnode].vouchedBy[_targetnode] = false;
8          //Update inbound link array which target node vouch to
9          uint len_inlink = vouchedList[out_targetnode].vouchedByList.length;
10         for(uint j = 0; j < len_inlink; j++){
11             if(vouchedList[out_targetnode].vouchedByList[j] == _targetnode){
12                 delete vouchedList[out_targetnode].vouchedByList[j];
13             }
14         }
15         //Update inbound count parameter of node that target node vouch to
16         vouchedList[out_targetnode].total_inbound_link -= 1;
17         //Check if node must be delete out from network
18         if(vouchedList[out_targetnode].total_inbound_link == 0 && vouchedList[out_targetnode].total_outbound_link == 0){
19             //nodes.pop(_unvouchTo);
20             uint len_nodes = nodes.length;
21             for(uint k = 0; k < len_nodes ; k++){
22                 if(nodes[k] == out_targetnode){
23                     delete nodes[k];
24                 }
25             }
26             vouchedList[out_targetnode].isAdded = false;
27         }
28     }
}

```

FIGURE 4.23: Node deletion function code snapshot

```

1  function nodes_punishment(address _node, uint _allnodesnum) internal {
2      uint punishment_cost = 100 / (_allnodesnum - 1);
3      vouchedList[_node].vouch_token -= punishment_cost;
4  }

```

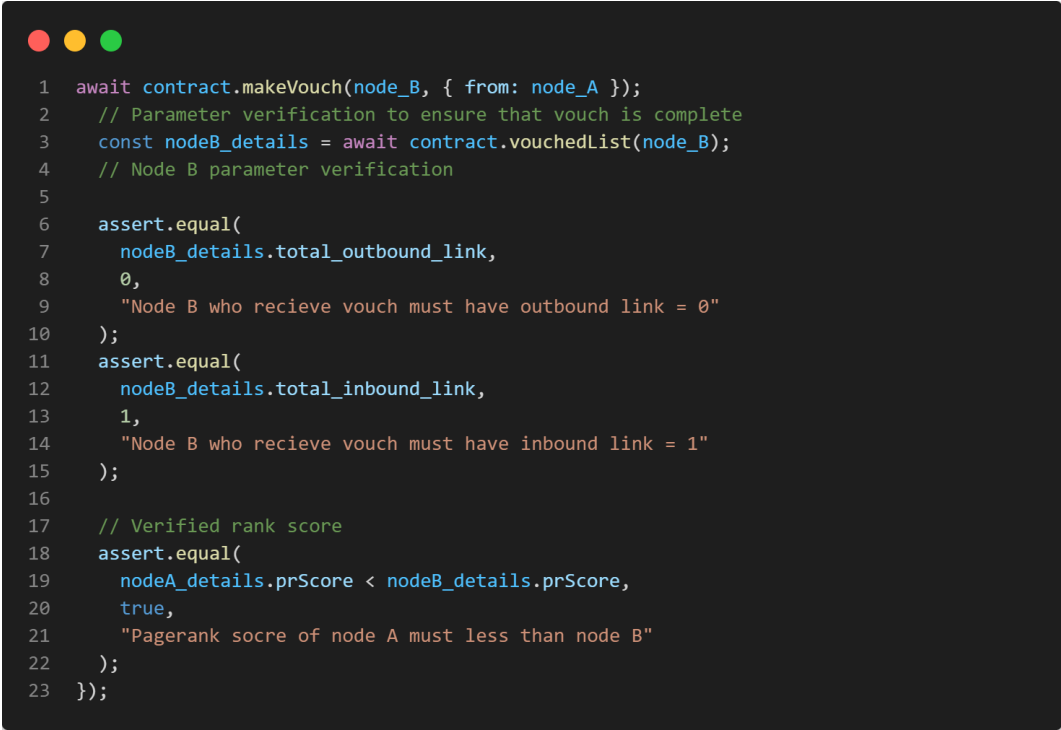
FIGURE 4.24: Node penalties function code snapshot

Chapter 5

Test and Evaluation

5.1 Unit testing

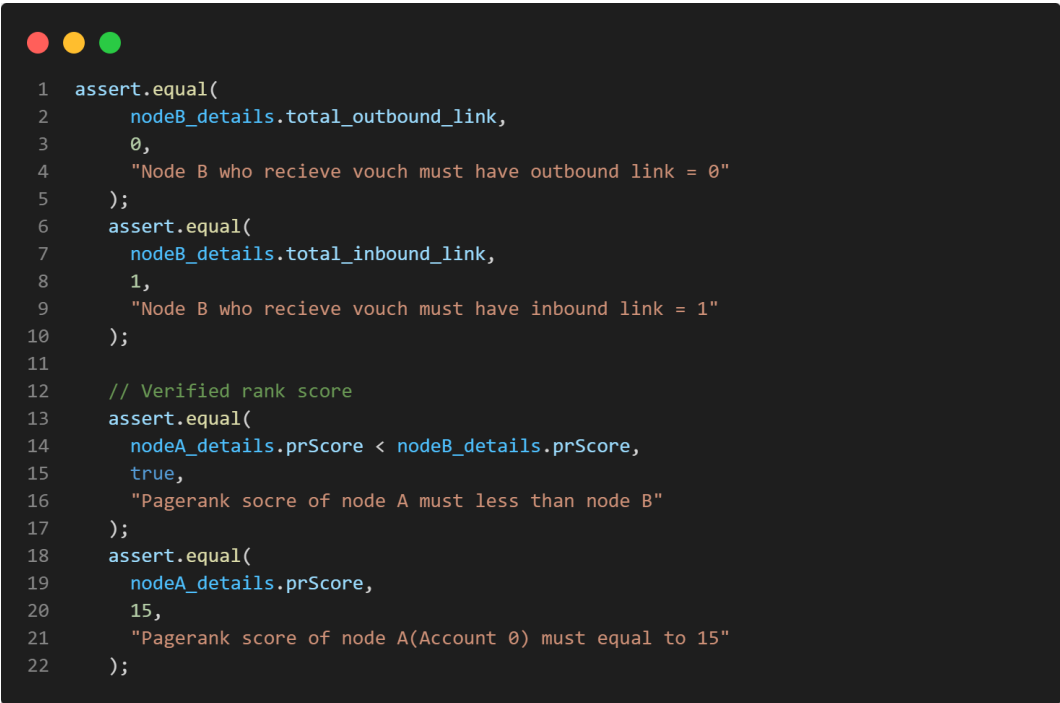
The unit testing for this project is integrated with the "truffle" test framework and the "web3" framework to interact with the deployed smart contract of the proposed reputation system. The unit test is divided into five central parts, categorised along with system requirements - make vouch function, vouch detaching, node deletion, Pagerank score calculation, and penalties mechanism. The individual sub-section of the testing unit will measure and verify the related variable on each function, which is essential in fulfilling the function requirement.



```
1  await contract.makeVouch(node_B, { from: node_A });
2  // Parameter verification to ensure that vouch is complete
3  const nodeB_details = await contract.vouchedList(node_B);
4  // Node B parameter verification
5
6  assert.equal(
7    nodeB_details.total_outbound_link,
8    0,
9    "Node B who recieve vouch must have outbound link = 0"
10 );
11 assert.equal(
12   nodeB_details.total_inbound_link,
13   1,
14   "Node B who recieve vouch must have inbound link = 1"
15 );
16
17 // Verified rank score
18 assert.equal(
19   nodeA_details.prScore < nodeB_details.prScore,
20   true,
21   "Pagerank socre of node A must less than node B"
22 );
23 };
```

FIGURE 5.1: Unit test on vouch function

The Figure 5.1 depicts the part of code snap to verify the **make vouch** function. The testing started by asserting the vouch token, which must be purchased on each transaction. As mentioned in section 4.2.1, the vouch function must be related to the penalties mechanism as the vouch must allow in case the "vouch token" exceeds fifty per cent, which represents the entity's vouching power. Next stage, the parameters related to the vouch mechanism, such as inbound and outbound links, are involved in the unit test process by measuring the expected value. However, the vouch function also recalculates the entities' centralities score. The voucher must be less critical because the trustworthiness was transferred to another person, and the system consists of two entities. On the other hand, **vouch detaching** - the reverse step for vouch transaction, the test procedure starts with the same parameters (for vouch and unvouch). The variable includes the count of inbound and outbound links, which are noticed as important parameters since these are used to calculate the Pagerank score. The system might generate unsatisfied results if the count and list of inbound and outbound entities are inappropriately updated. Additionally, the unvouch unit test might need to verify the edged removal between both entities; in other words, a related parameter representing the link must be updated decreasingly. However, the ranking score of test set entities might be verified following with declared test scenario.



```

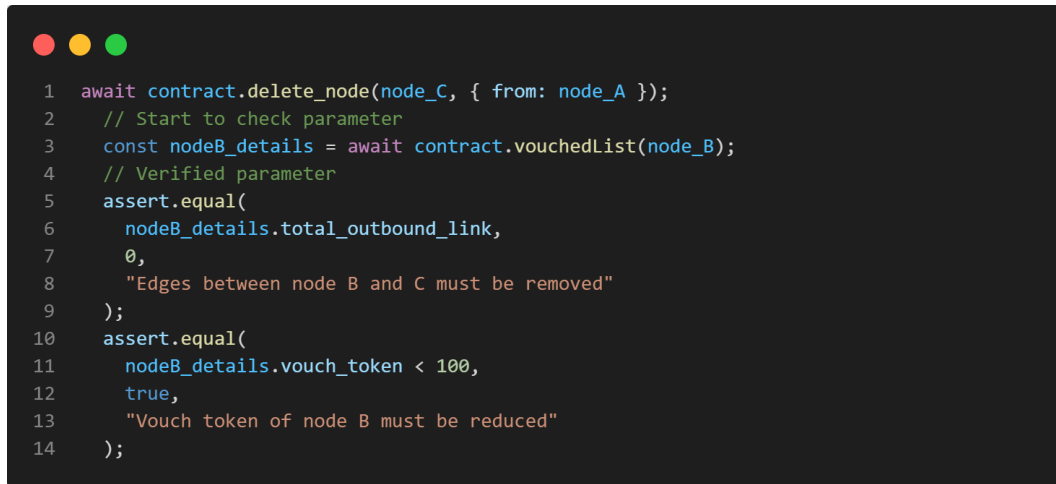
1  assert.equal(
2      nodeB_details.total_outbound_link,
3      0,
4      "Node B who recieve vouch must have outbound link = 0"
5  );
6  assert.equal(
7      nodeB_details.total_inbound_link,
8      1,
9      "Node B who recieve vouch must have inbound link = 1"
10 );
11
12 // Verified rank score
13 assert.equal(
14     nodeA_details.prScore < nodeB_details.prScore,
15     true,
16     "Pagerank socre of node A must less than node B"
17 );
18 assert.equal(
19     nodeA_details.prScore,
20     15,
21     "Pagerank score of node A(Account 0) must equal to 15"
22 );

```

FIGURE 5.2: Unit test on pagerank calculation function

Moreover, **Pagerank calculation** testing uses the same test scenario with making vouch. As mentioned, Figure 5.2 measures the variable and the exact Pagerank score that the contract generated based on the test case of make vouch. In order to emphasise the scenario, node A makes the vouch transaction to node B, and the number of inbound

and outbound entities of A must be zero and one, respectively; contrastingly, node B will have one inbound and zero outbound count value. Similarly, the node A score as the voucher must be less than node B when the system consists only of nodes A and B.



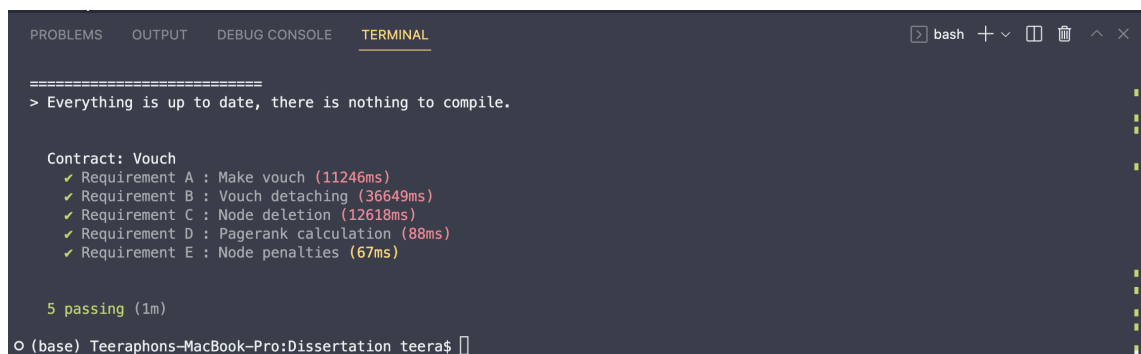
```

1  await contract.delete_node(node_C, { from: node_A });
2  // Start to check parameter
3  const nodeB_details = await contract.vouchedList(node_B);
4  // Verified parameter
5  assert.equal(
6    nodeB_details.total_outbound_link,
7    0,
8    "Edges between node B and C must be removed"
9  );
10 assert.equal(
11   nodeB_details.vouch_token < 100,
12   true,
13   "Vouch token of node B must be reduced"
14 );

```

FIGURE 5.3: Unit test on node deletion function

In the case of **node deletion** function, the extension of the edge removal mechanism, the node representing the deletion target might consist of a link to another entity; this link must be taken into account as part of the link detachment. The testing process for the removing function begins with the variable assessment on both incoming and outgoing entities listed from the target node as mentioned in Figure 5.3. In addition, the Pagerank score and the token of the related nodes (vouchers who vouch for the deleted node) in the test scenario also got verified. However, the token must be verified with an extra step to measure the remaining token to fulfill and ensure that the **apply penalties** function was appropriately performed since it must be called while the deletion is executing. These mentioned test cases are used to verify the execution process of all five primary requirements; the automated unit test result is shown below as a screenshot.



```

=====
> Everything is up to date, there is nothing to compile.

Contract: Vouch
✓ Requirement A : Make vouch (11246ms)
✓ Requirement B : Vouch detaching (36649ms)
✓ Requirement C : Node deletion (12618ms)
✓ Requirement D : Pagerank calculation (88ms)
✓ Requirement E : Node penalties (67ms)

5 passing (1m)
O (base) Teeraphons-MacBook-Pro:Dissertation teera$

```

FIGURE 5.4: Unit testing result screenshot

5.2 Performance

Performance test environment			
Parameters	Environment 1	Environment 2	Environment 3
Maximum nodes (N)	10	50	100
Random degree	Low	Medium	High
Hard fork	MUIRGLACIER	MUIRGLACIER	MUIRGLACIER
Gas limit	2×10^{10}	2×10^{10}	2×10^{10}
Gas Price	2×10^{10}	2×10^{10}	2×10^{10}
Init. Balance	1000 ETH	1000 ETH	1000 ETH

TABLE 5.1: Performance test environment configuration details

The environment for this performance testing can be separated into three different sets. These three simulated networks are created with a maximum number of nodes in the system, starting with (1) ten nodes, followed by (2) fifty nodes, and (3) one hundred nodes which is the maximum capacity of the framework. Regardless, each environment is used in different randomness degrees of transaction behavior to reflect the real-world network. Initially, in the smallest network (Environment 1), which consists of the minimum number of entities, the simulation of transaction behavior depicts zero degrees of randomness by all nodes initiating a transaction to every entity in the system. Although this simulation might not represent the real-world situation since it consists of a small set of entities, the network might reveal the maximum capacity of small-network simulation. Figure 5.2: Performance evaluation

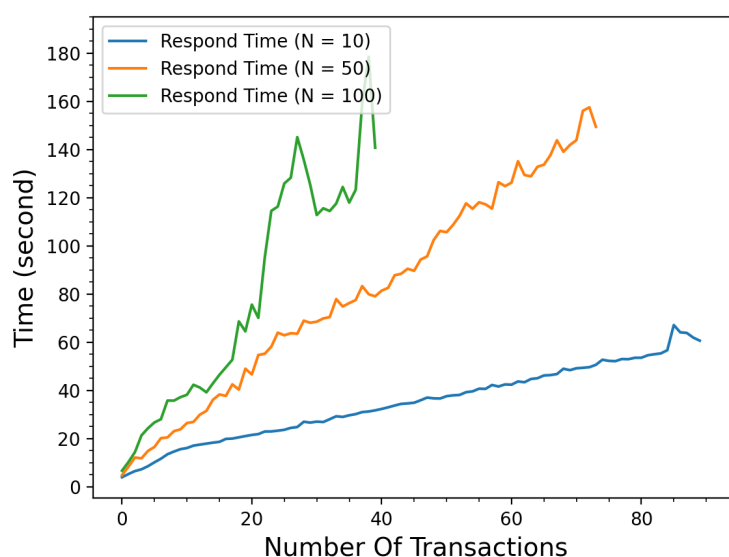


FIGURE 5.5: Performance evaluation

On the other hand, compared to the first network, the second and the third environment simulated a more realistic behavior in the network. In the second environment, each node randomly creates one or two transactions for another entity. Furthermore, in the third environment (environment 3), random entities will make one or two transactions with another peer. This simulation depicts the complete freedom to allow random entities to make the transaction with another, as mentioned in table 5.1.

The Figure 5.5 illustrated the responding time on each transaction interval of each environment. Overall, the response time of each simulation environment with ten, fifty, and one hundred nodes are linearly rising over the network's number of transactions. Although the transaction time of all simulation sets might not initially depict a significant increasing step, the noticeable distinction is depicted when the number of transactions rises over twenty. The response time on simulation with fifty maximum entities is approximately doubled from the environment with ten nodes, whereas the response time of the transactions on environment 3 is four-time more. Each simulation's performance or response duration might reach one to three minutes. This might reflect that the accumulated number of transactions in the system can affect the response time of each transaction, while the number of maximum entities in the environment may affect the slope of the graph, which depicts the capacity of computation performance on each network (how fast that performance will decrease). In other words, the number of maximum nodes in the network can dramatically decrease the system performance.

5.3 Results

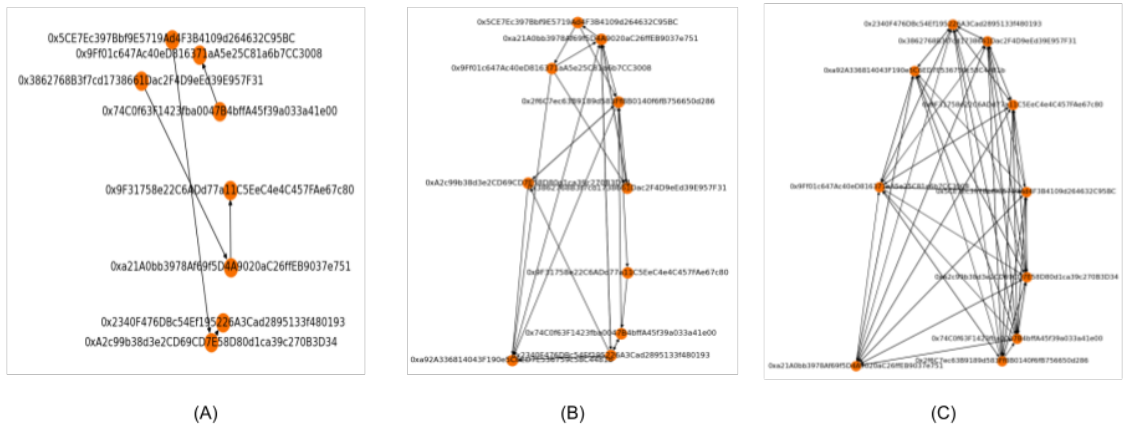


FIGURE 5.6: Sample graph for testing

Results (Graph A edges = 5)			
Id	Address	Score (Smart con- tracts)	Score (Nor- mal)
1	0x2f6C7ec63B9189d583Ff8B0140f6fB756650d286	0	0
2	0xa92A336814043F190e5C6ED7E536759c58C4481b	0	0
3	0x74C0f63F1423fba0047B4bffA45f39a033a41e00	15	0.073
4	0x5CE7Ec397Bbf9E5719Ad4F3B4109d264632C95BC	15	0.073
5	0x3862768B3f7cd1738661Dac2F4D9eEd39E957F31	15	0.073
6	0xA2c99b38d3e2CD69CD7E58D80d1ca39c270B3D34	27	0.135
7	0x9Ff01c647Ac40eD816371aA5e25C81a6b7CC3008	27	0.135
8	0xa21A0bb3978Af69f5D4A9020aC26ffEB9037e751	27	0.135
9	0x9F31758e22C6ADd77a11C5EeC4e4C457FAe67c80	37	0.187
10	0x2340F476DBc54Ef195226A3Cad2895133f480193	37	0.187

TABLE 5.2: Result of rank and score of graph A

The resulting measurement is conducted in the same environment of the Ethereum network. The experiment aims to benchmark the ranking and reputation score generated from the proposed reputation system and the standard external network analysis library to compute the ranking through the Pagerank algorithm. The library involved in this experiment is called "networkx", as it provided the built-in feature for calculating the Pagerank score of the directed graph [5]. The experiment process starts by randomly creating the node and edges across the entities in the simulation environment with a maximum node of ten. The process will call the vouch function from the deployed contract to get the ranking score called "Score (Smart-Contracts)". The score generated via the implemented contract is used to benchmark with the score generated from the python library - called score (Normal)- to explore and evaluate the smart contract's computation result. However, the evaluation uses three different graphs with a random number of edges, as mentioned on Figure 5.6 A, B, and C. The sample graph consists of a graph with five, twenty-four, and seventy, respectively. Each sample will reproduce the same evaluation process as mentioned above. However, the result illustrated in 5.2 depicts that the contract performs well in the part of ranking result computation, even though the score (Smart-contracts) might denote a difference from the score generated by a standard library function. However, the simulated network has the same ranking order of nodes. The result of an experiment in graphs B and C showed below also emphasises this assumption that the suggested system might give an accurate ranking result to reveal the most trustable - importance entities by ordering in the small network.

Results (Graph B edges = 24)			
Id	Address	Score (Smart con- tracts)	Score (Nor- mal)
1	0x9F31758e22C6ADd77a11C5EeC4e4C457FAe67c80	43	0.046
2	0x9Ff01c647Ac40eD816371aA5e25C81a6b7CC3008	44	0.047
3	0xA2c99b38d3e2CD69CD7E58D80d1ca39c270B3D34	65	0.070
4	0x5CE7Ec397Bbf9E5719Ad4F3B4109d264632C95BC	71	0.076
5	0x74C0f63F1423fba0047B4bffA45f39a033a41e00	72	0.078
6	0x3862768B3f7cd1738661Dac2F4D9eEd39E957F31	83	0.090
7	0x2340F476DBc54Ef195226A3Cad2895133f480193	101	0.109
8	0xa92A336814043F190e5C6ED7E536759c58C4481b	104	0.113
9	0x2f6C7ec63B9189d583Ff8B0140f6fB756650d286	137	0.148
10	0xa21A0bb3978Af69f5D4A9020aC26ffEB9037e751	200	0.218

TABLE 5.3: Result of rank and score of graph B

Results (Graph C edges = 70)			
Id	Address	Score (Smart con- tracts)	Score (Nor- mal)
1	0x2f6C7ec63B9189d583Ff8B0140f6fB756650d286	60	0.075
2	0x5CE7Ec397Bbf9E5719Ad4F3B4109d264632C95BC	65	0.081
3	0xa21A0bb3978Af69f5D4A9020aC26ffEB9037e751	72	0.091
4	0xa92A336814043F190e5C6ED7E536759c58C4481b	77	0.098
5	0x3862768B3f7cd1738661Dac2F4D9eEd39E957F31	80	0.101
6	0x9Ff01c647Ac40eD816371aA5e25C81a6b7CC3008	83	0.104
7	0x9F31758e22C6ADd77a11C5EeC4e4C457FAe67c80	87	0.109
8	0xA2c99b38d3e2CD69CD7E58D80d1ca39c270B3D34	87	0.110
9	0x2340F476DBc54Ef195226A3Cad2895133f480193	88	0.111
10	0x74C0f63F1423fba0047B4bffA45f39a033a41e00	91	0.115

TABLE 5.4: Result of rank and score of graph C

Chapter 6

Discussion

The suggested decentralised reputation system provides a method to assign a global trustworthiness score through the Pagerank algorithm to the participants in a specific network. Based on this score, one can infer an entity's centralities and trustworthiness; the node with a higher score can refer to the most critical and trustworthy entities in the blockchain environment. According to the result in section 5.2 and 5.3, the result of the purpose system might depict the performance of the proposed system, which acts in accordance with the number of nodes and transactions accumulated in the network. However, even though the design mechanism might not be robust in enormous network size, it noticeably considered the high-efficiency performance in both ranking accuracy and transaction performance in a small network.

The implementation process and evaluation result might emphasise the research question and follow.

- How to implement a blockchain-based reputation system?

The reputation system implemented through the Ethereum blockchain platform can use various frameworks and implementation techniques. These might essentially be to implement the system through the blockchain environment. In addition, the system can be illustrated by the mechanisms of the immutable trust system, which numerical measurement can apply to the trustworthiness of entities. The implementation process might be consists of four essential phases, iterative studies, algorithm and function design, implementation, and evaluation. The first stage is used to research the potential component which might use in the project. Continuity, the next step, can be utilised the knowledge explored in the previous step. In the implementation step, this process also applied several implementation techniques to produce an efficient contract code. Finally, the result must be evaluated to depict how the proposed system performs, which also use to answer the following research question.

- How well does the mechanism perform for computing ranking scores in smart contracts?

According to the test and evaluation results described in section 5, the verified system performs satisfactorily, especially in diminutive networks. The graph 5.5 can be exemplified the time and the complexities of the suggested mechanism, which might be denoted as $O(n)$ of complexity. Since the $O(n)$ represents the response time that will rely on a number of inputs - in this case, the number of nodes must be calculated as the reputation score each time that entities and links in the system are modified. For this reason, the mechanism is robust in the environment that limits the number of nodes since it can preserve the high rate of ranking calculation compared with the standard library.

6.1 Limitation

Scalabilities: A couple of aspects still have some room for improvement in this dissertation. Essentially, for this algorithm to be used in a real-world scenario, more assessments should be done, especially regarding scalability. For example, even though this research involves a hundred nodes, the response time for each transaction might not be suitable for a real-world problem, especially when there are numerous edges in the network. To emphasise the mentioned issue, the response time is dramatic growth according to the number of transactions that occurred in the system and the number of entities involved in the environment. However, the number of nodes created in the simulated network in each experiment might not reflect the real-world situation. This highlight issue can be noticed that the proposed system might represent the expensive cost of operation and seems not robust to apply in massive real-world networks even though the proposed system can generate high accuracy of ranking computation.

Penalties approach: Another critical characteristic is the penalty system. In this study, the penalty was added to the whole system by reducing the vouch token when the nodes vouch for misbehaving entities. However, this might not be the best approach since the token reduction might not be consistent with the proposed system's goals which aim to preserve the reputation of entities in the system. A better alternative is to reduce or remove the ranking score, which will be discussed further in the Future Work section.

Weighed directed graph: Additionally, since this work's ultimate goal is to develop a practical reputation system for a blockchain network, other aspects should be explored more. For instance, Node ranking is only one factor that plays a role in deciding whether a node is trustworthy. In other words, the centralities of entities might refer to the incoming link from other peers in the system, which might not accomplish the realistic scenario of the reputation mechanism. Other factors may include a weighted ranking

that could skew the rank to favour or disfavour each node according to certain conditions other than its in and outbound links. Another interesting possibility is adding a sentiment analysis system that considers each node's actual reputation amongst users. The following section will dive into a plan to improve these limitations.

Reputation score: The proposed system is mainly developed on smart contracts to allow the node to create a reputation environment. As mentioned on 4.5, the Ethereum smart contract is not fully supported decimal variable in the contract code. The score generated through the designed function might give a score in the form of an integer number. The ranking of trusted entities might be correctly generated, as depicted in the experiment. However, this limitation can produce an unsatisfactory result when the network is scaled up to fulfil real-world usage.

6.2 Future Work

This thesis project's time constraints prevented the exploration of several existing literature on various reputation ranking models which might support the system capabilities and the commission of a comprehensive reputation system that assesses all aspects. As a result, this research seems to miss addressing several factors. The following is a list of proposals for further study or for expanding and contributing to the existing implemented system:

- The scalability problem can be solved using a "Dynamics Pagerank" algorithm. Unlike the traditional Page ranking algorithm, the Dynamics Pagerank algorithm aims to rank each node according to its in and outbound links. However, this evolved algorithm can accomplish the same task much quicker due to less computation complexity. As mentioned in the study of Ryan A. et al.[38], the proposed dynamic Pagerank algorithm indicated the external study effect on the centralities of a node by presenting the Pagerank as a dynamic network which matches the proposed mechanism. In addition, the empirical study of the real-world network is necessary to explore since the randomness might not be enough to reproduce the real behaviour of a massive network. This mentioned scenario tends to have a degree of distribution following the power-law; in other words, a free-scale network [7, 10, 44].
- Another limitation that needs to be sorted out is the penalty system which the solution was hinted at in the previous section. By switching the token penalty for a ranking score penalty, the whole system becomes more coherent, and the score can better reflect the actual reputation of each node.

- As for the last limitation of this dissertation, other aspects of a standard reputation system should be researched to optimise this work further and eventually reach a real-world practical solution phase.
- According on existed evaluation of the design system, the secondary function, such as node and link modification, deletion is excluded from the evaluation scope. This facet also considers the important aspect of reflecting the actual behaviour of graph networks, in which transactions and peers are randomly created, updated, and delete.

Chapter 7

Conclusion

A reputation system for a directed graph network can assist in pre-evaluating a transaction's outcome and help increase the reliabilities of the system's entities. The reputation score can illustrate the importance and trustworthiness of people in the specific network. This master's thesis project studied the approach to implementing the reputation system through blockchain networks which notice as a transparent transaction system. It proposed a reputation model that allowed entities to vouch, reject, and delete each other. It provided a method to aggregate those endorsements score to associate a reputation with the participants via the Pagerank algorithm as the primary approach to calculate trustworthy of each node. The ranking score assigned by the endorsement system represents a given entity's trustworthiness and importance. This system can depict each entity's importance through access through inbound and outbound links. However, even the node with a few numbers of vouch but receives the vouch from high reputation entities, the node which receives vouch is also considered the critical node in the network.

This research also tries to evaluate how the suggested mechanism is well performed on a decentralised network since the proposed model of reputation calculation - the Pagerank algorithm which might require the expensive cost of computation power. The experiment reveals that the suggested system performs efficiently in a small network with a few entities. However, while the number of nodes and transaction increases, the performance drops significantly. Thus, this study also mentions the limitation of scalability and another factor worth exploring to contribute to this work to reflect a more realistic scenario, for instance, the alternative penalties algorithm and weighted graph. In conclusion, research represents the proof-of-concept of reputation system development methodology on the blockchain network and performance evaluation, which depicts the weakness and highlights the contribution area in the future.

Bibliography

- [1] Bitcoin platform. <https://developer.bitcoin.org/reference/intro.html>. Accessed: 2022-09-29.
- [2] Ethereum and evm, howpublished = <https://ethereum.org/en/developers/docs/gas/>, note = Accessed: 2022-09-17, .
- [3] Ethereum documentation. <https://ethereum.org/en/developers/docs/>, . Accessed: 2022-09-29.
- [4] Hyperledger documentation. https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/developing_applications.html. Accessed: 2022-09-29.
- [5] Networkx library / pagerank function. <https://ethereum.org/en/developers/docs/gas/>. Accessed: 2022-09-29.
- [6] R3 documentation. <https://www.r3.com/products/corda/>. Accessed: 2022-09-29.
- [7] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.
- [8] Ahmed S Almasoud, Farookh Khadeer Hussain, and Omar K Hussain. Smart contracts for blockchain-based reputation systems: A systematic literature review. *Journal of Network and Computer Applications*, 170:102814, 2020.
- [9] Alon Altman and Moshe Tennenholtz. Ranking systems: the pagerank axioms. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 1–8, 2005.
- [10] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
- [11] Imran Bashir. *Mastering Blockchain: A deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more*. Packt Publishing Ltd, 2020.

- [12] Z Bahrami Bidoni, R George, and K Shujaee. A generalization of the pagerank algorithm. In *ICDS 2014, The Eighth International Conference on Digital Society*, pages 108–113, 2014.
- [13] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. Pagerank as a function of the damping factor. In *Proceedings of the 14th international conference on World Wide Web*, pages 557–566, 2005.
- [14] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.
- [15] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.
- [16] Elisabetta Carrara and Giles Hogben. Reputation-based systems: a security analysis. *ENISA position paper*, 424, 2007.
- [17] Mohammad Dabbagh, Mohsen Kakavand, Mohammad Tahir, and Angela Amphawan. Performance analysis of blockchain platforms: Empirical evaluation of hyperledger fabric and ethereum. In *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICALET)*, pages 1–6. IEEE, 2020.
- [18] Ernesto Damiani, De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 207–216, 2002.
- [19] Mazin Debe, Khaled Salah, MH Rehman, and Davor Svetinovic. Towards a blockchain-based decentralized reputation system for public fog nodes. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–6. IEEE, 2019.
- [20] Richard Dennis and Gareth Owen. Rep on the block: A next generation reputation system based on the blockchain. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 131–138. IEEE, 2015.
- [21] Nadav Eiron, Kevin S McCurley, and John A Tomlin. Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web*, pages 309–318, 2004.
- [22] Christopher Engström and Sergei Silvestrov. Generalisation of the damping factor in pagerank for weighted networks. In *Modern problems in insurance mathematics*, pages 313–333. Springer, 2014.
- [23] Joshua AT Fairfield. Smart contracts, bitcoin bots, and consumer protection. *Wash. & Lee L. Rev. Online*, 71:35, 2014.

- [24] Hwai-Hui Fu, Dennis KJ Lin, and Hsien-Tang Tsai. Damping factor in google page ranking. *Applied Stochastic Models in Business and Industry*, 22(5-6):431–444, 2006.
- [25] David Gleich, Leonid Zhukov, and Pavel Berkhin. Fast parallel pagerank: A linear system approach. *Yahoo! Research Technical Report YRL-2004-038*, available via <http://research.yahoo.com/publication/YRL-2004-038.pdf>, 13:22, 2004.
- [26] Ferry Hendriks, Kris Bubendorfer, and Ryan Chard. Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing*, 75:184–197, 2015.
- [27] Laurie Hughes, Yogesh K Dwivedi, Santosh K Misra, Nripendra P Rana, Vishnupriya Raghavan, and Viswanadh Akella. Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda. *International Journal of Information Management*, 49:114–129, 2019.
- [28] Audun Josang and Roslan Ismail. The beta reputation system. In *Proceedings of the 15th bled electronic commerce conference*, volume 5, pages 2502–2511. Citeseer, 2002.
- [29] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.
- [30] Mary C Lacity. Addressing key challenges to making enterprise blockchain applications a reality. *MIS Quarterly Executive*, 17(3):201–222, 2018.
- [31] Zhaojun Lu, Qian Wang, Gang Qu, and Zhenglin Liu. Bars: A blockchain-based anonymous reputation system for trust management in vanets. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 98–103. IEEE, 2018.
- [32] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.
- [33] Francesco Alessandro Massucci and Domingo Docampo. Measuring the academic reputation through citation networks via pagerank. *Journal of Informetrics*, 13(1):185–201, 2019.
- [34] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 280–287, 2002.
- [35] Greeshma Nair and Shoney Sebastian. Blockchain technology; centralised ledger to distributed ledger. *International Research Journal of Engineering and Technology*, 4(3):2823–2827, 2017.

- [36] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.
- [37] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59(3):183–187, 2017.
- [38] Ryan A. Rossi and David F. Gleich. Dynamic PageRank using evolving teleportation. 7323:126–137, 2012.
- [39] Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.
- [40] Sujata Tamang. Decentralized reputation model and trust framework blockchain and smart contracts, 2018.
- [41] Martin Valenta and Philipp Sandner. Comparison of ethereum, hyperledger fabric and corda. *Frankfurt School Blockchain Center*, 8:1–8, 2017.
- [42] Kevin Walsh and Emin Gün Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI*, volume 6, pages 1–1, 2006.
- [43] Yao Wang and Julita Vassileva. Trust and reputation model in peer-to-peer networks. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, pages 150–157. IEEE, 2003.
- [44] Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’ networks. *nature*, 393(6684):440–442, 1998.
- [45] Gavin Wood et al. Ethereum: a secure decentralised generalised transaction ledger (2014), 2017.
- [46] Shale Xiong, Andrea Cerone, Azalea Raad, and Philippa Gardner. Data consistency in transactional storage systems: a centralised approach. *arXiv preprint arXiv:1901.10615*, 2019.
- [47] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *2017 IEEE international conference on software architecture (ICSA)*, pages 243–252. IEEE, 2017.
- [48] Atsushi Yamamoto, Daisuke Asahara, Tomoko Itao, Satoshi Tanaka, and Tatsuya Suda. Distributed pagerank: a distributed reputation model for open peer-to-peer network. In *2004 International Symposium on Applications and the Internet Workshops. 2004 Workshops.*, pages 389–394. IEEE, 2004.
- [49] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375, 2018.