UNIVERSITY OF MOHAMED KHIDER BISKRA

MASTER'S THESIS

Service/Consumer Reputation Management System using Blockchains

Author:
Mohamed Cherif KELALA

Supervisor:
Dr. Okba TIBERMACINE

A thesis submitted in fulfillment of the requirements for the degree of Master of Computer Science

in the

Computer Science Department

July 4, 2021

Declaration of Authorship

I, Mohamed Cherif KELALA, declare that this thesis titled, "Service/Consumer Reputation Management System using Blockchains" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Mohamed Cherif KELALA

Date: July 4, 2021

UNIVERSITY OF MOHAMED KHIDER BISKRA

Abstract

Faculty of Exact Sciences, Natural and Life Sciences
Computer Science Department

Master of Computer Science

Service/Consumer Reputation Management System using Blockchains

by Mohamed Cherif KELALA

While blockchain technology is currently being investigated for its potential to provide distributed, decentralized, and time-stamped transactions in a variety of applications, it is also being investigated in the following disciplines: It is allegedly attributed to, as a result of its fault-tolerant methodologies, which ensure the preservation of immutable data records, making it impossible to modify the data. The trust frameworks and reputation models of an online interaction system infer the trustworthiness of interacting entities. When dealing with an entity, the risk of failure or likelihood of success is determined by the information provided by the reputation system. As such, the system must include a trustworthy reputation score.

Acknowledgements

I want to thank my supervisor Dr. Okba TIBERMACINE, for guiding me, giving me information without limits and for giving me the chance to fulfill my goal and get the Master's degree, Thank you!

Also I want to think my father for always believing in me and being my greatest supervisor, my sisters and my wife, Thank you so much!

Contents

List of Figures

List of Tables

Dedicated to my Mother's soul, who's resting in paradise Inshaa Allah, who wanted to see her son graduate an did not get the chance to. May Allah forgive you for your sins and rest you in peace.

Chapter 1

Introduction

Today, we rely on the internet for almost everything, from communicating with one another via email to searching for news or media files to shopping for everyday necessities. It is fair to say that the internet has become an inseparable part of our culture and way of life. According to statistics, the number of internet users worldwide is projected to increase by more than a billion people each year. Given the global reach of the internet and the diversity of its users, it is reasonable to assume that not all online interactions take place between known entities or parties. The online identities that everyone uses to interact with one another provide no means of verifying whether or not the person behind the profile is the same as his/her real-world identity, or the manner in which the person presents themselves. Even in the real world, determining the trustworthiness of an entity is a difficult task to accomplish.

When used in online encounters, trust and reputation can be used to predict the outcome of online transactions. It is possible to define an online interaction system as any platform where users communicate with one another for a variety of purposes, including payment processing, news dissemination and file sharing. A negative outcome can have a significant impact on those who are involved as a result of this. Failure to receive a high-quality music file, to give another example, differs from failure to complete a transaction in which one is purchasing real estate. Failure of a transaction can result in significant financial loss, and in order to avoid this, the integrity of the interaction system and the reputation models in that system are critical. They make every effort to provide as much information as possible about the entities in question in order to forecast the outcomes. A reputation system collects information about the entities that interact with one another on a continuous basis, and it keeps track of how things change.

Information offered by the underlying reputation system influences whether there is a risk of failure or a high possibility of success when dealing with a different entity. More commonly, the identity's trustworthiness is presented as a trust score, which is calculated for each online identity. One cannot conduct an online transaction unless both parties first choose an online identity to connect with prior to starting the transaction. Because the outcome has been observed and stored, the current trust value is updated, resulting in further updates to the reputation system.

Chapter 2

Reputation management Systems

2.1 Introduction

As we all came to a point in time where most of our interactions with the world are made through the internet, reputation of entities has become a very important factor to consider before engaging any transaction or interaction with an unknown party. Therefore, reputation systems play a big part in making these decisions much easier. eBay, an e-commerce platform, StackExchange, a question-and-answer platform, and Reddit 3, a content rating and discussion platform, are all examples of current online interaction platforms that incorporate reputation systems. Users' trust scores are calculated using their feedback, positive and negative ratings, as well as upvotes and downvotes from other participants who share the same ability to interact with the system. The final score derived from these measures has the potential to increase or decrease the user's reputation, as well as restrict their ability to interact with other users.

To safeguard an online interaction system, it is critical to have a trustworthy security framework and reputation model that define the rules of interaction between online identities as well as the extraction of useful information from them. This critical factor ensures that the information remains accurate, unaltered (i.e., no changes have been made), and accessible for future use. The overwhelming majority of web-based interaction systems are client-server in nature. Centralization also exposes the system to external attacks and the introduction of internal modifications and enhancements. If a server fails, the entire network will fail. If one of the servers fails, the network as a whole will fail.

2.2 Reputation Systems

We begin by referring to Gambetta's definition of reliability and trust [2], which is frequently used in the field of trust management research: "trust is the subjective probability with which an agent assesses that another agent will perform a particular action in a context in which it affects his own action".

When attempting to quantify Gambetta's subjective probability, or what is referred to as trustworthiness, reputation is frequently used as a measure.

Reputation systems are systems that have been used to foster trust between participants in a variety of activities over the years, including online activities.

These systems are frequently used on e-commerce websites such as eBay and Amazon.com, as well as on online advice forums such as Stack Exchange. These reputation management systems represent a significant advancement in the field of "decision support for Internet-mediated service provision." With the popularity of

online communities for shopping, advice, and the exchange of other critical information growing, reputation systems have become critical to the online experience. Even if a consumer cannot physically try a product or service or observe the person providing information, they can have confidence in the outcome of the exchange due to the trust established by recommender systems [8].

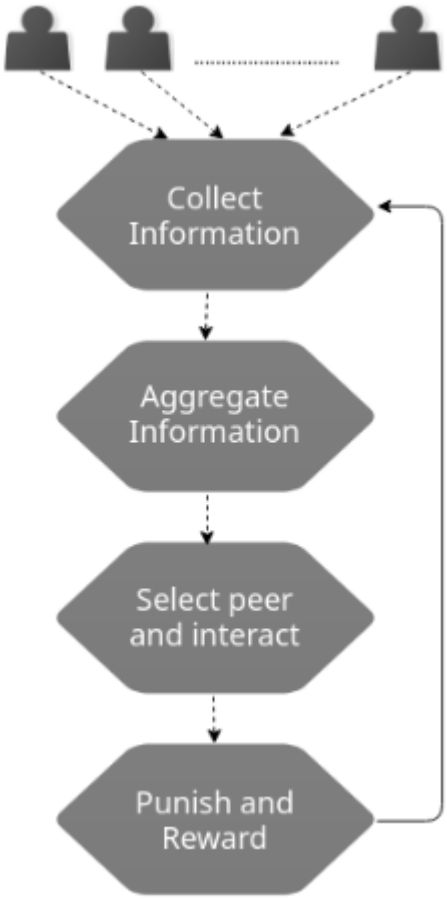

FIGURE 2.1: Trust and Reputation Model steps based on[7]

As stated previously by [7], there are four distinct steps that must be completed before a fully functional "trust and reputation system" can be established which is shown by Figure 2.1.

The reputation system focuses a lot on the first two steps, namely data collection and data computation.

Entities and the values that they represent interact with one another in order to form reputation interactions, and the information contained within the reputation system is concerned with these interactions.

The final two steps of peer selection and punishment or reward are carried out via a transactional system.

Due to the fact that implementing a transaction-based system is not the primary objective of this project, feedback regarding the success or failure of a transaction is entirely based on an assumption about the transaction network.

2.3 Exploiting Reputation Systems

In recent years, the literature [4, 13] has described a diverse range of different attacks on reputation systems, indicating a significant increase in the number of different attacks.

Koutrouli and Tsalgatidou [5] classified attacks into three categories in order to provide a structured overview of vulnerabilities:

- dishonest feedback,
- inconsistent behavior,
- and identity-based attacks.

TABLE 2.1: Example attacks on reputation systems.

Attack class	Name	Description
Unfair feedback	Vote tossing	The attacker provides numerous favorable ratings in order to artificially boost an entity's reputation.
	Defamation	The attacker provides numerous negative ratings in order to unfairly harm an entity's reputation.
Inconsistent behavior	Exploitation of a value imbalance	The attackers establish a favorable reputation by selling inexpensive items while committing fraud on the more expensive ones.
	On-off attack	Initially, the attackers act honestly in order to earn a good reputation, after which they capitalize on that good reputation. After a certain threshold value has been reached, he resumes his honest behavior and begins the process all over again.
Identity-based	Whitewashing	From the beginning, the attacker exhibits malicious behavior. After receiving negative feedback, he creates a new account to try again.
	Sybil-attack	The attacker creates a large number of accounts (Sybils) at the same time in order to increase his influence in a community.

2.4 How does it work?

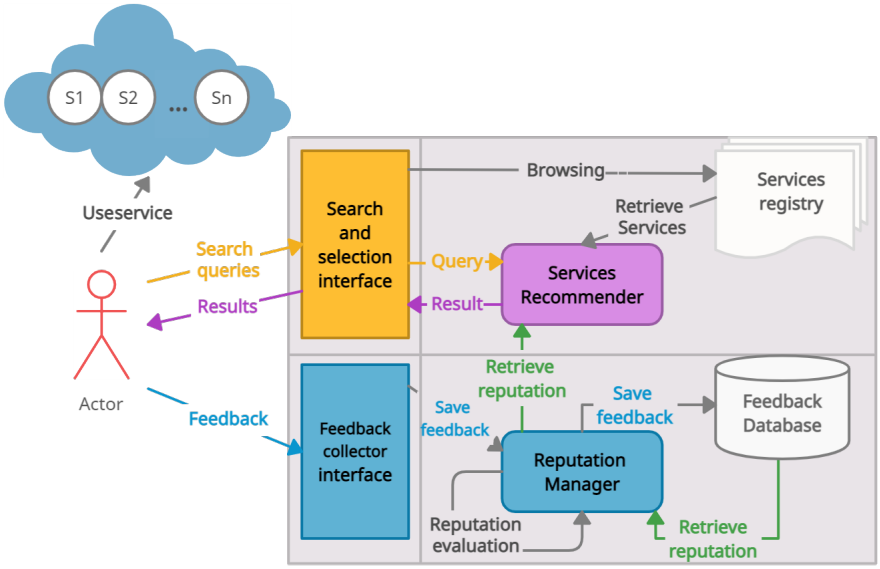

FIGURE 2.2: Reputation System architecture.

Using a simple narration to reputation systems mechanism, participants can promote a rate-able entity (in our case a web service) based on their honesty factor. Participants give their feedback on services - that are either searched for, recommended by the system or simply found - to the system which stores them into a data base for further process. The system assesses periodically these feedbacks to calculate reputation for the effected services, and then updates credibility (honesty) factor for these users, and lastly updates the database.

- The diagram 2.2 illustrates the architecture of the reputation management framework. Users can search for services via the Search and Selection Interface by entering search queries or by browsing directly through the framework's registries. Following the completion of a selected service, a user may submit to the system via the Feedback Collector a feedback rating indicating her/his degree of satisfaction. The feedback database stores the collected feedback ratings.

- The Reputation Manager evaluates the reputation of services on a periodic basis using new feedback that is received and entered into the feedback database. This component is used to determine a service provider's reputation. During the service recommendation process, newly assessed reputation scores are saved in the reputation database for later use.

- To assist its users in finding services that match their search queries, the Search and Selection Interface presents them with sorted sets of services. The Service Selector is responsible for the preparation of these sets, which:

- Receives services from service registers,
- Extracts from the reputation database reputation scores.
- It sorts services according to their reputation ratings.
- The Search and Selection Interface receives the results and displays them to users.

The following metrics form the basis of the model for assessing reputation:

1. **User honesty:** credibility of a user has an effect on the feedback ratings during the reputation assessment process. In continuation of what we learned in "tab," it is critical for an accurate reputation assessment to take the credibility of users into account.
2. **The history of user ratings:** malicious users can act in a variety of ways; they can start out as honest users before gradually changing their behavior. As a result, the assessment model must estimate and update the credibility of users in light of their previous rating history.
3. **Punishment of suspicious users:** users who are suspected of being suspicious are those whose estimated credibility is less than a certain level (i.e. a given threshold). In order to ensure the security of the system, the model neutralizes the feedback ratings of all suspicious users. This mechanism ensures that the feedback ratings used in the evaluation of service reputation are as accurate as possible.
4. **Feedback history:** during each reassessment, the reputation assessment model makes use of all of the feedback ratings that have been collected in the database.

5. **Temporal sensitivity:** As part of its response to the effect of time on ratings, the model includes a temporal sensitivity factor (denoted by λ) that assumes that newer feedback ratings will have a greater impact on the assessed reputation values.

2.5 System struggle

It is critical for an entity's reputation to be as robust and transparent as possible, and the underlying reputation system must meet these requirements. To minimize the risk of fraud, it is necessary to provide assurance that the available information has not been tampered with and that the claimed identity is correct. As the current system's architecture is centralized, the dissemination of reputation information is susceptible to external attacks as well as internal modifications. As a result, it is unable to guarantee the reliability and immutability of data in its current state.

This master's thesis proposes the use of blockchain technology for the storage and governance of reputation data in order to ensure that all parties involved have access to reliable and publicly verifiable information.

2.5.1 Conclusion:

Reputation systems using centralized database can no more be trusted, as we saw earlier. So in this thesis we are trying to provide a solution to the risks associated with the centralized database architecture. For that matter, we propose using Blockchain technology.

Chapter 3

Blockchain

3.1 Introduction:

The term blockchain technology is used by people to mean various things and it is sometimes confusing. It can be The Bitcoin Blockchain they talk about, The Ethereum Blockchain, other digital tokens or virtual currencies. Most of the time, however, it's distributed ledgers, i.e. a list of transactions replicated on a number of computers instead of it being stored on a central server.

In almost all recording events like transactions, Blockchain has demonstrated its valuable application. This utility could be extended to medical records that are often inaccurate and divergent. The healthcare professionals are responsible for ensuring the accuracy, completeness and availability of sensitive health records for authorized individuals only. This can all prove difficult if healthcare providers all have different information storage systems.

3.2 What is blockchain?

It is the genius design of Satoshi Nakamoto, originally invented for Bitcoin currency. But it has since developed into something bigger.

Blockchain is a time-stamped decentralized information storage and transmission system that is transparent, secure, and operates without the need for a central authority.

A blockchain, by extension, as we said a time-stamped database that stores the history of all transactions between its users since its beginning. This database is both secure and distributed: it is shared by its various users without the use of intermediaries, allowing each of them to verify the channel's validity.

There are public blockchain, which are open to everyone, and private blockchain, which are only accessible to a select group of people. As a result, a public blockchain can be compared to a public, anonymous, and unfalsifiable accounting book. [16]

The reason for this great admiration of the blockchain is that:

- It is decentralized because it is not owned by a single entity.
- The information is stored inside encrypted.
- The blockchain is unchangeable, such that the data within the blockchain can be modified by no one.
- The blockchain is open, so that if you want you can trace the data.

3.2.1 Blockchain structure

The blockchain is a series of blocks of data (desired to be stored), in which cryptographic algorithms (i.e. chains) are used to secure and link each of these blocks of data (i.e. block) see figure 3.1

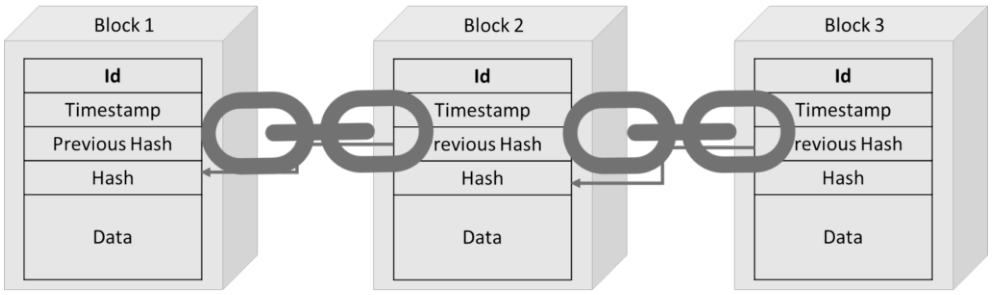

FIGURE 3.1: Pictorial representation of a blockchain.

3.3 How does it work?

Information held on a blockchain is a shared database that is duplicated over a computer network hundreds of times, and this network is intended to update this sheet regularly. The blockchain database is not stored in one location, so it is truly public and easy to check the records that it keeps. A hacker does not have a centralized version of this information. Hosted simultaneously by millions of computers, its data is available on the Internet to everyone.

The blockchain network exists in a consensus state which is checked periodically by itself, every node of that network has a full copy of the ledger and joins the network voluntarily, and has an incentive to process and verify transactions to win bitcoins or any other cryptocurrency the network uses. The term "block" refers to a group of transactions. The transactions in a block can't be tampered with because changing any information on the blockchain would require your computer to take control of the entire network; this is theoretically possible, but unlikely in practice.

- **Blockchain algorithm:** The operation and purpose of a blockchain depends on its algorithm. For Bitcoin, the blockchain algorithm is designed to accommodate a public, decentralized transaction system based on encryption evidence rather than trust (allowing parties to directly transact without the need for a trusted intermediary). Blockchain algorithms are not, however, limited only to the public 'permit-less' format of Bitcoin and can be adapted to accommodate private, centralized books (or hybrid systems) where transactions are no longer anonymous and only approved administrative bodies are able to verify transactions.

- **Miners (or nodes):** A public, decentralized blockchain is managed on a peer-to-peer basis by all of the network's participants (or "miners"). A person becomes a miner by downloading a "node" that connects them to the blockchain network (i.e. a software program that the miner runs on their computer).

These nodes download the most recent version of the blockchain and compare each new block and transaction to the rules specified by the blockchain 's algorithm in real time. Miners can also participate in blockchain transactions using nodes.

- **Adding a block to the chain:** A miner can participate in blockchain transactions via their node software. Each transaction is digitally signed and encrypted with a public and private encryption key that the miner receives upon downloading their node. The consensus of all active miners determines the order in which transactions are recorded on the blockchain (rather than one or more trusted intermediaries). Each blockchain technology has its own version of this process. To complete a transaction in Bitcoin and other public, decentralized Blockchains, miners must solve a system-generated proof of work puzzle (which may include a 'hash' or one-way function).
- **Verification of blocks:** When a miner considers a transaction complete, the miner will transmit the transaction data block along with the hash (containing also a time stamp) to all other network miners that collectively check the block content. Upon verification, miners update their respective nodes to include the new transaction block.

The simplicity of blockchain technology, as well as its security provided by multiple node verification, are two of its most attractive characteristics. In the event of a successful attempt to defraud the system (for example, by artificially including a block with false transaction data), a 'fork' in the blockchain will occur, resulting in two chains (and two accounts of events) running side by side. As a result, the blockchain protocol requires that miners work off the chain with the greatest number of blocks in order to avoid a timeout. Because blockchain transactions are verified through a process of consensus across all nodes, they are more resistant to unauthorized change and malicious tampering than centralized, legacy systems are.

Traditional commerce may become obsolete as a result of this new method of sending and verifying transactions. The blockchain has the potential to make stock market trades instantaneous, to keep a fully public record of Land Registry transactions, and even to host elections. The possibilities are endless with blockchain technology.

3.3.1 The Three Fundamental Elements of Blockchain Technology

The three primary characteristics of Blockchain Technology that have contributed to its broad acceptance are as follows:

- Decentralization
- Transparency
- Immutability

Decentralization

Before the invention of Bitcoin and BitTorrent, we were more accustomed to centralized services. The concept is straightforward: you have a centralized entity that stores all of the data, and you would only be able to interface with this entity in order to obtain whatever information you required. However, the concept is complicated.

We have taken full advantage of centralized systems for many years; unfortunately, they are susceptible to a number of security risks.

- To begin, because they are centralized, all data is held in a single location. In turn, this makes them attractive targets for potential hackers.
- Suppose the centralized system were to go through a software upgrade, the entire system would come to an abrupt halt.
- What happens if the centralized entity is unable to function for whatever reason? Nobody will be able to gain access to the information that it possesses in this manner.
- What if, in the worst-case scenario, this entity becomes corrupted and malicious? If this occurs, then all of the data contained within the blockchain will be compromised as well.

Information is not stored by a single entity, but by a collection of entities in a decentralized system. In reality, everyone on the network owns the information.

A decentralized network enables you to communicate directly with your friend, which is especially advantageous in social situations. That was the primary motivation for Bitcoin's creation. You, and only you, are responsible for your financial well-being. Your money does not need to pass through a bank to reach its intended recipient.

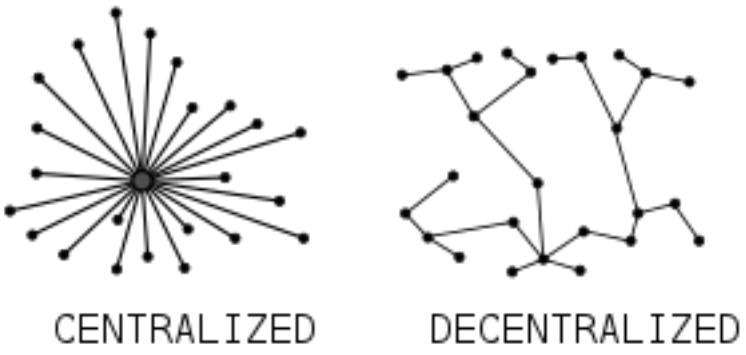

FIGURE 3.2: Centralized vs Decentralized network architecture

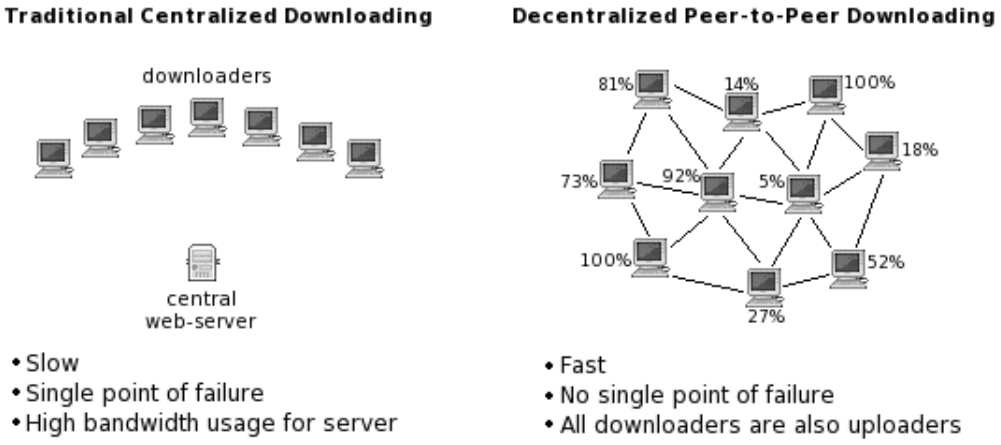

FIGURE 3.3: Centralized vs Decentralized network pressure

Transparency

"Transparency" is an enticing yet misunderstood concept in the world of blockchain. According to some, blockchain technology provides privacy, while others assert that it is completely transparent. What, in your opinion, is the root cause of this?

A person's identity is concealed using sophisticated cryptography, and they are only represented by their public address. Thus, if you were to look up "Bob sent 1 BTC," you would see "1MF1bhsFLkBzzz9vpFYEmvwT2TbyCt7NZJ sent 1 BTC," rather than "Bob sent 1 BTC."

TxHash	Block	Age	From		To
0x2d055e4585ae2a...	5629306	16 secs ago	0x003e3655090890...		0x2bdc9191de5c1b...
0xb4d37c791ff4cde...	5629306	16 secs ago	0x6c3b4faf413e0e4...		0xf14cb3acac7b230...
0x9979410dcb5f4c...	5629306	16 secs ago	0x99bcd75abbac05...		0x2d42ee86390c59...
0x189c4d4aae09be...	5629306	16 secs ago	0x175cd602b2a1e7...		0xd39681bb0586fb...
0xda0e9bbb11fb77...	5629306	16 secs ago	0x73a065367d111c...		0x01995786f14357...
0x6be498fafad9acb...	5629306	16 secs ago	0xa3eb206871124a...		0x8a91cac422e55e...

FIGURE 3.4: Snapshot of some Ethereum **??** transactions

Even though the individual's true identity is protected, you will still be able to view all transactions made using their public address. This level of transparency has never been achieved in the financial sector's history. It adds a much-needed layer of accountability to some of the world's most powerful institutions.

And to be clear this feature is a great privacy factor, but some blockchain systems are not obliged to implement it, as their system does not require to.

Immutability

Immutability refers to the fact that once something is added to the blockchain, it cannot be changed.

Consider the long-term value of this to financial institutions.

If people are aware that they are not permitted to "work the books" or tamper with company accounts, imagine the number of embezzlement cases that could be averted.

This property of the blockchain is due to the use of a cryptographic hash function.

Hashing, in its simplest form, is the process of taking an input string of any length and generating an output string of a fixed length. In the context of cryptocurrencies such as bitcoin, transactions are taken as input and passed through a hashing algorithm (Bitcoin uses SHA-256), which results in an output of a fixed length, no matter how big or small your input is, the output will always have a fixed 256-bits length.

As we said, blockchain consists of a linked list of data as well as a hash pointer to the previous block, which together form a chain of transactional records.

What is the operation of a hash pointer? While it performs the same function as the pointer, instead of just containing the block's address, it also contains the hash of the data that was contained within its predecessor's block.

INPUT	HASH
Hi	3639EFCD08ABB273B1619E82E78C29A7DF02C1051B1820E99FC395DCAA3326B8
Welcome to blockgeeks. Glad to have you here.	53A53FC9E2A03F9B6E66D84BA701574CD9CF5F01FB498C41731881BCDC68A7C8

FIGURE 3.5: SHA-256 hash size comparison using different length inputs

This one small distinction is what distinguishes blockchains as highly dependable and trailblazing technologies in the first place.

3.3.2 Blockchains applications

When it comes to recording all types of data, blockchains have proven to be an effective method, providing a range of benefits including transparency and trust, as well as security and cost savings, depending on how the design parameters are configured. Table 3.1 summarizes the industries in which the application of blockchain technology has the potential to be beneficial. At this point, it's worth noting that these opportunities face a number of obstacles, but significant research is being conducted in each category to assist in resolving those obstacles.

A hype cycle, as is the case with many emerging technologies, is characterized by an enormous surge in interest in the technology, which is triggered by large swings in expectations. Blockchain is no exception to this rule. It is this pattern that is elaborated upon by the "Gartner Hype Cycle," which was developed by Gartner, a global research and advisory firm headquartered in the United States (Gartner, 2017). This chart depicts the maturity, adoption, and potential relevance of technologies and applications for resolving real-world business problems and capitalizing on new opportunities in the context of the information age.

3.4 Smart Contracts

One of the most advantageous features of the blockchain is that, as a decentralized system that exists between all permitted parties, there is no need to pay intermediaries (Middlemen), which saves time and keeps you out of conflict. While blockchains do have some drawbacks, they are unquestionably faster, cheaper, and more secure than traditional systems, which is why banks and governments are increasingly using them for transactions [12].

In 1994, Nick Szabo, a legal scholar and cryptographer, recognized that a decentralized ledger could be used to implement smart contracts, also known as self-executing contracts, blockchain contracts, or digital contracts, and he created a protocol for their implementation [14]. Contracts could be converted to this format and then stored and replicated on the blockchain system, where they would be supervised by the network of computers that runs the blockchain. Additionally, this

TABLE 3.1: Possible opportunities of blockchain technology per category.

Sector	Possible opportunity of blockchain technology	Example
Legal	Reduce the need for human intervention by implementing "smart contracts" that keep track of the contract's parties, its terms, the transfer of ownership, and the delivery of goods or services.	Propy
Supply chain	Increase the level of transparency around shipment tracking, deliveries, and progress with other suppliers in cases where the relationship lacks inherent trust.	Maersk
Government	Serve as an intelligent database for the storage of information such as personal identity information, criminal histories, and e-citizenship, all of which are verified using bio-metrics.	ID2020
Energy	Facilitating a decentralized power grid system that enables the secure use of micro-transactions for both the receiving and sending ends of energy distribution is a priority.	PowerLedger
Food	Enhancing the efficiency of supply chain data processing in order to improve product origin, batching, processing, expiration, storage temperatures, and shipping methods traceability.	Walmart
Retail	Provide a secure market environment for peer-to-peer retail transactions, with product, shipment, and bill of lading data stored on the blockchain and payments made in cryptocurrencies.	OpenBazaar
Health care	Utilize biometrics and/or multi-signature technology to secure electronic medical records, with biometrics and/or multi-signature technology controlling access and update permissions. This simplifies the process of storing, sharing, and updating patient data in a safe, secure environment.	HealthWizz
Insurance	The use of immutable insurance ledgers eliminates the need for auditing and authenticating data, resulting in a reduction in premium costs.	Fizzy
Education	Establish a credible database of digital credentials that includes academic certificates, degrees, transcripts, and assessments, as well as strict rules for granting and revoking.	Blockcerts

would result in ledger feedback, such as money transfers and the receipt of a product or service.

Without the assistance of a middleman or third party, smart contracts enable you to exchange money, property, shares, or anything else of value in a transparent, conflict-free manner.

When comparing smart contracts to vending machines, the best analogy is that the technology is similar. Normally, you would visit a lawyer or notary, pay them, and then await delivery of the document. Simply depositing a bitcoin into the vending machine (i.e. ledger) will result in the delivery of your escrow, driver's license, or whatever else you require to begin transferring funds into your bank account. On the other hand, smart contracts not only define the rules and penalties that apply to an agreement in the same way that a traditional contract does, but they also enforce those rules and penalties automatically. If you're looking for a more comprehensive walkthrough of smart contracts, we recommend checking out our blockchain courses on smart contracts.

Additionally, there are platforms that offer more complex contractual functionality and flexibility, such as Ethereum, which fully implements the smart contract language's features and functionality. Neo and Hyperledger Fabric [3], two relatively new blockchain platforms, enable the creation of smart contracts in a variety of high-level programming languages. The following diagram illustrates the evolution of smart contracts 3.6.

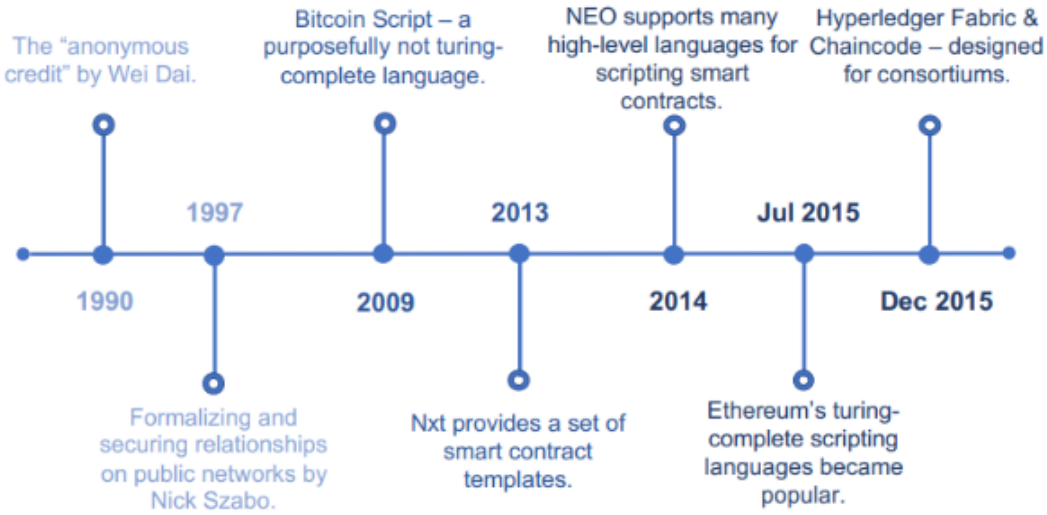

FIGURE 3.6: Evolution of Smart Contracts.

3.4.1 How does a Smart Contract work?

Smart contract development is linked to the adoption of blockchain technology. Which is nothing more than a blockchain-encrypted digital node. Smart Contracts include code-based specifications and permissions, as well as time constraints that can be used to define contract completion dates and deadlines. The smart contract system is illustrated in Figure 3.7.

Bitcoin was the first cryptocurrency to employ a smart node for value transfer between users. The contract's effectiveness requires both the use of electronic signature techniques and the implementation of significant specific conditions, such

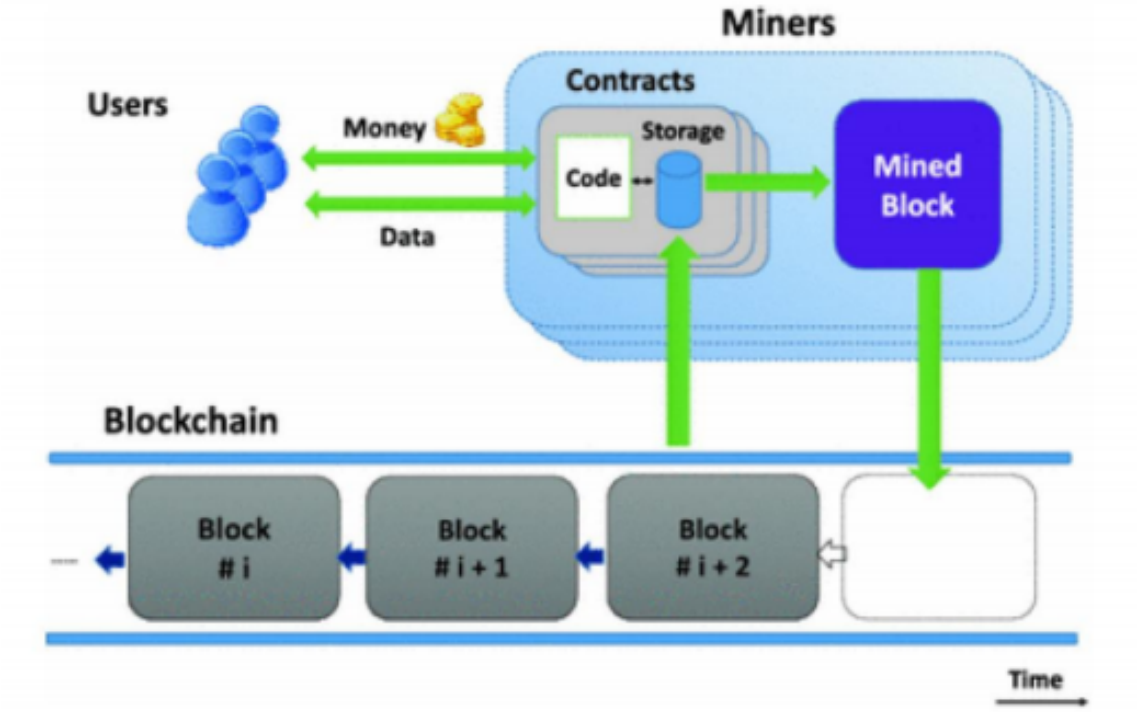

FIGURE 3.7: Smart Contract System.

as ensuring that the amount being transferred is actually available in the sender's account. Without these, automation of the process will be impossible. [9]

Steps to produce a smart contract [11]

- To begin, the contractual parties should agree on the agreement's terms. After all contractual terms have been agreed upon, they are converted to computer code. In other words, the code is a collection of conditional statements that describe the various possible outcomes of a future transaction.
- Once created, the code is stored in the blockchain network and replicated among all blockchain participants.
- It is stored in the blockchain network and replicated among all blockchain participants as soon as a blockchain participant creates the code.

3.4.2 Smart Contract Features:

Smart contracts based on blockchain technology have risen to prominence as a rapidly growing area of blockchain technology. Particularly when these contracts already offer a number of advantages over more traditional arrangements. We'll go over a few of the advantages [10]:

- Autonomy : Their technical architecture enables a diverse range of applications, from automatic self-help to the enforcement of non-legally enforceable agreements. Additionally, because the network manages execution mechanically, the risk of manipulation by a third party is significantly reduced.
- Trust : they generate complete confidence in their implementation by utilizing encryption on the shared ledger. Transparency, autonomy, and security of the contract preclude any possibility of manipulation, alteration, or error.

- Security : As with cryptocurrencies, they employ the highest level of data encryption currently available, ensuring the maximum level of security possible. They provide one of the best and most secure levels of protection available on the network.
- Savings : Because they eliminate the need for a large number of middlemen, smart contracts have the potential to save money. No advocate, witnesses, banks, or other third-party intermediaries are required. This is one of the primary benefits of using a Smart Contract.
- Speed : They are software programs that automate online tasks. As a result, they can complete transactions quickly. When compared to more traditional contracting methods, this speed can result in significant time savings.
- Backup : Smart contracts are used to record all of the details of each transaction, and they are retained indefinitely for future reference. As a result, if someone loses access to their savings account, regaining access is relatively simple.

3.4.3 Conclusion:

As demonstrated, smart contracts have the potential to significantly improve the world while requiring no commission. It has the potential to significantly reduce fraud, delays, and the overall cost of a wide range of activities. However, as technology advances, certain occupations will become obsolete.

Chapter 4

System Design

4.1 Introduction

In this chapter we will approach a general description then we highlight the conceptual side of our application, which constitutes a fundamental step which precedes the implementation, it aims to find technical solutions to implement and build the system analyzed during the previous phase and allows to detail them Different diagrams and scenarios to be implemented in the next phase. This will help to better understand our application. We have used UML language (Unified Modeling Language). In this walkthrough, the UML diagrams used are as follows: Use case, sequence and class diagrams.

4.2 Reputation System Architecture

As we have seen in Figure 2.2, this architecture the reputation system offers a collection of services with their reputation score. Actors can either look for a service, get a recommendation or rate a service. In the case of an actor submitting a feedback (rating a service), the system stores the feedback in a simple database (any normal way of storing data) then evaluates it and updates the score (reputation) of this service, which means storing the new scores into the database.

4.2.1 Search and Selection Interface

Using the Search and Selection interface, users can interact with the system in order to view services, test them and then give his feedback. The user can browse directly through service registries or do a search query. These searches are typically comprised of a number of keywords that describe the services that are being sought. The interface sends queries to the service recommender component, which analyzes them and searches for services that match the queries sent through the interface. The recommender returns sorted lists of services to the interface to display them to the user.

4.2.2 Service Recommender

In order to process user requests, the service recommender component searches the services registry for any service that contains either a keyword that is in the query or by comparing with the service's name.

Then, the collection of services that meet up with the user's search query; is arranged according to the value of each service's reputation.

Finally, sent to the search and selection interface to be presented to the user; the highest reputation service first.

4.2.3 Feedback Collector Interface

A user interface through which the actor can submit his feedback. Feedbacks are expressed as integer values from 1 to 10 grading ratings, with 1 representing complete dissatisfaction, and 10 being fully satisfied. Each service is identified by a unique identification as well as the actors. The obtained feedback is then sent to the Reputation Manager to process it.

4.2.4 Reputation Manager

The Reputation Manager component evaluates the reputation of web services by aggregating feedbacks that have been collected:

- Recover feedback ratings from the last feedback database evaluation round,
- Recalculate the credibility of the users,
- Get all feedbacks for each effected service, and evaluate its reputation,
- Store evaluation results into the reputation database.
- Every time slot T is reached, the reputation manager initiates a new assessment round. The time slot T can be determined by the administrator based on the system's performance.

As part of every feedback transaction, the component stores the following information in the database:

- **Feedback ID:** indicates the current feedback record's identifier.
- **User ID:** indicates the user's identifier.
- **Service ID:** indicates the service's identifier.
- **Rating:** indicates the score given by the user to the service. Rates are unsigned integers between 0 and 10.
- **Timestamp:** indicates timestamp of the feedback.

Due to performance concerns, the feedback collector stores only one record in the database for the same "user to service" feedback. This record is updated when a this user gives another feedback on that same service.

4.2.5 Formula

- Let σ(i,k) be the feedback rating given by user i for service k.
- Rating score range is between 1 and 10, let there be two classes of feedback: negative feedback $\sigma- \in \{1,2,3,4,5\}$ and positive feedback $\sigma+ \in \{6,7,8,9,10\}$
- Let R(S_k) be the reputation of service S_k, it is calculated by the following formula

$$R(S_k) = \frac{\sum_{i=1}^{n} \sigma(i,k) \times h(i)}{\sum_{i=1}^{n} h(i)}$$

Such that : h(i) is the Honesty degree

4.2.6 Honesty Factor

The honesty factor (or credibility value) of a particular user indicates the likelihood that this user will provide honest feedback in the majority of raters' opinion. By default, a newly created user is assigned the value 1/2. This value indicates that the user is neither truthful nor untruthful. Thus, new user feedback ratings have a negligible effect on the reputation of the rated services. However, by providing additional ratings in the future, this user's honesty factor will increase, which is calculated as follows:

$$h_i = \frac{\sum_{s=1}^{t}\left(\frac{nf(s)}{nf^{+}+nf^{-}}\right)}{t} \tag{4.1}$$

Such that :

- nf(s) = nf^{+}, if the feedback is positive.
- nf(s) = nf^{-}, if the feedback is negative.

- t: is the number of services rated by user (i)
- nf+(s): number of positive ratings for service (s)
- nf-(s): number of negative ratings for service (s)

4.3 Adapting with Blockchain

In this work we are proposing a solution of replacing the traditional way of storing data with blockchain technology, as shown in Figure 4.1

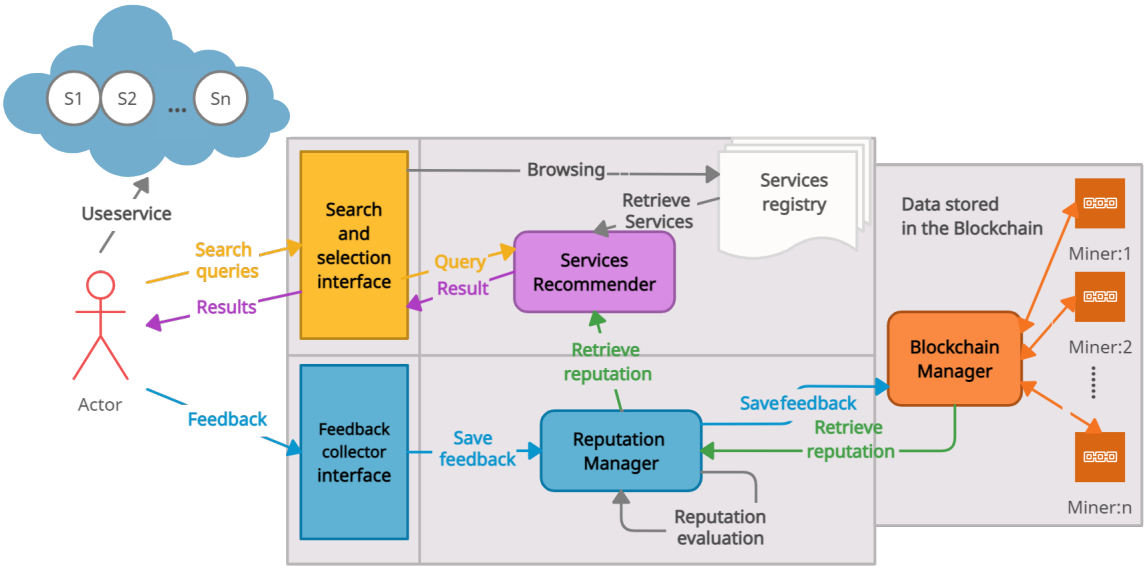

FIGURE 4.1: Reputation Management System Architecture using Blockchain

In this proposed architecture, the components:
- Feedback Collector Interface
- Reputation Manager
- Search and Selection Interface

- Service Recommender
have the same mechanism as the old system. We are proposing a new component in the system, a replacement to the old centralized database manager component, "**Blockchain Manager**".

4.3.1 Blockchain Manager

This component is the responsible for storing all the sensitive data of the system using blockchain technology 3.
- We assume that the sensitive store-able data to this particular system are only the feedbacks given by the users.
- Using blockchain to store other data is possible, and that depends on the system performance.
- Blockchain Manager stores the data obtained by the reputation manager in the same old data structure to the miners.

4.4 Detailed Conception of our system using UML

4.4.1 Unified Modeling Language:

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

The creation of UML was originally motivated by the desire to standardize the disparate notational systems and approaches to software design. It was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in 1994–1995, with further development led by them through 1996.

In 1997 UML was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since. In 2005 UML was also published by the International Organization for Standardization (ISO) as an approved ISO standard.Since then the standard has been periodically revised to cover the latest revision of UML.

4.4.2 Use case diagram:

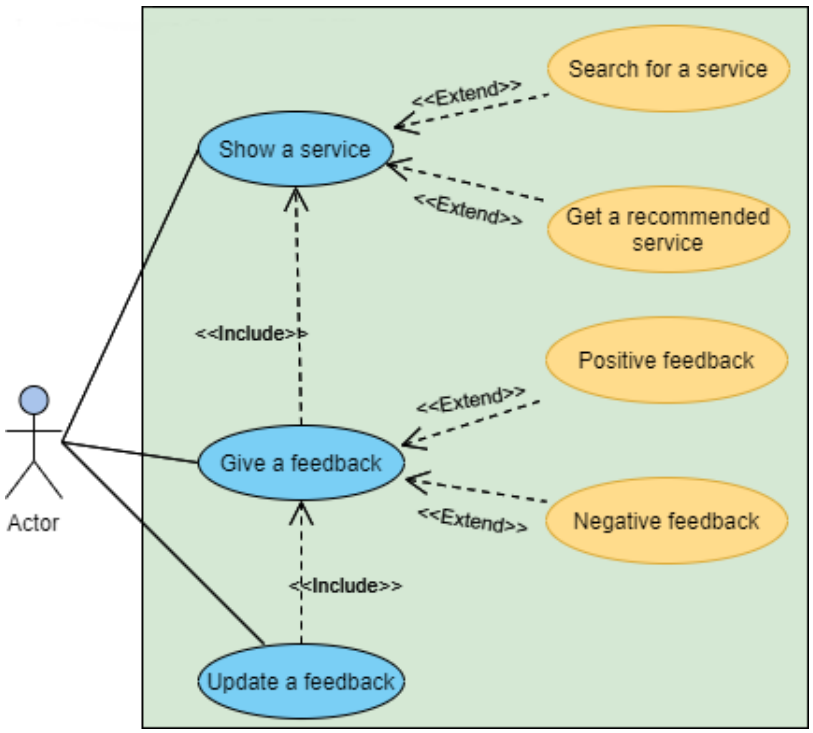

FIGURE 4.2: Reputation System using Blockchain: Use Case diagram.

4.4.3 Sequence Diagrams:

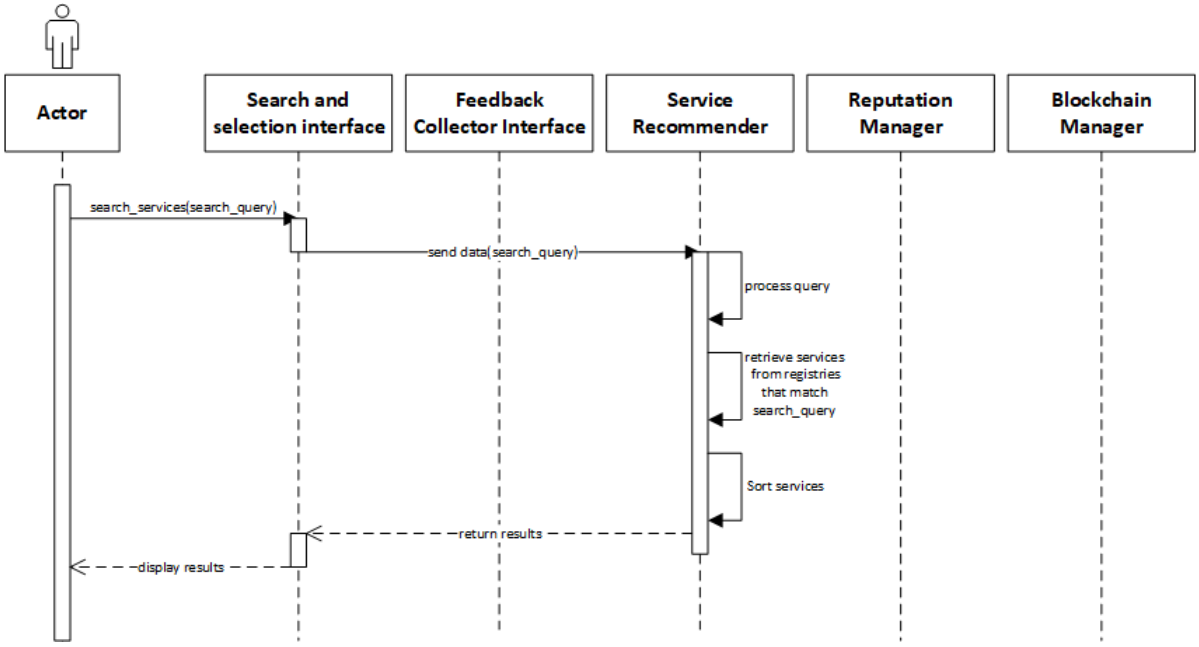

FIGURE 4.3: User search sequence diagram

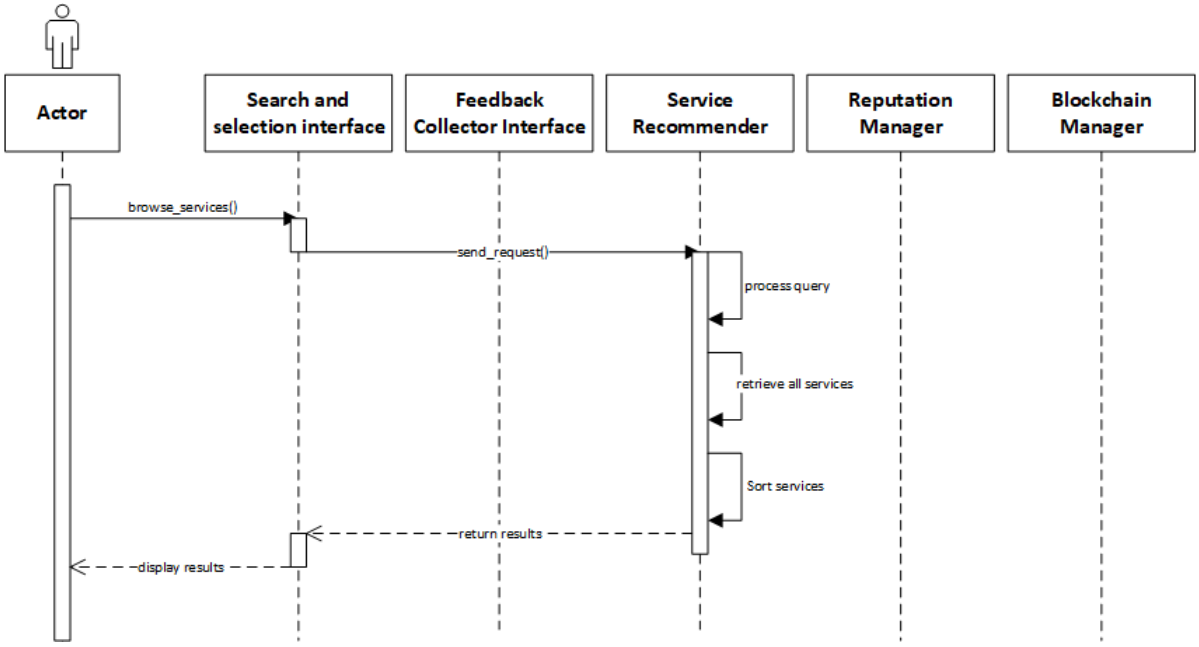

FIGURE 4.4: User search sequence diagram

4.5 Conclusion:

In this chapter, we presented our solution for a "Reputation Management System" using Blockchain, and we have detailed this architecture with the UML modeling

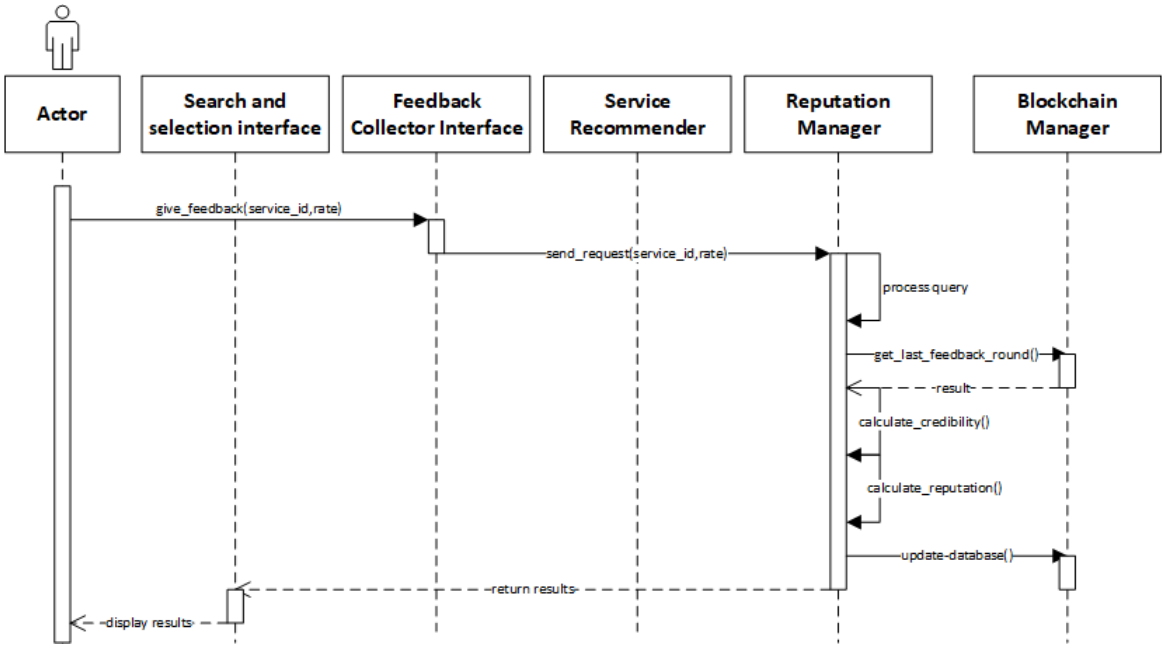

FIGURE 4.5: User give feedback sequence diagram

language through use case, sequence and class diagrams for a purpose of determining the way to solve the problem to study by analysis and thus to propose dedicated solutions of implementation and realization.

The next chapter is the final phase it is the realization of the project we highlight, the approaches and the tools as well as the languages used for the development of the "Reputation Management System using Blockchain" application.

Chapter 5

System Implementation

5.1 Introduction

In this chapter, we try to implement the solution presented previously 4. we will mention tools used that involves the choice of technologies that simplify the implementation of system. This choice comes after the conceptual study and on the hardware and software environment that we have at our hands.

5.2 The hardware and software environment:

5.2.1 Hardware

A PC with the following characteristics :
- Processor : Intel Core i7 8thgen 1.99 GHz.
- Installed RAM 8.00 GB.

5.2.2 Software

We saw best if we implemented a web platform to simulate this system, and get a more realistic experience.

We use:

- OS : Windows 10 - 64 bits.
- Programming Languages: we used Laravel PHP Framework [6]:
  - Front-End:
    - JavaScript 5%
    - CSS 10%
  - Back-End: PHP version 7.2.34 :85%
- Web and SQL server: Xampp version 3.2.4 [17].

5.3 Search and Selection Interface

It is the first screen the user faces after login. Right away it gives you recommendations on highest rated services available, and the possibility to search for a service by keyword or name.

Also the possibility to see blockchain history for complete transparency.

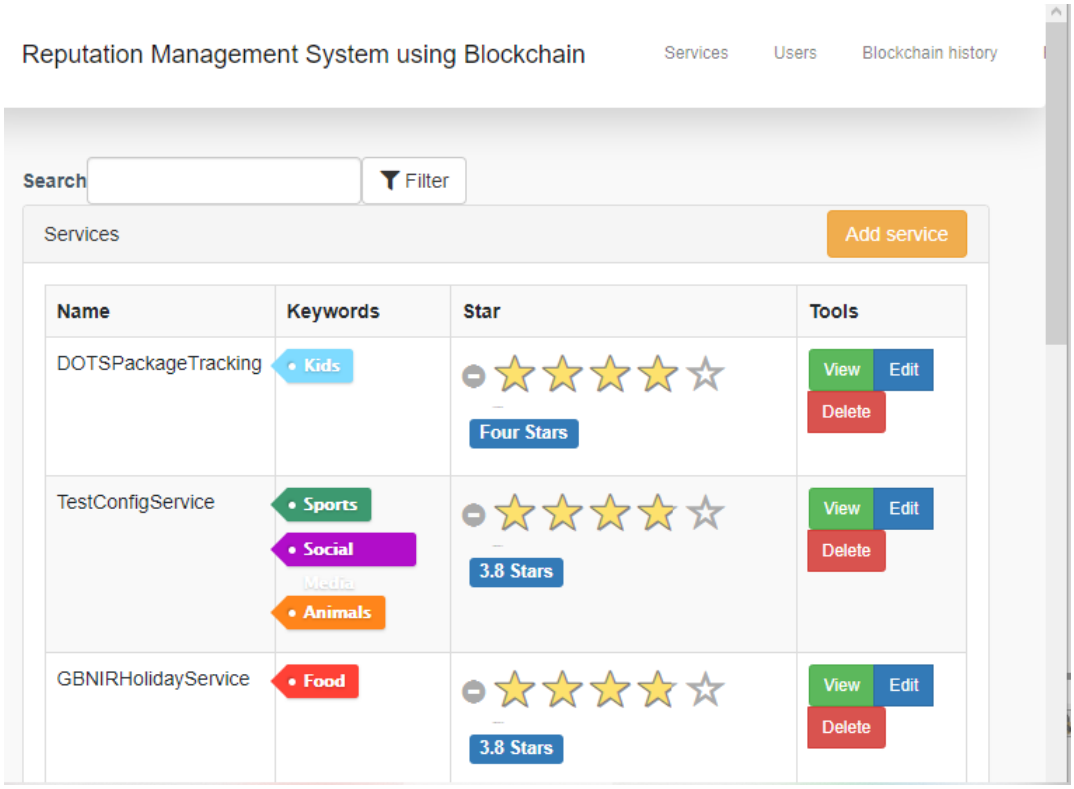

FIGURE 5.1: Search and Selection Interface screenshot
url: /home

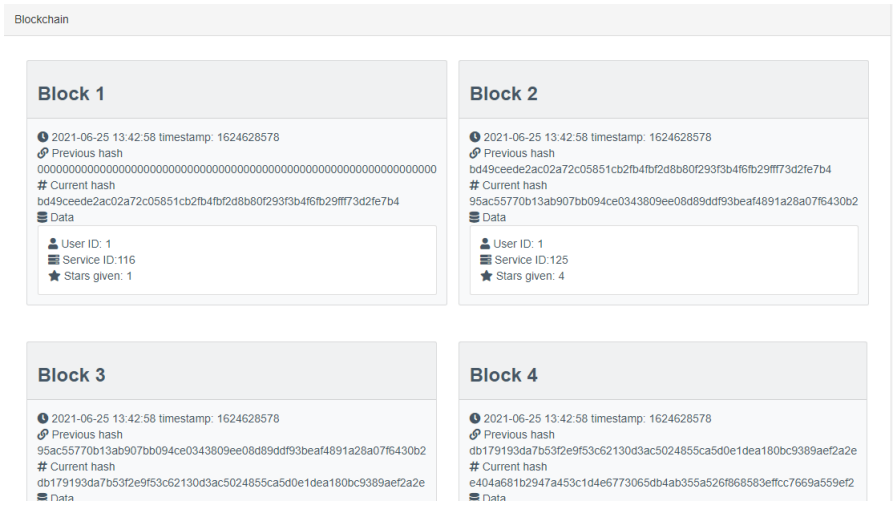

FIGURE 5.2: Blockchain history page screenshot
url: /blockchain

5.4 Feedback collector interface

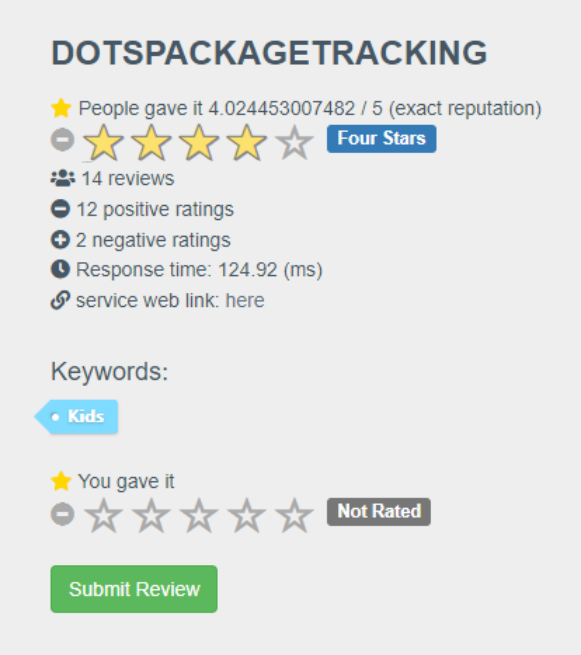

FIGURE 5.3: Feedback collector interface screenshot
url: click on view service button

In this form the user can give his feedback using those stars, as 1 star equals to 2 rating points, and then submits his feedback.

5.5 Blockchain Manager

This is the class in which happens all the blockchain data retrieval or storage operations.

5.5.1 Blockchain platform simulation

We are simulating Blockchain platform using a PHP package created by Flavienbwk [1], which stores the data as an encrypted file then gives the possibility to publish it for the miners. For demonstration purposes, we are working with a single machine to simulate all possible use cases of the system.

```php
class BlockchainManager
{

    private const blockchainFile = 'reputation.blockchain';
    public static function myAddBlock($data): Block{...}
    public function addRating($input): Block{...}
    public static function getHistory(){...}
    public function getNewRatings(): array{...}
    public static function getUserRatingHistory($user_id){...}
    public function getUserRatings($user_id): array{...}
    public static function getLastUserRating($user_id, $service_id): array{...}
    public static function getServiceRatingHistory($service_id, $user_id = null){...}
    public function getServiceRatings($service_id){...}
    public function getServicePosRatings($service_id){...}
    public function getServiceNegRatings($service_id){...}
    private function getNewestRatings(array $options = []): array{...}
    public static function regenerate(): void{...}
    private static function randomGen($min, $max, $quantity) {...}


}
```

FIGURE 5.4: Blockchain Manager class screenshot
Code in file: app/BlockchainManager.php

```php
class Blockchain {

    private $_magic = 0xD5E8A97F;
    private $_hashalg = "sha256";
    private $_hashlen = 32;
    private $_blksize = null;

    public function __construct() {
        $this->_blksize = (13 + $this->_hashlen);
    }

    /** Add a block to your blockchain. ...*/
    public function addBlock($file_name, $data) {...}

    private function write_block(Block &$Block, &$fp, $data, $prevhash) {...}

    private function update_index(Block &$Block, &$fp, $pos, $datalen, $count) {...}

    /** Walks through your blockchain and returns all the data in a JSON format. ...*/
    public function getBlockchain($file_name, array $options = []) {...}

    /** Returns a Block object. Block->hasError() === false if found. ...*/
    public function getBlockByHash($file_name, $hash_search) {...}

    /** Returns a Block object. Block->hasError() === false if found. ...*/
    public function getBlockByPrevHash($file_name, $hash_search) {...}
```

\flavienbwk\BlockchainPHP › Blockchain › construct()

FIGURE 5.5: Flavienbwk Blockchain class
Code in file: /vendor/flavienbwk/blockchain-php/src
Blockchain.php

```php
class Block {

    private $_error = true; // Has correctly initialized ?
    private $_message = false; // Message (generally, an error message).
    private $_offset = null;
    private $_offset_end = null;
    private $_position = null;
    private $_magic = null;
    private $_version = null;
    private $_timestamp = null;
    private $_prevhash = null;
    private $_hash = null;
    private $_datalen = null;
    private $_data = null;
```

FIGURE 5.6: Flavienbwk Block class
Code in file: /vendor/flavienbwk/blockchain-php/src Block.php

5.6 Reputation Manager

```php
class ReputationManager
{
    public const maxStars = 10;

    public const getPosRatings = 1;
    public const getNegRatings = 2;

    public static function areInSameFeedbackClass($score, $wantedFeedback): bool{...}

    public static function isPositiveFeedback($score): bool{...}

    public static function calculateAllCredibility(): void{...}
    public static function calculateAllReputation(): void{...}
    public static function calculateReputation($service){...}

}
```

FIGURE 5.7: Reputation Manager class screenshot
Code in file: app/ReputationManager.php

This is the class in which happens all the calculations and assessments either for users or services.

5.6.1 Calculation of user credibility

As seen with the formula of honesty factor 4.2.6, this code is the responsible for calculating a credibility or honesty factor for a given user.

```php
public function credibility(): float
{
    //$scores  = (new BlockchainManager())->getUserRatings($this->id);
    $scores  =  Rating::where('user_id', '=', $this->id)->get();
    $cred = 0.0;
    if ( count($scores) === 0 ) {return $this->credibility;}
    foreach ($scores as $score)
    {
        $service = Service::find($score['service_id']);
        if (ReputationManager::isPositiveFeedback($score['score']))
        {
            $cred += $service->posRatingsCount() / $service->countTimesRated();
        }
        else
        {
            $cred += $service->negRatingsCount() / $service->countTimesRated();
        }
    }
    $this->credibility =  $cred / count($scores)  ;
    $this->save();
    return $this->credibility;
}
```

FIGURE 5.8: Calculation of user credibility
Code in file: app/User.php

5.6.2 Calculation of Service reputation

As seen with the formula of service reputation 4.2.5, this code is the responsible for calculating a the reputation score for a given service.

```
public static function calculateReputation($service)
{
    $serviceRatings = (new BlockchainManager())->getServiceRatings($service);

    if( ! count($serviceRatings)) {
        return 0;
    }
    $reputation = 0;
    $credibility_sum = 0;
    foreach ($serviceRatings as $score)
    {
        $user = User::find($score['user_id']);
        $user_credibility = $user->credibility;
        $credibility_sum += $user_credibility;
        $reputation += $user_credibility * ($score['score']+0)  ;
    }
    return $reputation / $credibility_sum;
}
```

FIGURE 5.9: Calculation of Service reputation
Code in file: app/ReputationManager.php

5.7 System boot

To boot the system, as with any PHP application, Xampp server is started on local machine, which then can be a server of the system to other nodes on the same network.

The system is provided with a login and registration interface. The system's admin have more privileges than other users.

5.7.1 Admin privileges

The main privileges that the admin has are:
- Edit service registries,
- See all users and can triggers their credibility calculation manually,
- Triggers reputation calculation manually
- Regenerate test data.

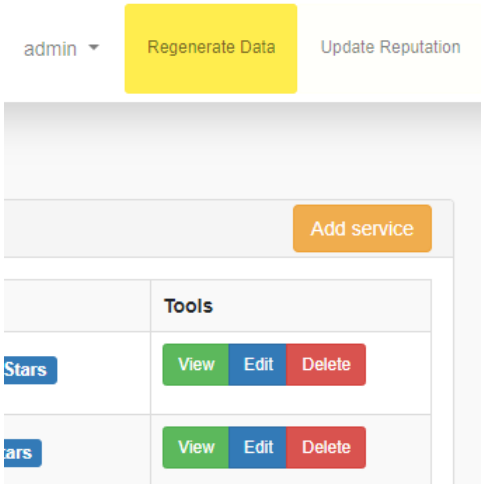

FIGURE 5.10: Admin privileges 1 screenshot

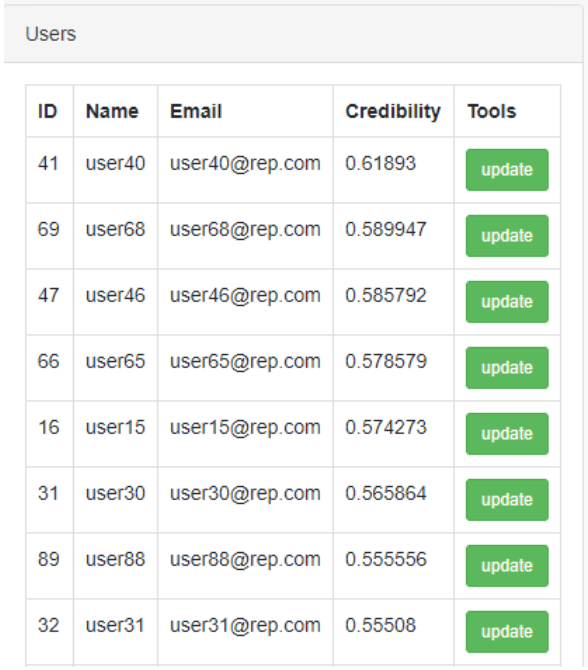

FIGURE 5.11: Admin privileges 2 screenshot

5.8 Test Data

5.8.1 Users

Regarding system performance and testing, We have pre-created a total number of 99 Users and one admin.

id	name	email
1	admin	admin@rep.com
2	user1	user1@rep.com
3	user2	user2@rep.com
4	user3	user3@rep.com
5	user4	user4@rep.com
6	user5	user5@rep.com
7	user6	user6@rep.com
8	user7	user7@rep.com
9	user8	user8@rep.com
10	user9	user9@rep.com
11	user10	user10@rep.com
12	user11	user11@rep.com
13	user12	user12@rep.com
14	user13	user13@rep.com
15	user14	user14@rep.com

FIGURE 5.12: Pre Created Users in the database

5.8.2 Services

We obtained for this, a list of services called **WSDream Dataset** [15]. The list can be downloaded from here: URL.

And we choose about 150 entries randomly.

5.8.3 Generating Interactions

Generating test data is implemented in **BlockchainManager.php** class, in the function "**regenerate(): void**". We followed this algorithm in order to get more realistic test data each time we regenerate it:
1- We get the number of services, and identifications from the registry.

```
$serv_count =  Service::getIds();
```

2- We randomly choose a collection of services, we named them "**golden services**" to give them high reputation scores.

```
$gnum = (int) $serv_count / 4;
$golden_services = self::randomGen(0, $serv_count - 1 , $gnum);
private static function randomGen($min, $max, $quantity) {
        $numbers = range($min, $max);
        shuffle($numbers);
        return array_slice($numbers, 0, $quantity);
}
```

3- Then, for each user, we define a random number of interactions he makes. If the current user is the admin, we give him more interactions just to test all the system functionalities.

```
$num_of_scores_to_give
        = random_int(($user == 1)? 10 : 0, (int) $serv_count / 3);
```

4- Each interaction is assigned to a randomly selected service, and if that service is golden then we give it high score.

```
foreach ($chosen_services as $chosenService)
{
        if (in_array($chosenService, $golden_services))
        {
        $score = random_int(
                (int) ReputationManager::maxStars * 2 / 3,
                 ReputationManager::maxStars);
        }
        else
        {
        $score = random_int(1, ReputationManager::maxStars);
        }
}
```

5- then we calculate credibility for all users 5.6.1 and reputation for all services 5.6.2.

5.9 Remark

During realization of this system, we faced some performance issues since the systems data is huge and it needs a powerful workstation in order to handle all the system's capabilities. For that reason, the supplied source code is intended to simulate the system running on low-end workstation.

Chapter 6

Conclusion

In this work we studied trust/reputation systems, we have seen that centralized architectures have a lot of risks. The most important to this work was the transparency factor.

As a solution we proposed the adaptation of Blockchain technology into the system, enforcing the systems transparency and users trusting the system itself, as this technology provides the magical factor of total immutability in system interactions and saved data.

Furthermore, the architecture of the system is based on a new technology (blockchain), which has its own set of issues. The scalability, transaction costs, and usability of blockchain technology are currently being contested.

Appendix A

Executing the supplied Source code

For any faced difficulties, we have already hosted the application to a free web hosting service online, so you can test it directly without having to go through the following steps.

The ready for test application URL:
https://reputation-system.000webhostapp.com/blockchain/blockchain/public/login
We are demonstrating with a windows machine.

A.1 Xampp server

Install Xampp server, and start Apache and MySQL servers.

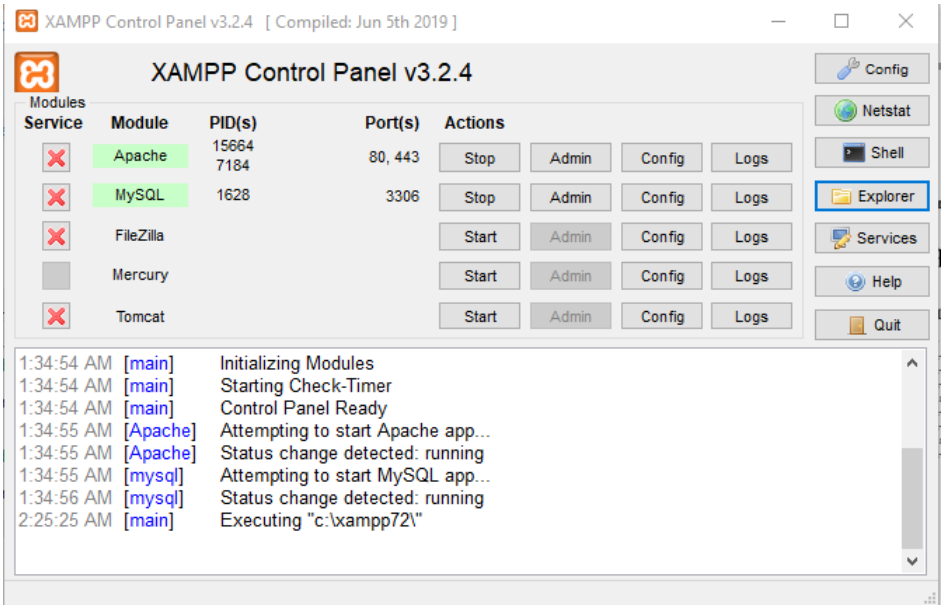

FIGURE A.1: Xampp server screenshot

A.2 Database Initialize

Go to this url, which open phpmyadmin interface.
- Create a new database, and name it "**blockchain**".
- Enter that database, then click "**Import**" button at the top bar.
- Select the supplied **blockchain.sql** file, and import it by clicking Go button at the bottom of the page.

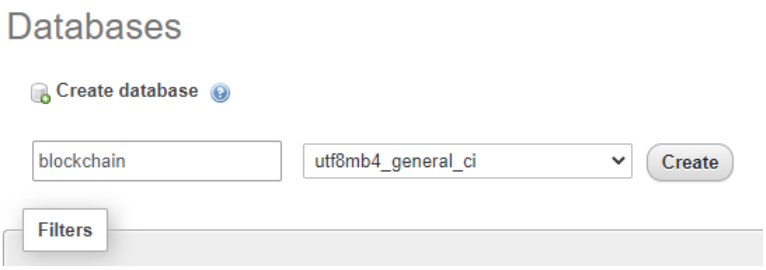

FIGURE A.2: Create database

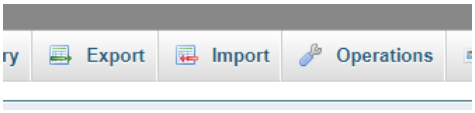

FIGURE A.3: Import button

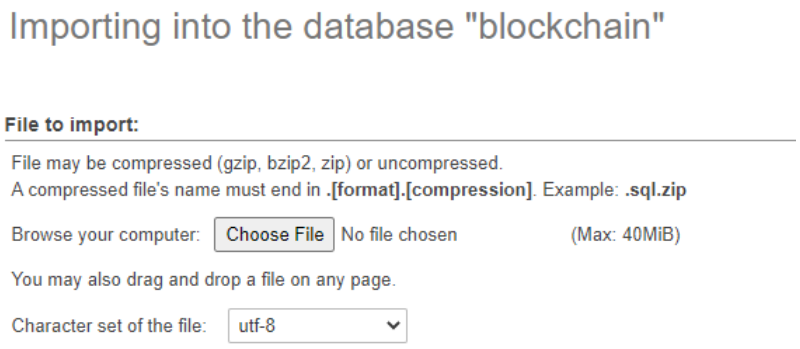

FIGURE A.4: Select the .sql file

A.3 Application Host

Then copy the supplied source code into the folder:

```
$xampp_instalation_folder$/htdocs/
```

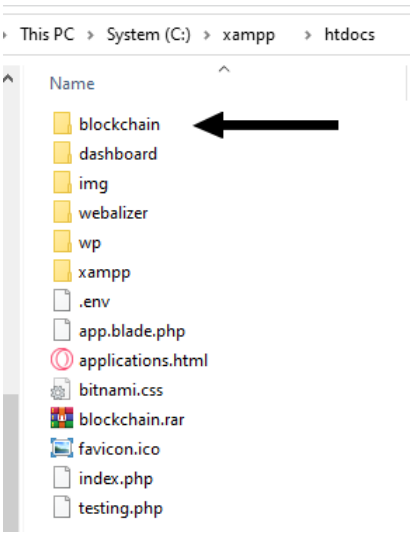

FIGURE A.5: Xampp Folder screenshot

A.4 Login

And then access this link to get to the login interface:

Application Url

The login credentials for the admin are pre-entered, you can directly click on login button:
- email: admin@rep.com
- passw: 12345678

And the login credentials for other users are in this form:
- email: user[number]@rep.com
- passw: 12345678

just replace [number] with the wanted user number,. ex: user9@rep.com

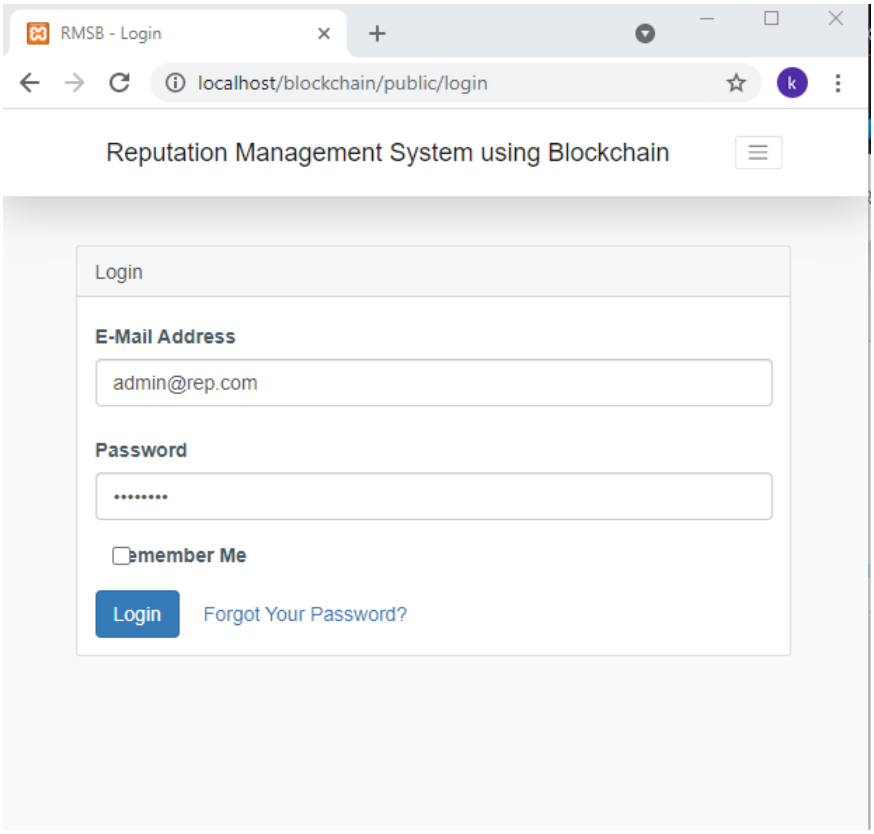

FIGURE A.6: Login interface screenshot

Bibliography

[1] *flavienbwk/blockchain-php: An object-oriented PHP library for creating a blockchain easily.* June 30, 2021. URL: https://github.com/flavienbwk/blockchain-php (visited on 06/30/2021).

[2] Diego Gambetta. *Can we trust trust?, D. Gambetta (Ed.), Trust*. 1988.

[3] *Hyperledger Fabric vs. NEO Comparison | IT Central Station*. June 26, 2021. URL: https://www.itcentralstation.com/products/comparisons/hyperledger-fabric_vs_neo (visited on 06/26/2021).

[4] Audun Jøsang and Jennifer Golbeck. "Challenges for robust trust and reputation systems". In: *Proceedings of the 5th International Workshop on Security and Trust Management (SMT 2009), Saint Malo, France*. Vol. 5. 9. Citeseer. 2009.

[5] Eleni Koutrouli and Aphrodite Tsalgatidou. "Taxonomy of attacks and defense mechanisms in P2P reputation systems—Lessons for reputation system designers". In: *Computer Science Review* 6.2-3 (2012), pp. 47–70.

[6] *Laravel - Wikipedia*. June 28, 2021. URL: https://en.wikipedia.org/wiki/Laravel (visited on 06/29/2021).

[7] Félix Gómez Mármol and Gregorio Martínez Pérez. "Security threats scenarios in trust and reputation models for distributed systems". In: *computers & security* 28.7 (2009), pp. 545–556.

[8] *Reputation system - Wikipedia*. Feb. 15, 2021. URL: https://en.wikipedia.org/wiki/Reputation_system (visited on 02/18/2021).

[9] Pablo Lamela Seijas, S. Thompson, and Darryl McAdams. "Scripting smart contracts for distributed ledger technology". In: *IACR Cryptol. ePrint Arch.* 2016 (2016), p. 1156.

[10] Voshmgir Shermin. "Disrupting governance with blockchains and smart contracts". In: *Strategic Change* 26.5 (2017), pp. 499–509. DOI: https://doi.org/10.1002/jsc.2150. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jsc.2150. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jsc.2150.

[11] *Smart Contract - Overview, How It Works, Role in Blockchain Tech*. June 26, 2021. URL: https://corporatefinanceinstitute.com/resources/knowledge/deals/smart-contract/ (visited on 06/26/2021).

[12] *Smart Contracts: The Blockchain Technology That Will Replace Lawyers*. Blockgeeks. Oct. 9, 2016. URL: https://blockgeeks.com/guides/smart-contracts/ (visited on 02/15/2021).

[13] Yan Sun and Yuhong Liu. "Security of online reputation systems: The evolution of attacks and defenses". In: *IEEE Signal Processing Magazine* 29.2 (2012), pp. 87–97.

[14] Nick Szabo. "Smart Contracts : Building Blocks for Digital Markets". In: 2018.

[15] WS-DREAM Team. *WS-DREAM: Towards Open Datasets and Source Code for Web Service Research*. Feb. 2, 2019. URL: https://wsdream.github.io/ (visited on 06/30/2021).

[16] *What is Blockchain Technology? A Step-by-Step Guide For Beginners*. Sept. 19, 2016. URL: https://blockgeeks.com/guides/what-is-blockchain-technology/ (visited on 02/15/2021).

[17] *XAMPP - Wikipedia*. June 27, 2021. URL: https://en.wikipedia.org/wiki/XAMPP (visited on 06/29/2021).