REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : SIOD21/M2/2021

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Systèmes d'information, Optimisation et Décision (SIOD)**

Blockchain-Based Application for Transparent and Genuine Financial Assistance

---

Par :

ZEHANA MOHAMED DIYA EDDINE

Soutenu le 27/06/2022 devant le jury composé de :

Zerarka M. Faouzi	MCA	Président
Guerrouf Fayçal	MCB	Rapporteur
Bendahmane Toufik	MAA	Examinateur

Année universitaire 2021-2022

Abstract

A large number of donation transactions are provided daily to the poor and needy to finance them. These transactions have been formally carried out from donors to the poor through intermediaries for decades. But the donations do not reach the beneficiaries in full. This is the cause of the mistrust between donors and intermediaries, especially due to the massive lack of transparency. Using Blockchain technology to address issues of poor transparency and lack of confidence that donations will reach the intended beneficiaries is an ideal solution as it relies on a decentralized tracking system and provides complete transparency. The latter allows donors to be fully aware of the steps involved in getting their donations to the intended recipients. A decentralized system based on Blockchain technology is introduced in this project to solve the problem of financial donations.

key-words : Blockchain, Financial donations, Smart contract, Transaction, Decentralization, Traceability.

Résume

Un grand nombre d'opérations de dons sont fournies quotidiennement aux pauvres et aux nécessiteux pour les financer. Ces transactions ont été formellement effectuées par des donateurs aux pauvres via des intermédiaires pendant des décennies. Cependant, les dons ne parviennent pas intégralement aux bénéficiaires. C'est la cause de la méfiance entre les donateurs et les intermédiaires, notamment en raison du manque massif de transparence. L'utilisation de la technologie Blockchain pour résoudre les problèmes de manque de transparence et de manque de confiance dans le fait que les dons atteindront leurs destinataires est une solution idéale car elle s'appuie sur un système de traçabilité décentralisé et offre une transparence totale. Ce dernier permet aux donateurs d'être pleinement conscients des étapes nécessaires à l'acheminement de leurs dons vers leurs destinataires. Dans ce projet, un système décentralisé basé sur la technologie Blockchain est présenté pour résoudre ce problème de dons financiers.

Mots clé : Blockchain, Dons financiers, Contrat intelligent, Transaction, Décentralisation, Traçabilité.

ملخص

يتم تقديم عدد كبير من عمليات التبرع يوميًا للفقراء والمحتاجين لتمويلهم. تم تنفيذ هذه المعاملات رسميًا من المانحين إلى الفقراء من خلال وسطاء لعقود من الزمن. ومع ذلك ، فإن التبرعات لا تصل بالكامل إلى المستفيدين. هذا هو سبب انعدام الثقة بين المانحين والوسطاء ، خاصة بسبب الافتقار الهائل للشفافية. يعد استخدام تقنية سلسلة الكتل لمعالجة قضايا ضعف الشفافية وانعدام الثقة في وصول التبرعات إلى المستفيدين المقصودين حلاً مثاليًا لأنه يعتمد على نظام تتبع لامركزي ويوفر شفافية كاملة. يتيح هذا الأخير للمانحين أن يكونوا على دراية كاملة بالخطوات المتبعة في إيصال تبرعاتهم إلى المستفيدين المقصودين. في هذا المشروع، يتم تقديم نظام لامركزي يعتمد على تقنية سلسلة الكتل لحل مشكلة التبرعات المالية.

الكلمات المفتاحية : سلسلة الكتل، التبرعات المالية، العقود الذكية، المعاملات، اللامركزية، التتبع.

DEDICATION

This master thesis is dedicated to my loving parents, my lovely mother and my father. I will never forget their ever-lasting support and their unconditional love.

I should also thank my dear brothers and sisters, Khaled and Younes, Anfell and Hiba, for their concern and constant help.

ACKNOWLEDGEMENTS

My current work would not have been done without the help of Allah, the most gracious and most merciful to whom I owe my existence.

I would like to thank my dear supervisor, Dr. Guerrouf Fayçal, for his tremendous efforts and guidance. His advice is always precious and beneficial and he was always there for me.

Special thanks to the board of examiners who accepted to evaluate my work and devote their precious time to provide me with the essential remarks and comments about my work.

I'd like to thank my dear friends who provided me with the needed support, distinguished professors, and everyone who contributed to my success.

Contents

List of Figures

General Introduction

Blockchain has swept the current debate, from tech circles to governments all over the world. Every day, we learn about the incredible potential of this new technology, as evidenced by initiatives like the UN World Food Programme's "Blockchain Against Hunger" program in Jordanian refugee camps. The World Identity Network's advocacy for the adoption of self-sovereign identity based on Blockchain is to combat human trafficking, and the startup Follow My Vote's initiative to increase transparency and protect against fraud elections [41][5][8]. Blockchain technology is based on the concept of continuously growing records stored in blocks that are linked together using cryptographic hashes [46]. Previously, Blockchain was used as a digital currency, such as bitcoin [38].

Donors are always trying to help the poor and needy people through donations. These operations are carried out through intermediaries. But these donations do not reach the beneficiaries in full. This caused mistrust between donors and mediators, especially due to the massive lack of transparency.

This project proposes a decentralized donation system based on Blockchain technology. It is based on eliminating the role of intermediary, preventing system manipulation, and increasing the digital transparency and credibility of data, where the Blockchain is used instead of relying on organizations and institutions like central banks and governments to provide security features and anti counterfeiting measures. The relevance of incorporating this technology into donation systems is that it allows for the tracking and management of donations that reach the needy, as well as the security and anonymity of the donation process, which contributes to donor trust. This item is suitable for donations.

This master thesis contains four chapters, the first chapter presents the basic concepts of Blockchain technology. The second chapter presents everything related to financial donation in general and its application to Blockchain. The third chapter presents the proposed system design and UML modeling, and the fourth chapter presents the implementation of the proposed system. Finally, we end our thesis with a general conclusion.

Chapter 1

Blockchain Technology

1.1 Introduction

The term "Blockchain" was first proposed in Satoshi Nakamoto's paper in 2008, which carries the promise of disintermediation and transparency at a time when third parties and traditional mediators institutions, banks, and states are experiencing a crisis of confidence and dissatisfaction. Blockchain has become one of the most talked-about technologies on the internet, and an increase in projects based on it has been seen ever since.

This chapter covers the basic concepts of this technology.

1.2 History of Blockchin

The idea of Blockchain began in the eighties and it has evolved since then to its current state:

1983: David Chaum, a computer scientist, proposed the e-Cash concept.

1990: David Chaum's DigiCash attempted to put the e-Cash concept into practice. In 1998, the company declared bankruptcy.

1991: Stuart Haber and W Scott Stornetta describe for the first time a cryptographically secure chain of blocks.

1997: Adam Back, a computer scientist, created Hashcash. It's similar to Bitcoin's underlying technology, but it's less secure [28].

1998: Nick Szabo, a computer scientist, is working on 'bit gold,' a decentralized digital currency.

2000: Stefan Konst publishes his cryptographic secure chain theory, as well as implementation suggestions.

2008: A white paper establishing the model for a Blockchain is released by developers working under the pseudonym Satoshi Nakamoto.

2009 : Nakamoto creates the first Blockchain as the public ledger for bitcoin transactions.

2014 : The potential of Blockchain technology for other financial and interorganizational transactions is investigated after it is separated from the currency. The term "Blockchain 2.0" is coined to refer to applications other than currency.

The Ethereum Blockchain system incorporates computer programs that represent financial instruments such as bonds into the blocks. These are referred to as smart contracts [3].

1.3 Definition of Blockchain

Blockchain is a decentralized, unchangeable ledger that makes it more easier to record transactions and track assets in a corporate network. A tangible asset (a home, vehicle, cash, or piece of land) can also be an intangible asset (intellectual property, patents, copyrights, branding). On a Blockchain network, virtually anything of value can be tracked and traded, lowering risk and costs for all parties involved [31].

A Blockchain is a network of computers that replicates transactional data across all computers (nodes) in the system. A distributed ledger is the name for this type of data. Blocks of data are entered into the chain at regular intervals. Each block is timestamped, and the order and transactions of each block are verified. This method of storing data in duplicate results in a Blockchain, or a chain of transactions [18].

Blockchain is a method of storing data in a distributed ledger, which is a collection of data that is shared among many people and locations. This can reduce the risk of fraud and data mismanagement while also speeding up transactions. The ledger keeps track of transactions in a secure way that can't be altered. It eliminates the need for third-party intermediaries like banks by allowing any two parties to transact directly [5].

1.4 Components and structure of Blockchain

Blockchain can be simplified by looking at each component separately : cryptographic hash functions, transactions, ledgers, blocks, node, and how blocks are chained together.

1.4.1 Node

It is the user, computer, or server within a Blockchain structure who has a separate copy of the entire blockchain's ledger. Nodes are used by developers to create blockchain-based applications.

1.4.2 Transaction

A transaction is a two- or more-party agreement, contract, exchange, understanding, or transfer of assets/cash or property that creates a legal obligation. Transactions are first recorded in a journal, then transferred to a ledger. A transaction is when a buyer and a seller exchange goods or services. Every transaction is made up of three parts: First, goods/services and money are transferred second, title is transferred, which may or may not be accompanied by possession; and third, exchange rights are transferred.

A Blockchain transaction is nothing more than a public record of all Blockchain transactions that have ever taken place. The figure 1.1 shows an illustration of the transaction process :

Figure 1.1: Transaction process

1.4.3 Blocks

A block is a data structure that collects transactions for inclusion in the Blockchain, which is a public ledger. The block is made up of a metadata-filled header and a long list of transactions that make up the majority of its size. The block header is 80 bytes long, while the average transaction is at least 250 bytes long and the average block contains over 500 transactions. As a result, a complete block, including all transactions, is 1,000 times larger than the block header and the block header is used to manage all of the blocks [10]. The figure 1.2 shows the components of a block.

1. **Block Header:** The Blockchain is defined as a collection of interconnected blocks. The headers of the blocks, however, are the only ones that are actually linked together in this way. The block's head is divided into six sections:

- **Timestamp** The timestamp is used as proof in the Blockchain that a particular block was used at what point in time, and it is also used as a parameter to verify the authenticity of any block.
- **Version**It specifies which Blockchain version the block is using, there are three types of Blockchain versions.
  - Blockchain Version 1.0(cryptocurrency) It stored the data on a public ledger, such as Bitcoin.
  - Blockchain Version 2.0(smart Contract) Self-executing programs, such as Ethereum, are referred to as smart contracts.
  - Blockchain Version 3.0(DAPPS)- Tor Browser, for example, uses it to create a decentralized structure.
  - Blockchain Version 4.0(Blockchain for Industry)- It is used to create a scalable, affordable Blockchain network that can be used by a larger number of people.
- **Merkle Root** uses mathematical formulas to see if the data has been tampered with, hacked, or manipulated in any way. For example, if a block has ten transactions, we'll need ten transactions to combine and form one Hash Value to identify it. To do so, it employs the binary tree concept to create the block's hash, which is referred to as the Merkle Root.

  Binary Hash Trees are another name for Merkle trees. Merkle Trees are binary trees containing cryptographic hashes. The term "tree" comes from the field of computer science, where it refers to a branching data structure.
- **Difficulty Target** It specifies the complexity and computation power required to mine the network; if the difficulty target is high, it means that mining it will require a computationally expensive machine.
- **Nonce** It is an abbreviation for 'number only used once,' and it is a number that Blockchain miners are looking for, with the correct nonce taking almost ten times on average. A nonce is a 32-bit number with a maximum value of 2^3 total possible value, so bitcoin miners must find the correct integer value, which is a random integer between 0 and 2^{32}, which is computationally costly.
- **Previous Hash** Because a block is made up of several interconnected nodes, previous hash stores the hashed value of the previous node's address. The Genesis Block is the first block in the Blockchain and has no previous block hash value. This block hash is created using the SHA256 algorithm's cryptographic methods and has a size of 32 bytes.

2. **Block Data:** A block of data is a list of validated transactions and ledger events that have been submitted to the Blockchain network.

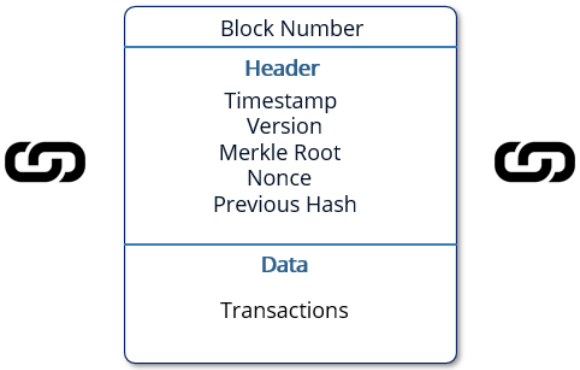

Figure 1.2: Block structure

The Genesis Block: The genesis block is the first block in any Blockchain. In other words, if it is started at any block and the work might set it backward in time, it will eventually reach the genesis block.

1.4.4 Chaining blocks

Chaining Blocks are made up of orderly linking blocks that contain transaction data. The Blockchain is formed by chaining together blocks, each of which contains the hash digest of the previous block's header. A different hash would be generated if a previously published block was changed. As a result, all subsequent blocks will have different hashes because they will include the previous block's hash. This makes it simple to spot and reject tampered blocks [55]. The figure 1.3 shows an illustration of the chaining blocks.

Figure 1.3: Chaining blocks

1.4.5 Cryptographic hash functions

Hashing is a method of calculating a relatively unique output (called a message digest, or simply digest) for an input of nearly any size using a cryptographic hash function (e.g., a file, text, or image). Individuals can independently take input data, hash that data, and derive the same result, proving that the data hasn't changed. Even the tiniest change in the input (such as a single bit) results in a completely different output digest [55].

A hash function is a one-way mathematical algorithm that takes an input and converts it into a hash or digest output. Hashing functions are fundamental to Blockchain technology and have a long history in computer science. It's important not to mix up hashing and encryption. A file is encrypted with a key and decrypted with a key when using encryption. There is no decryption step in hashing. Furthermore, a good hashing algorithm makes finding two input values that produce the same hash value (output) computationally impossible; this is known as collision resistance [43][33].

The Secure Hash Algorithm (SHA) is a cryptographic hash function with a 256-bit output size that is used in many Blockchain implementations (SHA-256). This algorithm is supported in hardware on many computers, making it quick to compute. SHA-256 produces a 64-character hexadecimal string with a 32-byte output (1 byte = 8 bits, 32 bytes = 256 bits) [55]. The figure 1.4 shows an illustration of the hash function.

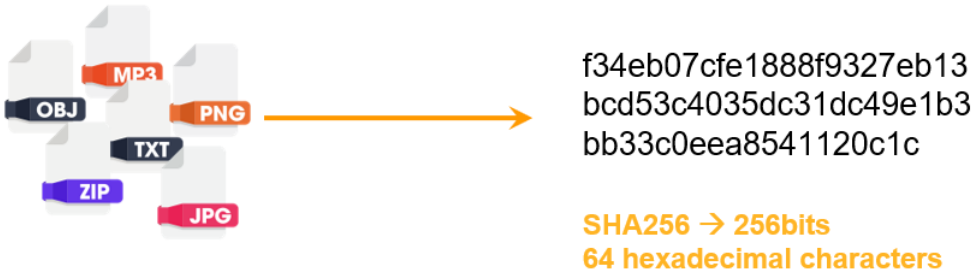

Figure 1.4: Hash functions

1.4.6 Smart contracts

Smart contracts are essentially small computer programs that are stored on a blockchain and will carry out a transaction if certain conditions are met. As a result, a smart contract is typically a statement like "transfer X to Y if Z occurs." Unlike a traditional contract, which requires parties to execute the contract after reaching an agreement, a smart contract is self-executing. Which means that once the instructions are written to a Blockchain, the transaction will take place automatically when

the appropriate conditions are detected, with no additional actions required by the parties to the transaction or other third parties [29]. The figure 1.5 shows an illustration of the smart contract.

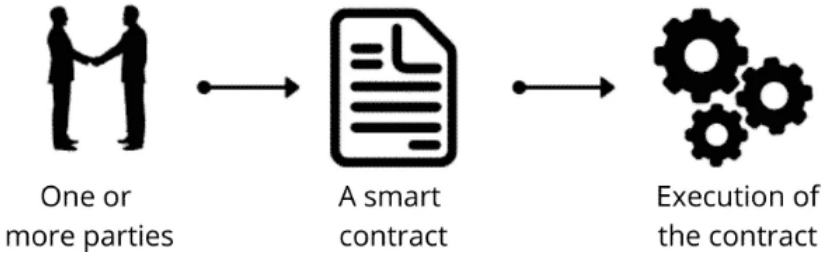

Figure 1.5: Smart contract

1.4.7 Consensus mechanisms

Any node in a Blockchain network has the ability to propose new information to be added to the Blockchain. The nodes must reach some sort of agreement in order to validate whether this addition of information (for example, a transaction record) is legitimate. A "consensus mechanism" kicks in at this point. In simple terms, a consensus mechanism is a predefined specific (cryptographic) validation method that ensures correct transaction sequencing on the Blockchain [32].

There are several ways to structure a consensus mechanism. The Proof of Work ("PoW") and Proof of Stake ("PoS") mechanisms are the two most well-known:

1.4.7.1 Proof of Work (PoW)

To add new blocks to the Blockchain in a PoW system, network participants must solve so-called cryptographic puzzles. Mining is a term used to describe the process of solving puzzles. Simply put, these cryptographic puzzles are made up of all previously recorded information on the Blockchain, as well as a new set of transactions to be added to the next block. The PoW mechanism necessitates a vast amount of computing resources, which consume a significant amount of electricity, because the input of each puzzle grows larger over time (resulting in a more complex calculation) [40][26].

When a network participant (also known as a node) solves a cryptographic puzzle, it verifies that he has completed the task and is rewarded with a digital form of value (or, in the case of a cryptocurrency, a newly mined coin). This reward serves as a motivator to maintain the network [40].

A PoW consensus mechanism underpins the cryptocurrency Bitcoin. Litecoin, Bitcoin Cash, Monero, and other cryptocurrencies are examples.

1.4.7.2 Proof of Stake (PoS)

To participate in the validation of transactions in a PoS system, a transaction validator (i.e. a network node) must prove ownership of a specific asset (or, in the case of cryptocurrencies, a specific amount of coins). Instead of mining, this process of validating transactions is referred to as "forging". To validate a transaction In the context of cryptocurrency for example, a transaction validator will have to prove his stake (i.e., his share) of all coins in existence. He will have a better chance of validating the next block depending on how many coins he owns (This is due to the fact that he has more seniority inside the network, granting him a more trusted position) [26].

The transacting parties pay the transaction validator a transaction fee for his validation services [34].

A PoS consensus mechanism is used by cryptocurrencies like Neo and Ada (Cardano) [2].

1.5 Peer-to-peer network

P2P (peer-to-peer) is a technology based on the principle of decentralization. Because of blockchain's peer-to-peer architecture, all cryptocurrencies can be transferred globally without the use of a middleman, intermediaries, or a central server. Anyone who wants to participate in the process of verifying and validating blocks can set up a node on the distributed peer-to-peer network.

A peer-to-peer system It refers to a decentralized peer-to-peer network in which all computers are linked and where each computer keeps a complete copy of the ledger and compares it to other devices to ensure data accuracy. In contrast to a bank, where transactions are stored privately and managed solely by the bank, this is not the case.

The interaction between two parties through a peer-to-peer model is easily accomplished with the use of Blockchain, and no third party is required. Blockchain is based on the peer-to-peer (P2P) protocol, which allows all network participants to have an identical copy of transactions, allowing for machine consensus approval. For example, if you want to make a transaction from one part of the world to another, you can do so in a matter of seconds using Blockchain. Furthermore, any delays or additional fees will not be deducted from the transfer [37]. The figure 1.6 shows an illustration of a peer-to-peer network with Blockchain technology.

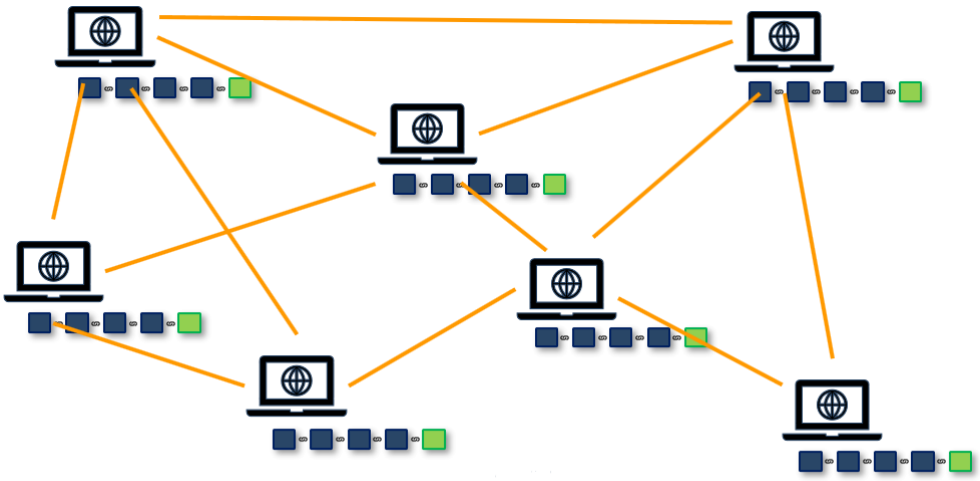

Figure 1.6: Blockchain for peer-to-peer network

1.6 Distributed ledger technology (DLT)

Blockchain records are distributed among all users rather than having a single owner. The approach's genius lies in its use of a complicated system of consensus and verification to ensure that, despite the lack of a central owner and time lags between all users, there is still a single, agreed-upon version of the truth.

Each "node" in a Blockchain keeps a copy of all the historical transactions that have been added to the ledger, and each record is kept synchronized by comparing it to the copies of the other nodes.

There is no node with special rights to edit or delete transactions, as there is in a traditional ledger system, and there is no central party at all [3]. Each transaction is encrypted by converting an unencrypted input (i.e., plain text) into an encrypted output using an algorithm and a "key" (i.e., ciphertext). As a result, distributed ledger transactions are expected to be [28]:

- Processed in a timely and cost-effective manner
- Validated by the network's entire membership.
- Less prone to mistakes
- Almost impenetrable by hackers

The figure 1.7 depicts the difference between the traditional (centralised) ledger system and the distributed ledger system.

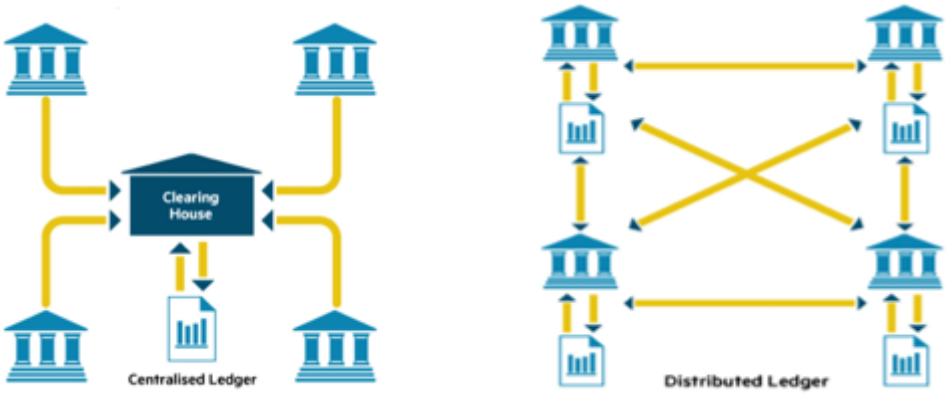

Figure 1.7: Centralised ledger and distributed ledger

1.7 Types of Blockchain

There are four different major types of Blockchain technologies. They include the following:

1.7.1 Public Blockchian

Public Blockchain can be accessed and verified by all nodes in the network. It is a permission distributed ledger technology (DTL) where anyone can join and perform transactions. It is an unrestricted copy where each peer owns a copy of the ledger. This also means that anyone can access the public Blockchain. Only a set of predefined nodes will participate in the consensus process of the Blockchain consortium, as no one is in charge. Proof of work (POW) and Proof of Stake (POS) are taken into account for making decisions. Some popular public Blockchain are Ethereum and Bitcoin [32] The figure 1.8 depicts a public Blockchain.

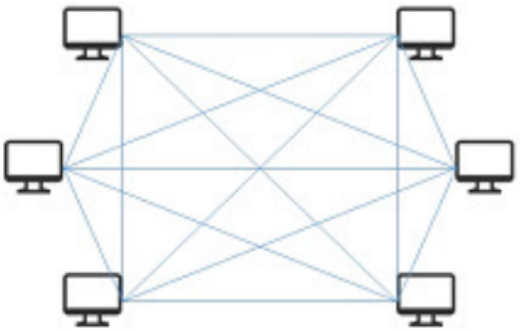

Figure 1.8: Public Blockchain

1.7.2 Private Blockchian

Private Blockchain are limited to an individual or an organization. The body is responsible for the read and write operations, i.e. a closed network. This authority is also responsible for selectively granting read and write access to users and can also grant mining rights selectively. Therefore, only authorized nodes can access certain transactions from the Blockchain or participate in the work to publish new blocks. In this way, the privacy of transactions is improved and the authority to verify transactions is decentralized to the control of the organization. Furthermore, with a high level of trust between nodes in the licensed network, a computation-intensive consensus algorithm may not be needed [32][36].

The main differences from the public Blockchain are in the way it is accessed and the private Blockchain is somewhat centralized as only one authority looks at the network. Therefore, it does not have a theoretical, decentralized nature, and they share features such as transparency, trust, and security for specific participants.Common examples of Private blockchains are Blockchain, Multichain, Hyperledger Fabric, Hyperledger Sawtooth, and Corda [32]. The figure 1.9 depicts a private Blockchain.

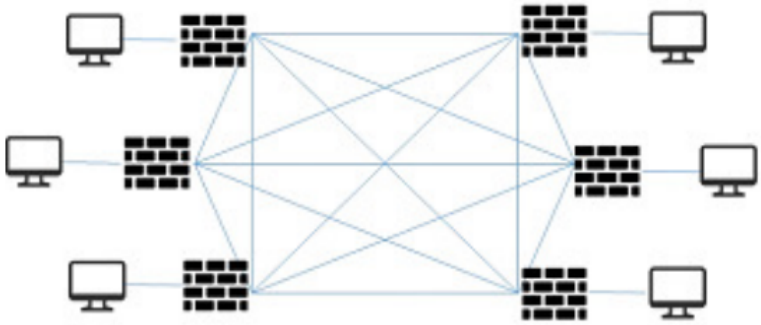

Figure 1.9: Private Blockchain

1.7.3 Consortium Blockchian

The Blockchain consortium is one of the different types of Blockchain technology. A Blockchain consortium(also known as Federated Blockchain) is a mixture of private and public Blockchain features, some aspects of the organizations are made public, while others remain private and participants can only join if invited, however. There is no single organization that controls the network but instead a group of them, they maintain the decentralized nature of a public Blockchain, eliminating the risks that come with just one entity controlling the network on a private Blockchain. Common examples of consortium Blockchain are r3 and EWF (Energy Web Foundation)

[20][32]. The figure 1.10 depicts a consortium Blockchain.

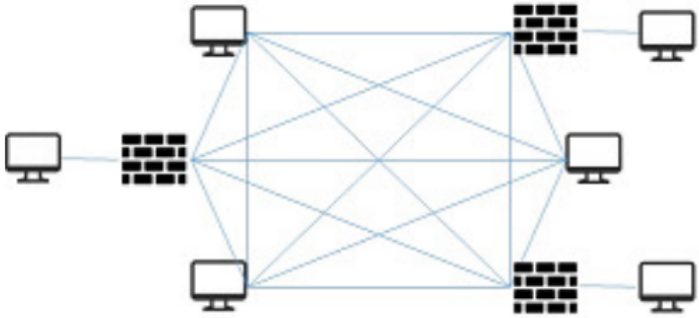

Figure 1.10: Consortium Blockchain

1.7.4 Hybrid Blockchian

Hybrid Blockchain It is similar to a consortium Blockchain, in that it combines the elements of a private and public Blockchain. Allows organizations to create a private permission-based system alongside a public permission-less system, allowing them to control who can access specific data stored in the Blockchain, have use cases in an organization that neither wants to publish a private Blockchain nor a public Blockchain and wants to publish the best of what in the worlds [15].

1.8 Cryptography in Blockchain

The most important aspect of Blockchain is cryptography. It is unquestionably a research field in and of itself, based on advanced mathematical techniques that are quite difficult to comprehend. This section will introduce some cryptographic concepts.

Plaintext refers to any information in the form of a text message, numeric data, or a computer program. The idea is to encrypt plaintext with an encryption algorithm and a key, resulting in ciphertext. The intended recipient can then decrypt the ciphertext using the decryption algorithm and key to obtain the plaintext [51].

1.8.1 Types of Blockchain cryptography

There are two types of cryptography: symmetric key cryptography and asymmetric key cryptography (also known as public key cryptography). The sections below demonstrate this.

1.8.1.1 Symmetric Key cryptography

Symmetric key cryptography, also known as secret-key or shared-key cryptography, is a type of cryptography that uses two keys. The sender and receiver share a common key for both encryption and decryption in this type of mechanism as shown in figure 1.11 [13]. The method is based on self-certification, which means that the key is self-certified. The key needs to be shared through secret communication. If it is compromised, the attacker can easily decrypt the encrypted message. This type of cryptographic technique is necessary because it allows for faster service while consuming fewer resources.

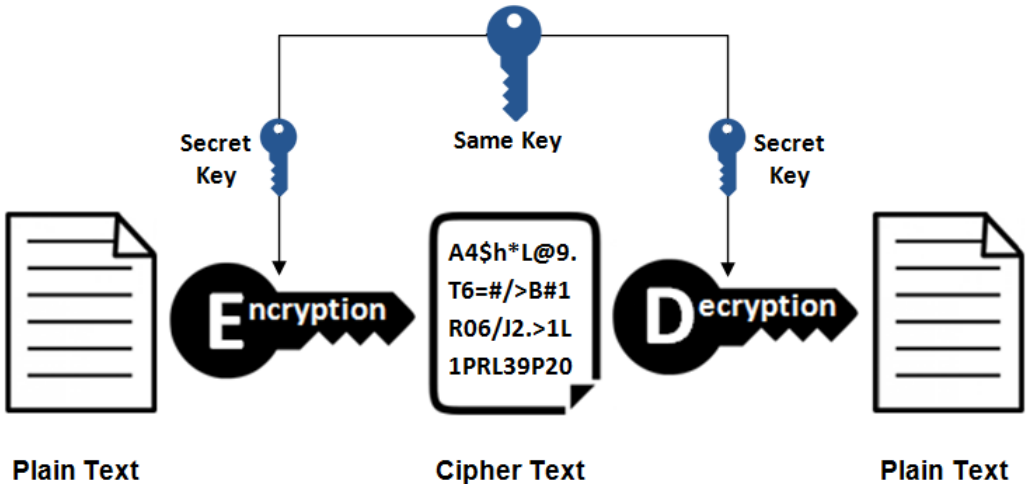

Figure 1.11: Symmetric key cryptography

1.8.1.2 Asymmetric Key cryptography

Whitfield Diffie and Martin Hellman invented public key cryptography in 1976. As a result, it's sometimes referred to as Diffie-Hellman encryption. Because it uses two keys instead of one, it's also known as asymmetric encryption.

Asymmetric-key cryptography is a cryptographic system that uses two keys: a public key that is known to everyone and a private or secret key that is known only to the message recipient as shown in figure 1.12. The public and private keys are linked in such a way that only the public key can encrypt messages and only the corresponding private key can decrypt them, which is an important feature of the public key system. Furthermore, knowing the public key makes it nearly impossible to deduce the private key [12].

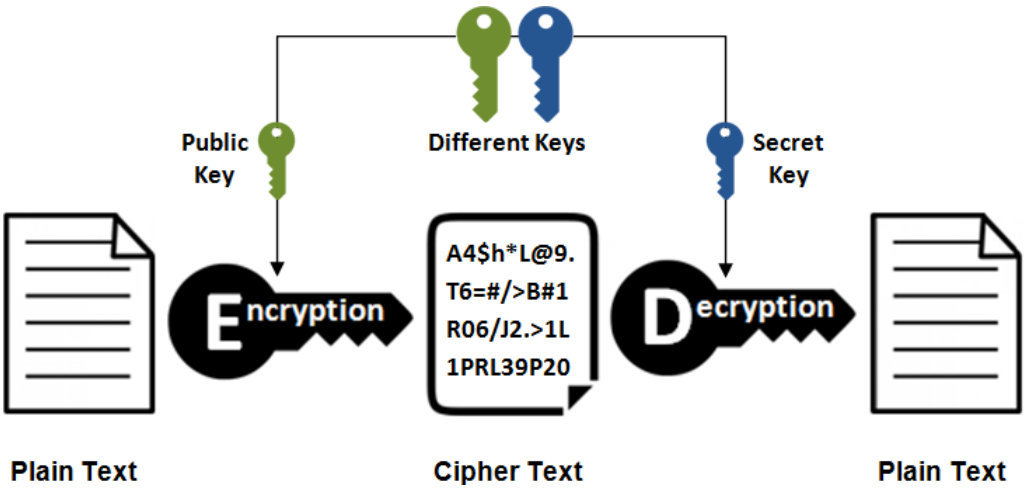

Figure 1.12: Asymmetric key cryptography

1.9 Blockchain applications

1.9.1 Cryptocurrencies

Cryptocurrencies are a new way of thinking about money. Their promise is to streamline existing financial infrastructure in order to make it more efficient and less expensive. Their technology and architecture decentralize existing monetary systems, allowing transacting parties to exchange value and money without the need for middlemen like banks.

Cryptocurrencies is a digital asset designed to work as a medium of exchange using cryptography to secure transactions, to control the creation of additional value units, and to verify the transfer of assets. Many different cryptocurrencies exist. each with their own set of rules and are meant to be extremely secure, with almost little danger of counterfeiting [25].

There are many currencies today, the most famous of which are Bitcoin and Ether.

1.9.1.1 Bitcoin

The most well-known and valuable cryptocurrency is Bitcoin. It was invented and introduced to the world in 2008 by an anonymous person named Satoshi Nakamoto through a white paper [25]. Bitcoin was one of the first cryptocurrencies to make use of Blockchain technology to make peer-to-peer payments possible. When compared to traditional payment gateways, bitcoin provides a relatively cheap transaction charge due to its decentralized network.

The first step is to obtain a bitcoin wallet, which is software that allows you to send, receive, and securely store bitcoins. It's available for download on your

phone, PC, or any other digital device. The second step is to earn bitcoins by trading, playing online games such as Bitcoin blackjack, or asking bitcoin payments from clients. Bitcoin is unlike any other money that is controlled by a central bank.

Bitcoins are not physically stored on any platform, and they are protected by a mathematical algorithm that encrypts a string of numbers called a public and private key [7].

1.9.1.2 Ethereum

Ethereum is an open-source, decentralized Blockchain technology with its own coin, Ether. ETH serves as a platform for a variety of different cryptocurrencies as well as decentralized smart contract execution.

Is a system that allows anyone to transmit bitcoin for a modest price to anyone else. It also powers open-source programs that no one can take down.

Vitalik Buterin initially defined thereum in a whitepaper published in 2013. In the summer of 2014, Buterin and his co-founders raised funds for the project through an online public crowd sale [1].

1.9.2 Electronic voting Blockchain

By providing a cost-effective and secure e-voting system, Blockchain technology can enable a more bottom-up and participatory social structure [42]. To distribute individual voting data to thousands of computers around the world, it is impossible to modify or delete votes once they have been cast. By reducing the number of people needed to run elections and providing officials with immediate results, the Blockchain protocol would also ensure electoral transparency [16].

1.9.3 Supply chain management

The Blockchain technology ensures that the entire supply chain process is transparent. It allows businesses to track goods from the point of origin to the point of delivery. These trucking's are precise and allow for a better handling of goods and their condition [49].

Blockchain can help supply chains detect unethical suppliers and counterfeit products, which can cause serious social harm [49].

1.10 Advantages and disadvantages of Blockchain

1.10.1 Advantages

- **Decentralization** It is a decentralized, distributed structure.The entire network does not have a centralized hardware or mechanism in traditional centralized transaction systems, which is the primary feature of the Blockchain. Unlike traditional centralized transaction systems, every transaction needs to be validated by a trusted central agency (for example, a central bank) [54].

- **Transparency** The Blockchain ensures transparency as each node on the network constantly shares information about the ledger, creating a permanent record that is public and accessible to the rest of the nodes [48].

- **Traceability** Traceability means the ability to find data in the chain, identify and trace assets (data, materials, processes) as well as actions, where an attempt to make fraudulent changes to the Blockchain can be detected by other nodes.

- **Anonymity** The user can interact with the Blockchain using a pseudonym and not a real name, which does not reveal the real identity of the user [56].

- **Persistency** Transactions can be quickly verified. Invalid transactions may be cancelled By honest miners. Transactions that are already part of the Blockchain cannot be deleted or undone. Blocks containing invalid transactions can be easily detected [56].

- **Security** Security is one of the main advantages as it is difficult for any type of attack to be successful with more people working on the network, so it is impossible for malicious actors take over the network and attack the system.

1.10.2 Disadvantages

- **Energy Consumption** It's expensive, energy consumption is a concern. The bitcoin consensus algorithm is based on Proof of Work (POW), which uses a large amount of electrical resources to operate. However, there are other consensus mechanisms and algorithms such as Proof of Stake (POS) that use much less electricity.

- **Slow** Experiencing a lack of processing speed. It may take a few minutes to hours before the transaction can be completed. For example, Bitcoin can only

manage seven transactions per second and Ethereum can perform 15 transactions per second compared to 24,000 transactions per second by VISA. This is because it takes time to solve the math problems and then complete the transaction [32].

- **Scalability** It cannot scale due to how they work. The more nodes that are joined, the slower the network becomes. There are steps taken to solve the problem.

1.11 Conclusion

The use of Blockchain technology is expanding beyond the realm of cryptocurrencies. It appeared to be used in a many areas, and many people are interested in and considering developing it. As can be seen, the current drawbacks are primarily due to the technology's novelty, which must be further developed in order to achieve efficiency. The financial donation system and its application to the blockchain are the subject of the next chapter.

Chapter 2

Financıal Donation

2.1 Introduction

In this chapter, we will explain all about financial donation in a general way, starting with a global identifıcation and followed with specifıc information from types to processes of donation mentioning the importance of the donation system found in non-profit organizations and payment mechanisms and also addressing the challenges facing organizations without Blockchain technology. By using the Blockchain in donation, the donor is informed about where his/her donation is going using the Blockchain tracking feature and gets a notification when it reaches the beneficiary. Once the donor makes the donation, the transaction is secured by smart contracts and proof of work where the locked payment cannot be tampered with.

At the conclusion of the chapter, the work environment that the study is based upon is clarified.

2.2 Definitıon of financial donation

A donation is a gift of cash or property made to a nonprofit organization to help it accomplish its goals, for which the donor receives nothing of value in return. In some countries donations can be deducted from the federal tax returns of individuals and companies making them [6].

To be classed as a donation or grant, a receipt of funds or assets must have been freely given, with no consequent obligation on the charity to provide goods or services to the benefit of the donor [6].

2.3 Types of income treated as donations

Donation revenue The term "donation revenue" as used in these processes can refer to any of the following: gifts, benefactions, bequests, legacies, grants from charitable trusts, grants from governmental departments and agencies. Research Grants and HEFCE revenue are not included in this. Make sure the revenue is not a research grant or any other type of consulting by exercising caution. Either way, the revenue is a result of a contractual arrangement and must be handled as such. Contracts and funding for research are handled by the ROO.

Donations can take the shape of one-time payments in cash, checks or credit card payments, waivers of fees, investments, consistent donations made through payroll giving, bank standing orders or other methods, or non-cash assets.

2.4 Income which is not a donation

2.4.1 Research grants

The Finance Division or ROO should be consulted if an offer of assistance seems to straddle the line between a research grant and a contribution since specific donor perks (use of logos, permission to attend seminars, access to academics, etc.) may give rise to trading difficulties that need to be handled.

No portion of money may be considered a contribution once it has been determined to be a research grant. The funding is either a grant for study or a contribution, not both.

Financial Regulation 13.4 complies with this requirement, stating that "all research grant or contract income and expenditure, from whatever source of funds, must be notified to the Research Office and that no part of this income may be transferred into donation accounts or other special funds, except for funding remaining unspent at the end of the research which the funder has agreed the Department may retain."

2.4.2 Excess research grant income

This is not a contribution and is credited to the departmental accounts for EDFF and EDBA sources of funding. It is still a part of the original deal made between the University and the funding source for the study.

Although it may be unclear, the final clause of Regulation 13.4 provides for the case when we must return any unused funds to the sponsor but the sponsor afterwards expressly declares that they would like to contribute the same amount back to

the university. However, in reality, the funds might never leave the university, Technically, there are two distinct transactions. Departments should communicate with ROO and Central Finance in these exceptional circumstances because any funds must be shifted to either a H or E source of funds code.

2.4.3 Sponsorshipe

If the University gets sponsorship money, it must recognize the sponsor's contribution. The revenue is not a contribution to the degree that it provides value to the sponsor. Consult the Tax Team in the Finance Division for guidance on how donations and sponsorship money are divided up (together with the associated VAT status).

2.4.4 Office of Students/HEFCE grants

Donations are not considered to be made by the Office of Students (formerly known as the Higher Education Funding Council for England, or HEFCE) or the Teacher Training Agency (TTA), regardless of whether they are attached to particular programs or have spending caps. These money received and the related expenses are identified using the J*** source of funds. The J Sources of Funds are used to account for non-research funding obtained from the Higher Education Funding Council for England (HEFCE) and other organizations for specified purposes, together with any related spending, whether it be revenue spending or capital equipment spending. 'J' Source of funding use restrictions Construction and renovation projects are not funded by J Sources of Funds. Unrestricted grants from HEFCE are credited to the Chest rather than J Sources of Funds, which are not utilised for them (Source of Funds AFAA).

2.4.5 Trading income

In exchange for the provision of products or services, the use of space or facilities, or both, a department may receive trade revenue from either another department within the university or from a third party. The revenue is not to be considered income from donations.

However, when a person forgoes personal fees received through consulting or external lectures earned outside the University and instead transfers money to the Department, this is regarded as contribution income.

Contracts for consulting and trading are handled through subsidiaries like Cambridge Enterprise Limited, which were created for this purpose, or, if made by the

University itself, utilizing the trade accounts (GAAA sources of funds). What trade activities charity may legitimately engage in are governed by rules.

2.5 Principles and practices on the acceptance of donations

2.5.1 Consultation with CUDAR re solicitation of donors

Before addressing any potential contributors, departments should follow these steps. When speaking with a possible donor, any University employee who is active in fundraising should first contact CUDAR. Early consultation can: lower the likelihood of a single potential donor receiving many haphazard efforts. familiarize them with the procedure for accepting benefits. assistance on how to apply the Proceeds of Crime Act and the ethical principles make sure that donations are only taken from and used for reasons that the university approves of. Permit anybody unintentionally approaching a possible benefactor whose gift is not likely to be approved to receive an early warning.

2.5.2 Authority to accept donations under 100,000 pound

Donations are accepted, and accounts are set up for them, with the understanding that they are charitable contributions that belong to the university rather than a specific person. Although the Head of Institution may put money at someone's disposal, it still belongs to the University.

Under the Vice-authority, Chancellor's heads of institutions may receive one-time contributions of up to £100,000, unless they have a personal stake in the donation, such as a job or research program that would benefit from it. In these situations, they must as soon as possible inform the Director of Development and Alumni of this potential conflict of interest.

Therefore, it is the duty of heads of institutions to see that all policies and moral principles are followed. It is crucial that CUDAR is alerted whenever gifts are collected so that they may be centrally recorded, monitored, and properly recognized. This will help to guarantee that contributors receive the appropriate amount of appreciation for their assistance.

The Vice-Chancellor must formally accept gifts worth more than £100,000

2.6 Types of financial donation

2.6.1 Cause-related marketing (CRM)

The defining feature of CRM is the linkage of a firm's contributions to a charitable cause to revenue-producing transactions with the firm [52]. CRM) is a mutually beneficial collaboration between a corporation and a nonprofit designed to promote the former's sales and the latter's cause.American Express first coined the term in 1983 to describe its campaign to raise money for the Statue of Liberty's restoration. American Express donated one cent to the restoration every time someone used its charge card. As a result, the Restoration Fund raised over $1.7 million and American Express card use rose 27% [4].

2.6.2 Unconditional donations

(donation to a cause that is not link-ed to revenue producing transactions with the firm) unconditional donation has no strings attached, except that the cause must often agree to the use of its name and logo by the firm in announcing the donation to the public [19] For example, Pearle Vision Center announced a $45,000 donation to the Children's Miracle Network without indicating whether or how this support was tied to corporate sales [11].

2.7 Non-profit organizations

non-profit organizations represent a group of private, voluntary and non-profit organizations and associations, they describe a group of organizations and activities, so they considered like community institutions (whether government or public sector) on the one hand and on the other represents a third sector of government because of their great importance. When a nonprofit charitable organization is qualified internationally the donors can generally deduct their contributions to the charity on their tax returns [47].

Nonprofit organizations collect donations, whether through funding or individual donors, via payment methods whether through nonprofit organizations websites, or bank accounts [14].

The donation process is done through the following steps [35]:

1. The donor goes to the non-profit organization's website, browses the projects, and selects the one for which he wants to donate.

2. The credit card information is entered by the donor on the web page, and the payment gateway transfers the credit card numbers. Some entities need to track the credit card charging time in each period of time if they have recurring information.

3. There is a near-instant authorization process that informs all parties whether or not a transaction will be completed within a time frame that could take hours or even a whole day.

4. As shown in Figure 2.1, money and information flow into the CRM database, where an email is sent to the donor once the transaction is completed.

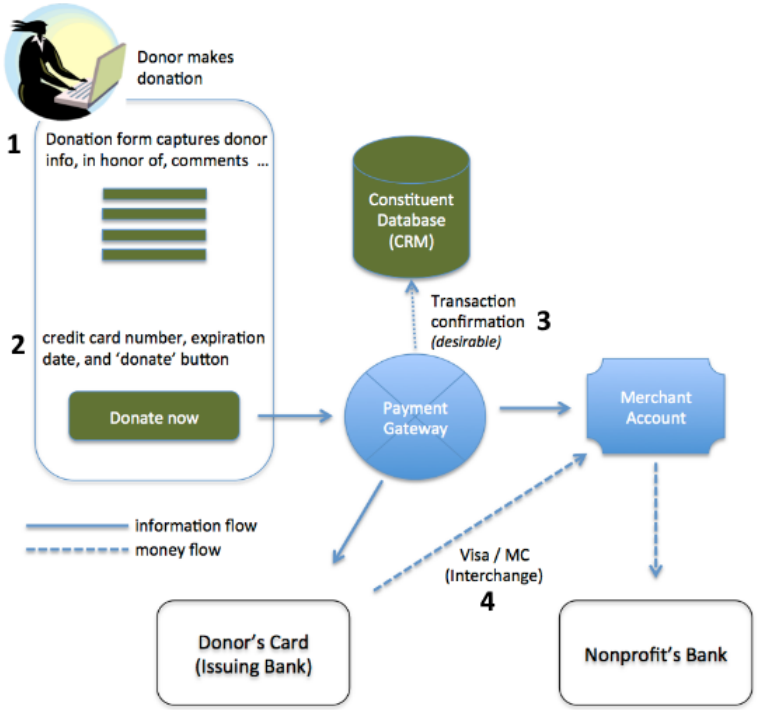

Figure 2.1: Donation process

2.8 The problem of financial donation

Donors have distrust about how donated money is spent cause of the lack of transparency has made people lose trust in charities, making social funding stagnant. Donor is unaware of the legitimate utilization of his funds.Corruption adds to the distrust of the donor [44].

Online donation systems for non-profit organizations have been known in many ways since their inception until now and are often characterized by a lack of trans-

parency,which scientific efforts have been made to address through many technologies, the most important of which is the Blockchain.

2.9 Financial donation system using Blockchain

After a marked decline in trust in non-profit organizations as the number of donors and donations decreased, the demand for more public scrutiny, accountability and transparency in donation processes increased, hence the need to use Blockchain technology as a new way to increase trust between donors and nonprofits by adding more transparency about how the donation process is tracked using the Blockchain [23].

The payment methods without using Blockchain technology face some challenges, namely that sending money is limited to a specific amount, and that the money reaches the other party takes longer than one day.

These organizations often have problems accessing finance in terms of institutional inefficiency or mistrust of financial transactions on the web [45].

Therefore, the use of Blockchain technology solves these problems, hence further analyzes of the donation system using the Blockchain have been carried out in several scientific papers.

2.10 Conclusion

We learned about financial donations and the numerous challenges they face due to a lack of transparency in this chapter, and we proposed Blockchain as a solution to this problem. In the following chapter, we'll propose a money donation system based on Blockchain technology to solve the problems with the financial donations system.

Chapter 3

System design

3.1 Introduction

We attempted to design a Blockchain-based system for managing financial donation data while maintaining user privacy in this chapter. We will present our system's global architecture as well as its detailed architecture, including various diagrams such as use case diagrams, class diagrams, admin, donor, and poor sequence diagrams. After that, we come to a conclusion.

3.2 Global Architecture

Decentralized applications (dApps) are digital applications or programs that exist and run on a Blockchain or peer-to-peer (P2P) network of computers instead of a single computer. are outside the purview and control of a single authority. DApps which are often built on the Ethereum platform [24].

Dapps are more transparent in their operation and more powerful which aim to eliminate the intermediaries who are omnipresent, and to improve traceability and transparency of information and that is what make us propose a decentralized system for managing Financial Assistance, in addition it have provided the ability to store and retrieve records from the Blockchain ledger, which makes it easy to track the transactions and ensure that the data cannot be changed. The figure 3.1 shows the global architecture of our application system, where the associated actors (admin, donor and poor) interact with the Blockchain through the interface and interface talk to the server to interact with and retrieve the data from the Blockchain network, then send it back to the user through the interface.

But this server is not like a traditional central server, it is considered a node. Where the Blockchain is a peer-to-peer network of computers, called nodes, that

share all the data and code in the network. So, if a device is connected to the Blockchain, it is a node in the network, and it talks to all other computer nodes in the network. He has a copy of all the data and token on the Blockchain. There is no central servers. Just a group of computers that talk to each other on the same network.

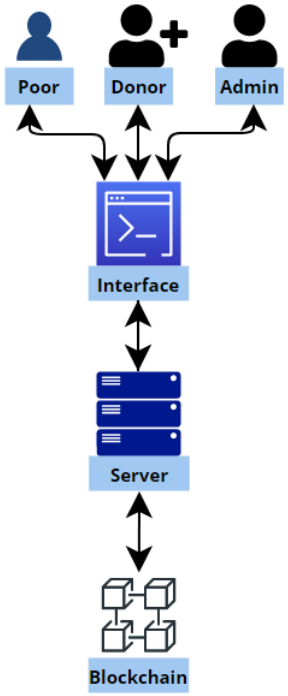

Figure 3.1: Global Architecture

3.3 System diagrams

3.3.1 Use case diagram

The figure 3.2 depicts a use case diagram for our system, which explains the role of each system actor.

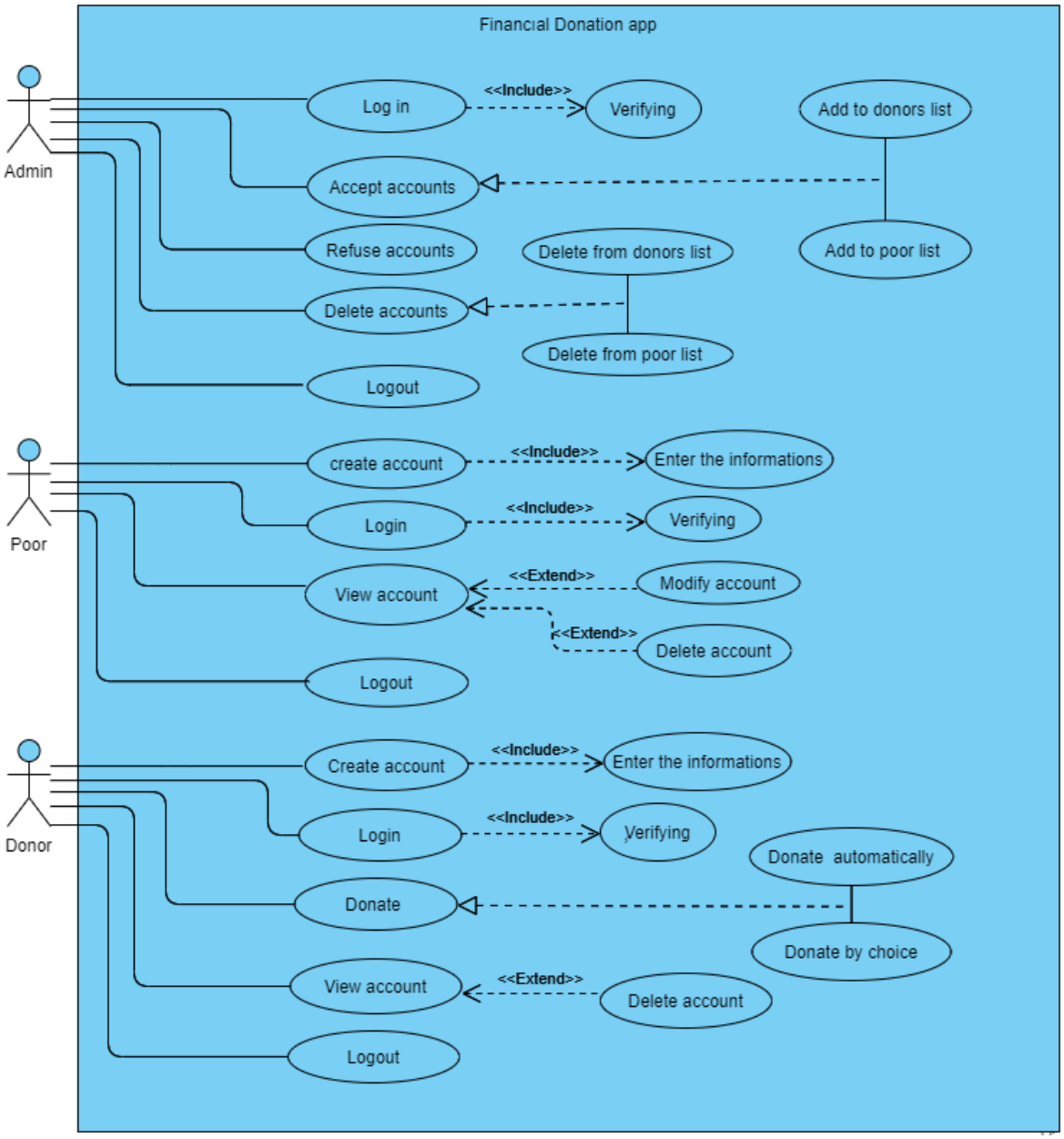

Figure 3.2: Use case diagram

3.3.2 Class diagram

the class diagram shown in the figure 3.3 shows the relationship and functions of the poor, the donor and the admin in our application.

The first task that the poor performs is to create an account that needs to enter his information, which is his full name, age, gender, his password, his monthly income, his marital status, the number of his children, his living address and a situation where he can explain his condition and why he needs financial assistance. This

information may help him to obtain financial assistance, and he has an attribute called donated which determines whether he has benefited from financial assistance or not.

The second task it does is log in if it has been accepted by the admin by entering a full name, password and be connected with owner address by wallet, which owner address is a unique address that identifies the user in the Blockchain. The third task that the poor person does after proving his authenticity is to either modify his account or delete it and finally he can log out.

A poor account request is accepted once by the admin, and he receives financial aid only once by the donor.

As for the donor, the first task that he performs, like the poor, is to create an account by entering his information, which is his full name, his password, gender, age and living address. The second task is to log in also like the poor. The third task that takes place after proving authenticity is a donation, and the system is the one who chooses the poor who deserves this financial assistance, or makes a donation by choosing the poor. And he can delete his account and in the end he can log out.

the donor account request is accepted once by the admin, And he can donate many times to the poor

As for admin, the first task he performs is to log in by entering his full name and password. The second task is to accept or refuse the requests of the poor and donors, and he can delete the accounts after accepting them, and in the end he can log out.

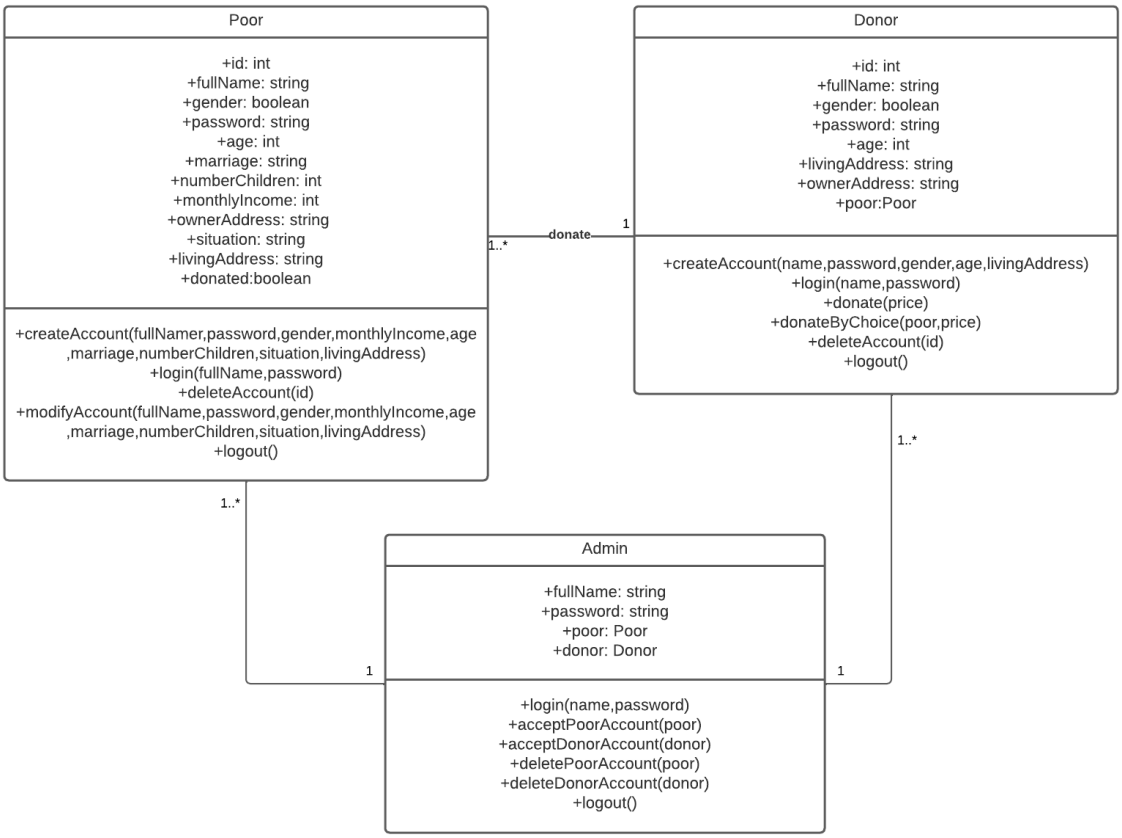

Figure 3.3: Class diagram

3.3.3 Poor sequence diagram

The "poor" sequence diagram shown in the figure 3.4 shows the interaction and all the steps of the poor with the system where the first step is to connect to the digital wallet, it helps to connect the address that is used in our application and receive the money. Followed by creating a poor account, it needs to fill out a form by entering the credentials such as full name, number of children, etc. through the interface, to the server and then sent to the Blockchain. This is information that may help him join(storage) to the list of donation requests if he is accepted by the admin if the necessary conditions are met, and thus his information helps him to obtain a fair donation. The next step is to log in with the same scenario previous step sends the login information through the interface, to the server and then sends it to the Blockchain and verifies the information. Once poor logs in, he will be able to see his information and can modify it or delete his account. and the most important step is to receive the money donated to him via the wallet.

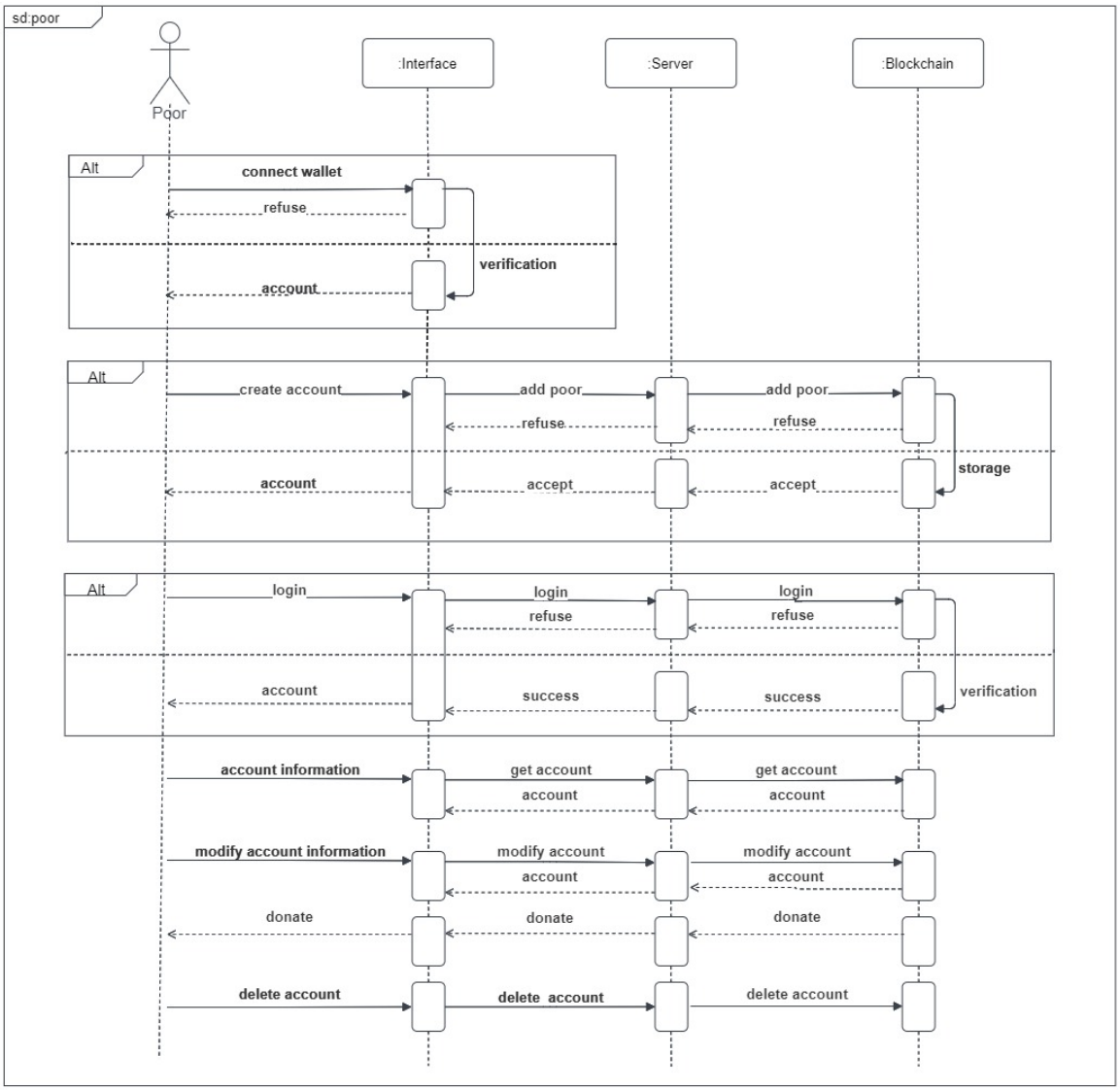

Figure 3.4: Poor sequence diagram

3.3.4 Donor sequence diagram

The "donor" sequence diagram shown in the figure 3.5 shows the interaction of the donor and all its steps with the system where the first step is to connect to the digital wallet, it helps to connect the address used in our application and send money. Followed by the creation of the donor account, it needs to fill out a form by entering the credentials through the interface, to the server and then send it to the Blockchain. This is information that may help him to be accepted by the admin if the necessary conditions are met. The next step is to login in the same scenario previous step is to send the login information through the interface, to the server and then send it

to the Blockchain and verify the information. Once the donor is logs in, they will be able to see their information or delete their account. The most important step is to send money (donation) by entering the price of the donation in the interface, and it may be refused in the Blockchain (smart contract) if he himself is poor or if he does not have enough money, and this is done in two ways, either he chooses a method automatically where the system is the one who chooses the best poor who deserves to donate from among the group of the poor, and this is through the algorithm shown in 1, or by a method of by choice where the donor can see the list of the poor and see their information that may help him choose the poor who wants to donate to him.

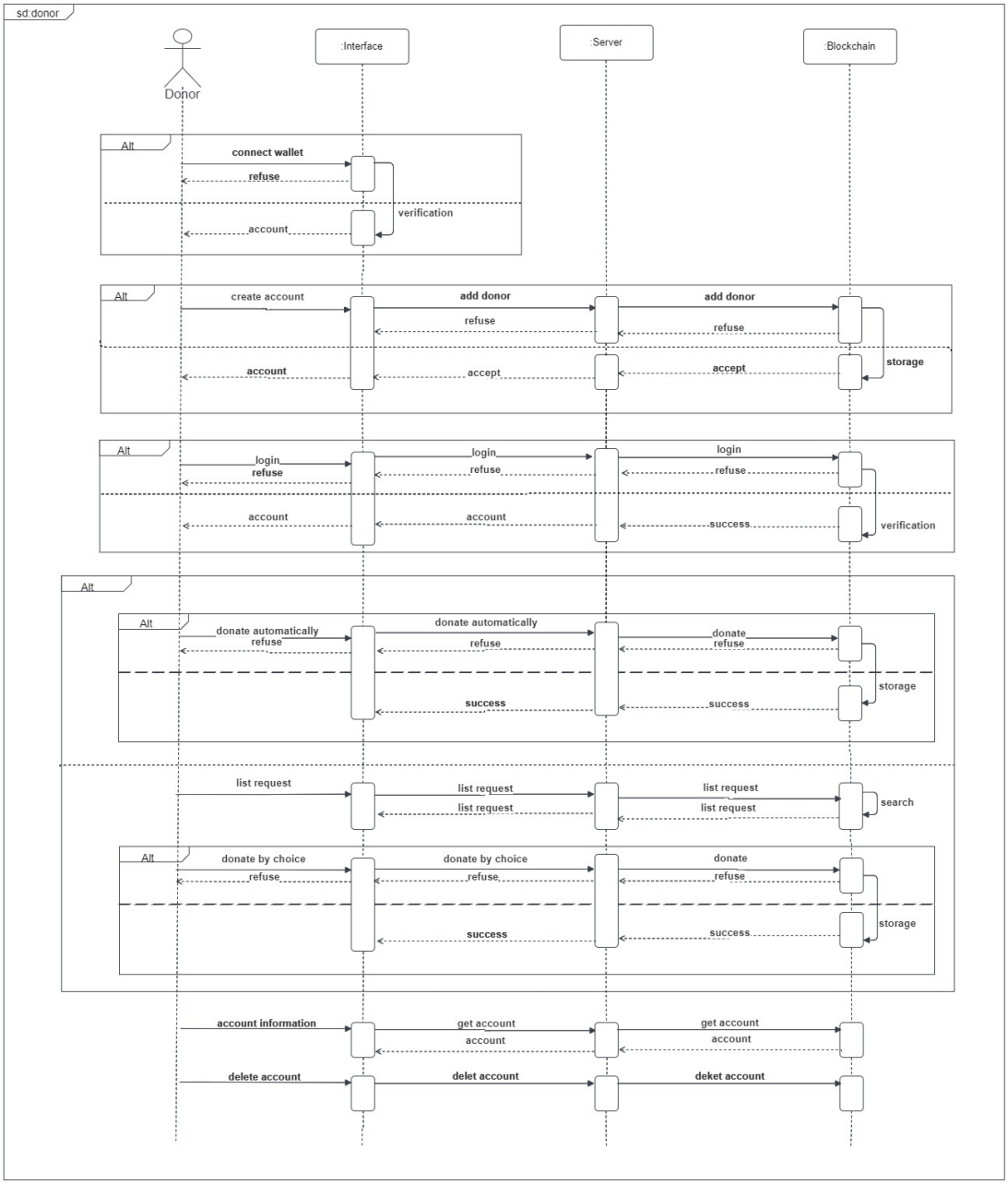

Figure 3.5: Donor sequence diagram

Donation algorithm: The donation algorithm 1 shows below showing the function donatePoor. This function receives the variable price, which is the value that the donor donated. This function calculates an average for each poor person from the list of the poor. The average of each poor person is calculated according to his age, the number of his children, his monthly income in dollar currency, and whether he is married or not. Whoever gets the highest average retains the ID that distinguishes him from the rest of the poor. And in the end, the donation is made for him using

his own ID and the price that the donor donated.

Algorithm 1 Donation algorithm

```
1: function DONATEPOOR(price)
2:     max ← 0
3:     i ← 1
4:     while i ≤ numberPoor do
5:         marriage ← poor(i).marriage
6:         numberChildren ← poor(i).numberChildren
7:         age ← poor(i).age
8:         monthlyIncome ← poor(i).monthlyIncome
9:         donated ← poor(i).donated
10:        if donated = false then
11:            if marriage = true and numberChildren = 0 then
12:                moy ← (age/5) + 3 + (300/(monthlyIncome + 4))
13:                if moy > max then
14:                    max ← moy
15:                    id ← i
16:                end if
17:            end if
18:            if marriage = false and numberChildren = 0 then
19:                moy ← (age/5) + (90/(monthlyIncome + 4))
20:                if moy > max then
21:                    max ← moy
22:                    id ← i
23:                end if
24:            else
25:                moy ← (age/5)+(numberChildren∗4)+(300/(monthlyIncome+4))
26:                if moy > max then
27:                    max ← moy
28:                    id ← i
29:                end if
30:            end if
31:        end if
32:        i ← i + 1
33:    end while
34:    donate(id,price)
35: end function
```

3.3.5 Admin sequence diagram

The "administrator" sequence diagram in the figure 3.6 shows the interactions and all steps of the administrator with the system. The first step is to connect to the digital wallet, like the previous schemes, but the login is via the interface and server only. The administrator first needs to authenticate to access the system, where he will log in with his username and password through the interface. Our system sends the data to the server which makes a query to verify the authentication and there is no error in the password or the name. Once the administrator is logs in, the administrator can access the list of requests for poor and donors through the interface, they can accept or reject them, and if they are accepted, an account is created whether it is poor or donor. Administrator can also delete accounts after they are created.

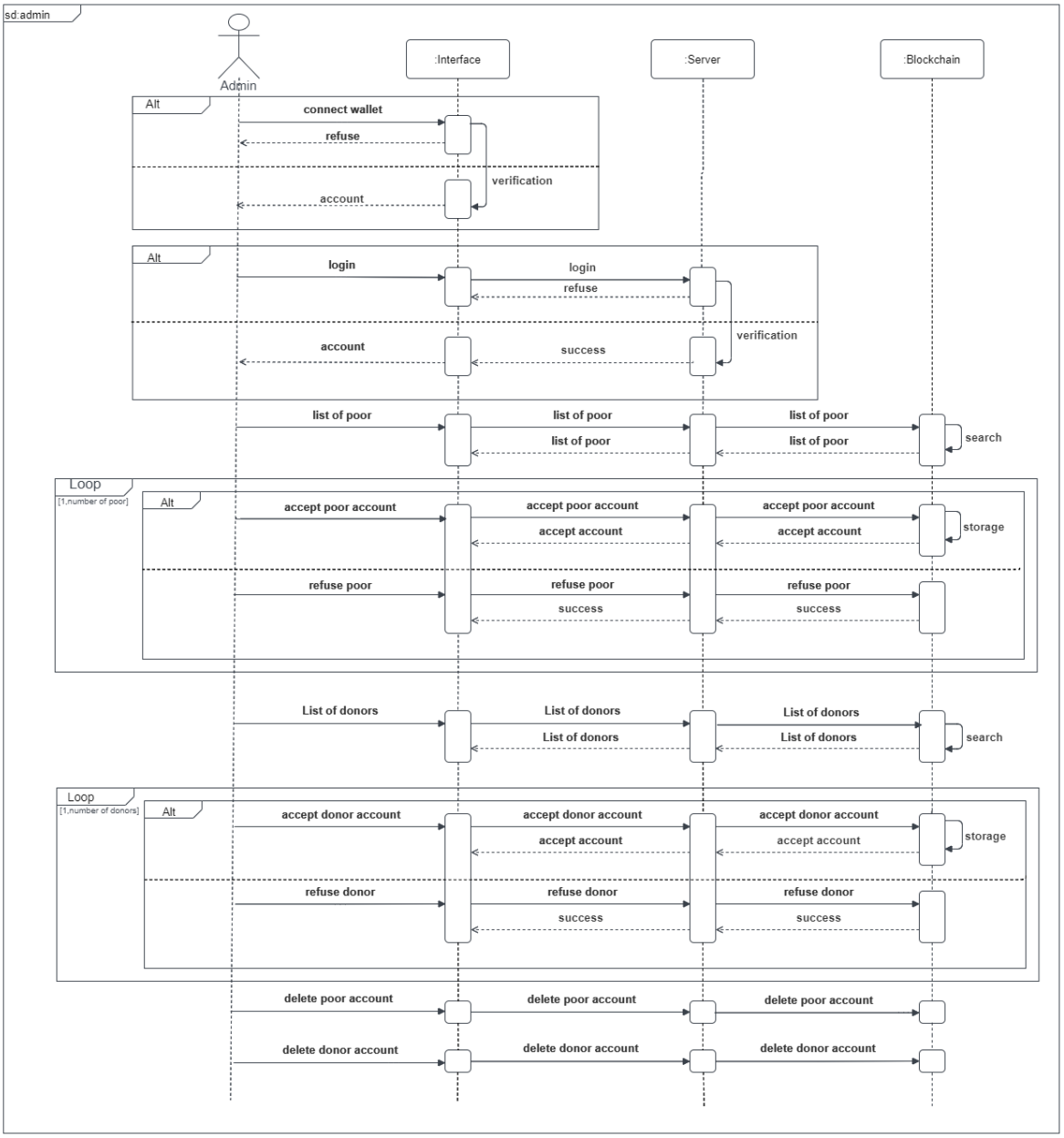

Figure 3.6: Admin sequence diagram

3.4 Conclusion

In this chapter we introduced the general design and proposed to model our system using UML through use case diagrams, class diagrams and sequence diagrams.

The next chapter is devoted to the implementation our systems.

Chapter 4

Implementation

4.1 Introduction

This chapter is dedicated to realizing the application, we will start with the components of the system and move to the development tools and we will implement a system using the Ethereum Blockchain to manage the transactions of financial donations. Using Ethereum smart contracts and configuring a web application to interact with the Blockchain.

4.2 System components

In this section, we discuss the major components of the proposed Ethereum system architecture in financial donation system. Figure 4.1 illustrates cystem components architecture.

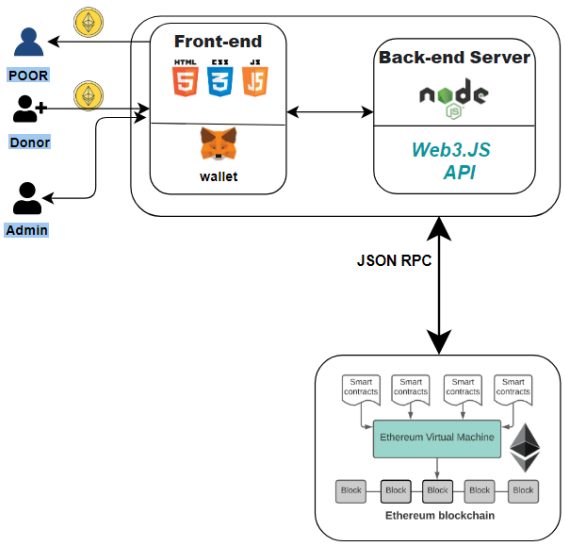

Figure 4.1: System components architecture

4.2.1 Ethereum Blockchain

The Blockchain is the system's most important component. Everyone in the system has access to the same information thanks to this technology. The goal of this technology is to save data in its original form, so that no one can change or modify it after it has been uploaded to the network. Hyperledger and Ethereum are two platforms for developing Blockchain applications. Because it is open source and free, we chose the Ethereum platform to build our decentralized application.

4.2.2 Smart contracts

The Ethereum Virtual Machine (EVM) is the software platform that developers can use to create decentralized applications (DApps) on Ethereum using the Ethereum Blockchain. All Ethereum accounts and smart contracts are stored on this virtual machine.

This contract is a piece of software that runs on the Ethereum Blockchain. Localhost is a local Ethereum network where the contract is deployed.

We establish a connection to a specific node on the local network in order for the contract to be published on that node. Our smart contract's goal is to enable the execution of financial donation transactions without the need for human intervention or trusted third parties.

4.2.3 Web3

Web 3.0 is the next generation of the Internet, with technologies such as Machine Learning (ML), Big Data, and Decentralized Ledger Technology (DLT) allowing websites and applications to intelligently process information. Because of Ethereum inherent decentralization, we chose Web3 to build dapps. Because anyone on the network now has permission to use the service in other words, permission is no longer required, and no one is barred from using it. Payments are now made using Ether tokens (ETH) [50].

Web3.js is a set of libraries that allow us to use HTTP, IPC, or WebSocket to communicate with a local or remote Ethereum node. It is developed and maintained by the Ethereum Foundation, and it serves well for the dapp backend. Use the web3.js package to communicate with an Ethereum node from within a JavaScript application, which provides a user-friendly interface for RPC procedures. This is the JavaScript API.

The Ethereum API allows applications to connect to an Ethereum node that is part of the Ethereum Blockchain. It is possible to interact with on-chain data and

send different types of transactions to the network by utilizing the endpoints provided by the API. The API follows a JSON-RPC standard.

JSON is a data-exchange format that is simple to use. Numbers, strings, ordered sequences of values, and collections of name/value pairs may all be represented using it.

JSON-RPC is a lightweight remote procedure call (RPC) protocol that is stateless. This specification primarily defines a number of data structures as well as the rules that govern their processing. It is transport agnostic in the sense that the concepts can be used within the same process, over sockets, HTTP, or in a variety of other message passing environments.

4.2.4 Backend

The server side of the site is known as the backend. It's a part of an application that can't be seen or interacted with that stores and organizes data. It's a part of the software that doesn't interact with users directly. It is accessed by users indirectly via the front end application. We relied on the language of solidity in smart contract programming and we'll build a Node.js API that will use Web3.js to interact with and retrieve data from the Blockchain network, then send it back to the user on the browser app using React.

4.2.5 Frontend

On an app, the front end is what we see and interact with. It includes everything a user sees directly, from text and colors to buttons, images, and navigation menus. It's also known as the client-side. Web technologies (HTML, CSS, JavaScript) were used for the client interface and React JavaScript library. Interactions with Ethereum, such as signing messages, sending transactions, and managing keys, were done through a web browser, via an extension such as MetaMask.

4.3 Development tools

4.3.1 System configuration and operating system

The project are performed on processor Intel(R) Celeron(R) CPU B840 1.90GHz 1.90 GHz with RAM 4.00 GB of memory and System type 64-bit operating system, x64-based processor We implement the project using windows 10.

4.3.2 Solidity

Solidity is an object-oriented, high-level language for creating smart contract programs that can be executed by the EVM. Solidity is based on the JavaScript, C, and Python programming languages. It enables the creation of contracts and their compilation into EVM bytecode. It is Ethereum's official language at the moment. It's the most widely used language library for the EVM, and it supports inheritance and libraries. The version of the Solidity compiler should be specified in each contract/library source file. The compiler version should be specified first and foremost in the source file [17]. There are several integration platforms for compiling, executing, and running Solidity code, the most well-known of which is Remix.

Figure 4.2: Solidity logo

4.3.3 Remix IDE

Remix IDE, an open source web and desktop application, was used to implement and test our smart contract. It promotes a quick development cycle and includes a large number of plugins with user-friendly interfaces. Remix is used throughout the contract development and learning process. Remix IDE is an Ethereum-based development tool that is part of the Remix Project, which is a platform for plugin-based development tools. It includes sub-projects such as the Remix Plugin Engine and Remix Libs.

Remix IDE is a powerful open source tool that allows to write Solidity contracts right in browser. It also includes modules for testing, debugging, and deploying smart contracts. It's written in JavaScript and aids each use within the browser, but it can also be run locally and on a desktop [9]. Remix IDE is available at http://remix.ethereum.org/

Figure 4.3: Remix IDE logo

4.3.4 HTML and CSS and JavaScript

- **HTML** (Hypertext Markup Language) is the internet's most basic building block. It establishes the meaning and shape of web content. Aside from HTML, a variety of technologies are commonly used to describe the appearance and presentation of an internet page. CSS or functionality JavaScript [21].
- **CSS** (Cascading Style Sheets) is a stylesheet language for describing the presentation of an HTML or XML document. CSS specifies how elements should be displayed on a screen, on paper, in speech, or in other media [22].
- **JavaScript** is a client-side and server-side dynamic programming language that is lightweight and most commonly has object-oriented capabilities.

  JavaScript is a scripting language used by many browsers. By converting a static web page into an interactive one, incorporating JavaScript improves the user's experience with the web page. JavaScript is a scripting language that adds structure to web pages and is used in conjunction with HTML and CSS. When it comes to formatting HTML elements, the scripting language works well with CSS.

Figure 4.4: HTML CSS JavaScript logos

4.3.5 Sublime text

The Sublime Text Editor is a cross-platform Integrated Development Editor (IDE) that works with Windows, Linux, and MacOS. It's similar to Visual Studio Code and NetBeans. We decided to implement the project because :

- Light and easy to use.
- Ability to resolve connector errors.
- Track all files and folders to work.
- Ability to solve a problem.
- Save the color combination for the syntax combination.

Figure 4.5: Sublime text logo

4.3.6 NPM

NPM (Node Package Deal Manager) is the world's largest software repository. There are over 800,000 code programs in the registry. It's a lot more open-source and free to use. There is no need to register or log in to download any of the public npm software programs. To install npm on your computer, you must first install node.js. All npm packages are defined in documents called package.json that are written in the JSON language, and npm can install all task dependencies defined in package.json with a single command line. The name (Node Package Manager) comes from the fact that npm was initially developed as a package manager for node.js [53].

Figure 4.6: NPM logo

4.3.7 Node js

Node.Js is a server-side, cross-platform open-source framework for creating highly scalable and fast packages. Node.Js is a platform that is built on v8, the JavaScript runtime that powers Google's Chrome browser, and it employs an event-driven, non-blocking I/O architecture. Node.js has the ability to serve functions in a synchronous manner. Node.js is written in JavaScript and works on a variety of platforms (Windows, Linux, Mac, etc.) [27].

It enables developers to create command-line tools and server-side scripting to generate dynamic web page content before sending the page to the user's browser. To develop smart contracts.

Figure 4.7: Node.js logo

4.3.8 Truffle

The truffle suite is a Web3 development ecosystem built on the Ethereum Blockchain that is divided into three parts (Truffle, Ganache and Drizzle). The truffle suite is used to develop and test smart contracts [39].

Truffle Suite is a "world-class development environment, testing framework, and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier," according to Truffle Suite. Truffle is a popular Ethereum DApp development framework that uses EVM and focuses on smart contract development and front-end development for DApps. One of the benefits of truffle is that it can be used to compile and test a smart contact.

Figure 4.8: Truffle logo

4.3.9 Ganache

Ganache, according to Truffle Suite, is "A personal Ethereum Blockchain that can be used to deploy contracts, develop applications, and run tests. It comes in two flavors: a desktop application and a command-line tool (formerly known as the TestRPC). Ganache is a cross-platform application that runs on Windows, Mac OS X, and Linux."

Ganache It's an Ethereum Blockchain Emulator that's been installed locally.

Ganache comes with a graphical user interface for simulating Blockchain networks. Ganache gives us ten addresses, each of which contains 100 eths. This is fictitious ether with no real value, which aids us in uploading smart contracts to the Ethereum Blockchain and paying gas fees. In addition, for each smart contract we want to test, we must pay a transaction fee.

MetaMask can also be added to the Ganache network by using the Ganache RPC server and supplying the wallet with fake ether using the address's private address key.

Figure 4.9: Ganache logo

4.3.10 MetaMask

MetaMask is a Blockchain-specific wallet for Ethereum. A wallet is a personal key that allows you to interact with the crypto world by buying, selling, and transferring assets on the Blockchain.

MetaMask is an app for iOS and Android that allows you to buy, store, send, and swap tokens. It can also be used as an extension for some web browsers, including Chrome, Firefox, Brave, and Edge. On a device, MetaMask generates passwords and keys so that only the user has access to their accounts and data.

Figure 4.10: MetaMask logo

4.3.11 React

React, also known as React.js, is an open-source JavaScript front-end library for creating user interfaces. It uses React to build single-page applications and allows to create complex user interfaces (UI) from small, isolated pieces of code called "components." It makes use of a virtual DOM (JavaScript object) to improve the app's overall performance, and we can use React on both the client and server sides. It was created and maintained by Facebook, and it was later used in Facebook products such as WhatsApp and Instagram.

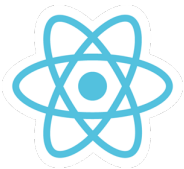

Figure 4.11: React logo

4.4 System description

The goal of this system is to design and implement a Blockchain-based web application to ensure that financial donations reach the poor in need with reliable transparency.

4.5 Environment configuration

After installing node.js and Ganache we will open a a terminal and install the Truffle Framework, which allows us to build decentralized applications on the Ethereum blockchain. It provides a set of tools that allow us to write intelligent contacts using the Solidity programming language. It also enables us to test and publish our smart contract in the Blockchain. It also gives us a place to develop our application from the client side.

We can install Truffle using NPM on the command line as follows:

```
$ npm install -g truffle
```

We will create a project directory for our dApp and React App in the command line :

```
$ npx create-react-app project
$ cd project
```

We start our new, empty Ethereum project from the command line as follows:

```
$ truffle init
```

We get the files shown in the figure 4.12

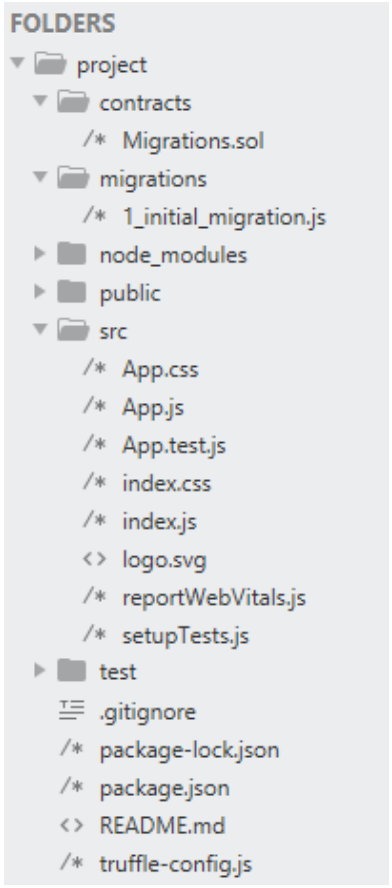

Figure 4.12: Project files

- contracts directory: All of our Smart Contacts can be found in this folder. Our migrations to the Blockchain are already managed by a Migration contract.
- migration Directory: All of our migration files are stored here. These migrations are similar to those required by other web development frameworks when changing the state of a database. We must migrate when we deploy smart contracts on the Blockchain because we are updating the state of the Blockchain.
- node_modules directory: All of our Node dependencies are stored here.
- public directory: This is where the HTML file is stored.
- src directory: This is where our client-side app will be developed.
- test directory: We'll write our smart contract tests in this folder.
- package-lock.json: It's the one that keeps track of the exact version of every package installed so that a product can be reproduced exactly the same way every time, even if the packages' maintainers update them.

- package.json This is where the project's viewable information (such as the project name and description) and functional metadata (such as the package version number and list of dependencies that the application requires) are stored.
- truffle-config.js : Our Truffle project's main configuration file.

To create our decentralized donation app, we first created an Ethereum smart contract, named DonationContract.sol in the '/src/contracts' directory. We need to select version 0.5.17 of Solidity compiler and work on the 127.0.0.1 localhost network and port 7545 to match the Ganache settings and we put * in network_id to match any network id , through the truffle-config file as follows:

```
module.exports = {
  networks: {
     development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*"
     }
  },
  contracts_directory :'./src/contracts',
  contracts_build_directory :'./src/build',
  compilers: {
    solc: {
      version: "0.5.17"
    }
  },
};
```

Listing 4.1: truffle-config.js

4.6 Writing smart contract

First, in our DonationContract smart contract we have 5 variables to count the number of the poor, the number of the poor accepted, the number of donors , the number of donors accepted, number of transactions and 5 mapping for each mapping have unique keys In our case. We used an id as a key in a struct and a set of variables that define both the poor and the donor, so we have 5 struct which is poor, poor accepted, donor, donor accepted and histore.

- **mapping poors** store all the information about the poor requests.
- **mapping poorsAcc** store all accepted poor account information.

- **mapping donors** store all the information about the donor requests.
- **mapping donorAcc** store all accepted donor account information.
- **mapping histore** store donation transactions that occur in our application.

```
pragma solidity ^0.5.17;

contract DonationContract {

    uint public poorsRequests = 0;
    uint public poorsRequestsacc = 0;
    uint public donorsNumber = 0;
    uint public donorsAccNumber = 0;
    uint public transactionNumber = 0;
    mapping(uint => poor) public poors;
    mapping(uint => poorAcc) public poorsAcc;
    mapping(uint => donor) public donors;
    mapping(uint => donorAcc) public donorsAcc;
    mapping(uint => histore) public histores;

    struct poor {uint id; string name; string password; uint
       monthlyIncome; bool marriage; uint NumberChildren; uint age;
       address payable owner; string situation;string livingAddress ;
        bool donated; }

    struct poorAcc {uint id; string name; string password; uint
       monthlyIncome; bool marriage; uint NumberChildren; uint age;
       address payable owner; string situation;string livingAddress ;
        bool donated; }

    struct donor {uint id; string name; string password; string sex; uint
         age; string livingAddress; address payable owner;}

    struct donorAcc {uint id; string name; string password; string sex;
            uint age; string livingAddress; address payable owner; }

    struct histore{uint id; address payable poorOwner; uint price;
       address payable donorOwner; }
```

- **createPoorAccount() function**

  The createPoorAccount() function receives the information of the poor. As for the while loop, it is to make sure that he doesn't create an already existing account. He is also required to type his full name and password that contain

more than 0 characters. This function stores the poor's information on Mapping poors with the associated owner's address and puts false on the donated variable. Meaning that it was not donated to him yet and in the last trigger a PoorRequestCreated event to communicate with our client application (front-end) as shown in the code:

```
event acceptPoorAccount(uint id,string name,string password,uint
    monthlyIncome,bool marriage,uint NumberChildren,uint age,address
    payable owner,string situation,string livingAddress,bool donated)
    ;

function createPoorAccount(string memory _name,string memory
    _password,uint monthlyIncome,bool _marriage,uint NumberChildren,
    uint age,string memory situation ,string memory livingAddress)
    public {
        uint8 i = 1;
        while ( i <= poorsRequests) {
            poor memory _poor = poors[i];
            address payable _poorReq = _poor.owner;
            require(_poorReq != msg.sender);
            i++;
        }
        // Require a valid name
        require(bytes(_name).length > 0);
        // Require a valid password
        require(bytes(_password).length > 0);
        // Increment poor count
        poorsRequests ++;
        // Create the poor
        poors[poorsRequests] = poor(poorsRequests, _name,_password,
            monthlyIncome,_marriage,NumberChildren,age,msg.sender,
            situation,livingAddress,false);
        // Trigger an event
        emit PoorRequestCreatedacc(poorsRequests, _name,_password,
            monthlyIncome,_marriage,NumberChildren,age,msg.sender,
            situation,livingAddress,false);
}
```

- **acceptPoorAccount() function**

  The function acceptPoorAccount() receives the variable _id is a key that specifies the poor to accept, all the poor information in the cell specified by the index _id is copied from mapping poors and pasted into the mapping poorsAccas and in the last trigger an event PoorRequestCreated and this function is for the admin shown in the code:

```
function acceptPoorAccount(uint _id) public  {
    poor memory _poor = poors[_id];
    string memory _name = _poor.name;
    string memory _password = _poor.password;
    uint monthlyIncome = _poor.monthlyIncome;
    bool marriage = _poor.marriage;
    uint NumberChildren = _poor.NumberChildren;
    uint age = _poor.age;
    address payable owner = _poor.owner;
    string memory situation= _poor.situation;
    string memory livingAddress= _poor.livingAddress;
    poorsRequestsacc ++;
    poorsAcc[poorsRequestsacc] =poorAcc(poorsRequestsacc,_name,
        _password,monthlyIncome,marriageNumberChildren,age,owner,
        situation,livingAddress,false);
    // Trigger an event
    emit PoorRequestCreated(poorsRequestsacc,_name,_password,
        monthlyIncome,marriage,NumberChildren,age,owner,situation,
        livingAddress,false);
}
```

- **createDonorAccount() function**

  The createDonorAccount() function receives all the information of the donor. As for the while loop, it is to make sure that he doesn't create an already existing account. He is also required to type his full name and password that contain more than 0 characters. This function stores the donor information on Mapping donors with the associated owner's address , and adding one to the donorsNumber variable, which is a counter for the number of donors who requested to create an account. In the last trigger an event donorCreated as shown in the code:

```
function createDonorAccount(string memory _name,string memory
   _password,bool _gender,uint _age,string memory _livingAddress)
   public {
        uint8 i = 1;
        while ( i <= donorsNumber) {
            donor memory _donor = donors[i];
            address payable _donorAdr = _donor.owner;
            require(_donorAdr != msg.sender);
            i++;
        }
        require(bytes(_name).length > 0);
        require(bytes(_password).length > 0);
        donorsNumber ++;
```

```
        donors[donorsNumber] = donor(donorsNumber, _name,_password,
            _gender,_age,_livingAddress,msg.sender);

        emit donorCreated(donorsNumber, _name,_password,_gender,_age
            ,_livingAddress,msg.sender);

}
```

- **acceptDonorAccount() function**

  This function works the same as the acceptPoorAccount() function but it is copied from mapping donors and pasted into mapping donorsAcc and triggered an event donorAccCreated. This function is also for the admin.

- **donatePoor() function** The donatePoor function receives a single variable that specifies the value the donor will pay to a the poor. First, we make sure that the person who makes a donation is in the list (mapping) donorsAcc

```
function donatePoor(uint _price) public payable {
    uint256 max = 0;
    uint _id;
    uint i = 1;uint j=1;
    bool donorAccbool=false;
    while(j <= donorsAccNumber){
        donorAcc memory _donor = donorsAcc[j];
        if(msg.sender == _donor.owner && _donor.id != 0){
            donorAccbool = true;}
        j++;
    }
```

This loop is repeated with the same number of acceptable poor calculations as specified by the poorsRequestsAcc variable. We calculate an approximate rate using whether he is married or not, the number of his children, his age and his monthly income And we calculate an approximate rate, and the poor person who gets the largest result, we keep his id.

```
    while ( i <= poorsRequestsacc && (donorAccbool == true)) {
    poorAcc memory _poor = poorsAcc[i];
    bool _marriage= _poor.marriage;
    uint _NumberChildren = _poor.NumberChildren;
    uint _age = _poor.age;
    uint _monthlyIncome = _poor.monthlyIncome;
        if(!_poor.donated && _poor.id != 0){
                if(_marriage == true  && _NumberChildren == 0){
                    uint256 moy = (_age / 5) + 3 + (300 / (
                        _monthlyIncome + 4));
```

```
                    if(moy>max){
                        max = moy;
                        _id = i;
                    }
                }
                else if(_marriage == false  && _NumberChildren == 0)
                    {
                    uint256 moy = (_age / 5) + (90 / (_monthlyIncome
                        + 4));
                    if(moy>max){
                        max = moy;
                        _id = i;
                    }
                }
                else{
                    uint256 moy = (_age / 5) + (_NumberChildren * 4)
                        + (300 / (_monthlyIncome + 4));
                    if(moy>max){
                        max = moy;
                        _id = i;
                    }
                }
        }
        i++;
    }
```

We call the poor from mapping poorAcc by the id we we got, we keep the owner address in the variable poorAddress, make sure he has a valid id and the value he paid is bigger than 0. We also make sure that he has not already been donated and that he is not the poor. Then we send the money to the poor and change the donated variable to true, It means he was donated to him and store this transaction in mapping histores and in the last Trigger an event Poorpaid.

```
    poorAcc memory _poor = poorsAcc[_id];
    // Fetch the owner
    address payable poorAddress = _poor.owner;
    // Make sure the poor has a valid id
    require(_poor.id > 0 && _poor.id <= poorsRequestsacc);
    // Require that there is enough Ether in the transaction
    require(_price > 0);
    // Require that the poor has not been donated already
    require(!_poor.donated);
    // Require that the donor is not the poor
    require(poorAddress != msg.sender);
    // Mark as donated
```

```
    _poor.donated = true;
    // count transaction
    transactionNumber++;
    // Update the poor
    poorsAcc[_id] = _poor;
    // Pay the poors by sending them Ether
    address(poorAddress).transfer(msg.value);
    // storage transaction
    histores[transactionNumber]=histore(transactionNumber,
       poorAddress, _price, msg.sender);
    // Trigger an event
    emit Poorpaid(transactionNumber, poorAddress, _price,
    msg.sender);
}
```

- **donatePoorByChoice() function**

  The donatePoorByChoice() function receives two variables, one that specifies the amount the donor will pay to the poor, and a special id that specifies the poor. The rest of the steps are the same in the donatePoor() function, but without the need to calculate the rate because we already have the poor id.

- **modifyPoor() function**

  modifyPoor() function receives all variables that containing new poor information with the condition that the number of characters for the variable name and password is bigger than 0 And store all of the new information in place of the old information in mapping poorsAcc as shown in the code:

```
function modifyPoor(uint id , string memory _name,string memory
   _password ,uint monthlyIncome,bool _marriage ,uint
   _NumberChildren ,uint _age,string memory situation ,string memory
    livingAddress)  public {
        require(bytes(_name).length > 0);
        require(bytes(_password).length > 0);

        poorsAcc[id] = poorAcc(id,_name,_password,monthlyIncome,
           _marriage,_NumberChildren,_age,msg.sender,situation,
           livingAddress, false);

        emit PoorRequestCreated(id,_name,_password,monthlyIncome,
           _marriage,_NumberChildren,_age,msg.sender,situation,
           livingAddress, false);
}
```

- **delete functions**

All delete functions receive the id variable that specifies the person to be deleted whether it is in mapping poors or poorsAcc or donors or donorsAcc as shown in the code:

```
function deletePoor(uint id) public {
    delete poors[id];
}
function deleteDonor(uint id) public {
    delete donors[id];
}
function deletePoorAccount(uint id) public {
    delete poorsAcc[id];
}
function deleteDonorAccount(uint id) public {
    delete donorsAcc[id];
}
```

4.7 Compiling the smart contract

After writing the Smart Contract, we need to compile the code to check for errors. using this command in the console of the project directory:

$ truffle compile

After compiling, we get the following result:

```
 Compiling your contracts...
===========================
> Compiling .\src\contracts\DonationContract.sol
> Compiling .\src\contracts\Migrations.sol
> Artifacts written to C:\Users\Dayou\Desktop\project\src\build
> Compiled successfully using:
   - solc: 0.5.17+commit.d19bba13.Emscripten.clang
```

We notice that a new file has been created (./src/build/DonationContract.json"). This file is the smart contract's ABI (Abstract Binary Interface) file. It contains the compiled bytecode version of the Solidity smart contract code that can be run on a the Ethereum Virtual Machine (EVM), i.e., an Ethereum Node. and contains a JSON representation of the smart contract functions that can be exposed to external clients, like client-side JavaScript applications [30].

4.8 Deploying the smart contract

Write a migrating for deploy our smart contract on the development network (local Ethereum Blockchain)

We create a new file in migration directory named "2_deploy_contracts.js", We numbered files inside the migrations directory so that Truffle knows which order to execute them in.

```
const DonationContract = artifacts.require("DonationContract");

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

Now we run our migrations from the command line:

$ truffle migrate

We have successfully migrated our smart contract to the local Ethereum blockchain as shown in the console:

```
Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0
      xfdec69c24956a6a63a1df9ac903b05c613eafffdb19f3b21c3315b1262965c79
- Blocks: 0            Seconds: 0
   > Blocks: 0            Seconds: 0
   > contract address:    0x838684A9804856E3F831c3e1eca628A829Bee138
   > block number:        1
   > block timestamp:     1655387160
   > account:             0x6fb9C61641d41a7801de802D898CeAD1267e8e01
   > balance:             4.99616114
   > gas used:            191943 (0x2edc7)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00383886 ETH
```

```
- Saving migration to chain.
   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00383886 ETH


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

   Replacing 'DonationContract'
   ----------------------------
   > transaction hash:    0
      x3b76a4f66f9551d7da5bdf9ec832b4779063024fc7fa7afedf9d53645c519775
- Blocks: 0            Seconds: 0
   > Blocks: 0            Seconds: 0
   > contract address:    0xb9E3e1620FD442d6c66776575ed314d0F0863d67
   > block number:        3
   > block timestamp:     1655387163
   > account:             0x6fb9C61641d41a7801de802D898CeAD1267e8e01
   > balance:             4.88994526
   > gas used:            5268456 (0x5063e8)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.10536912 ETH

- Saving migration to chain.
   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.10536912 ETH

Summary
=======
> Total deployments:   2
> Final cost:          0.10920798 ETH
```

After deploying our smart contract we notice that the ganache which is our local Ethereum blockchain has generated blocks as shown in the figure 4.13:

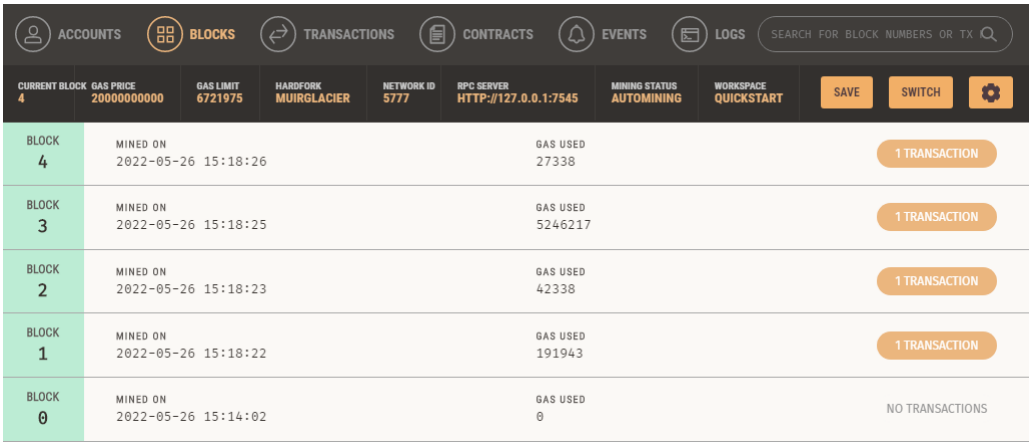

Figure 4.13: Ganache blocks

4.9 Testing smart contract

To test our smart contract, we created a DonationContract.test.js file that uses JavaScript to simulate client-side interaction. Figure 4.14 depicts the outcomes of some of the tests we devised. We run the tests from this command line:

```
$ truffle test
```

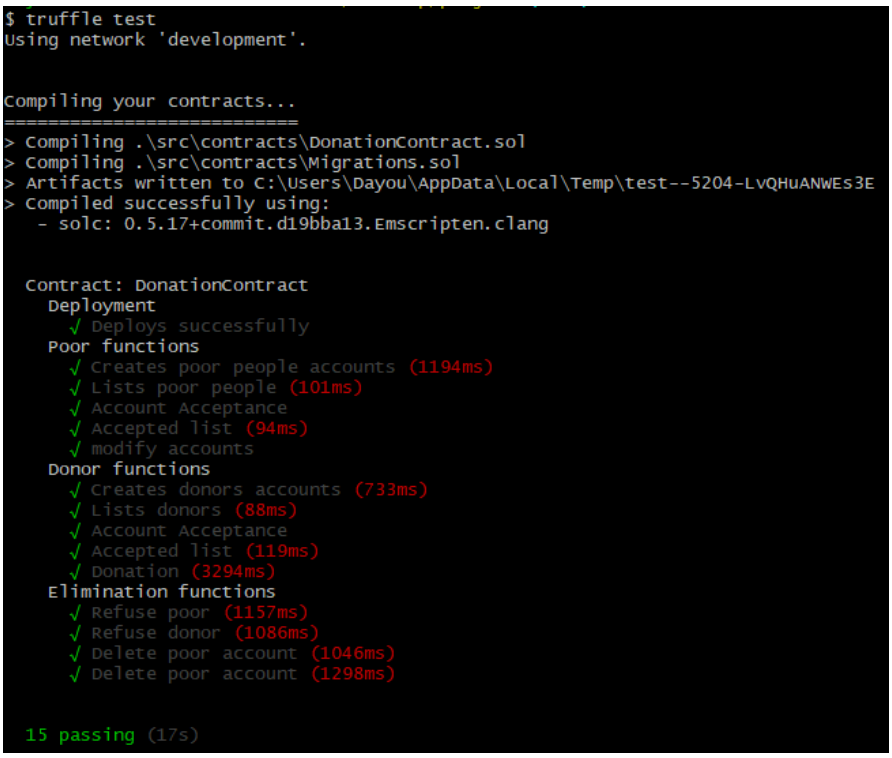

Figure 4.14: Testing smart contract

4.10 Compiling and deploying in remix

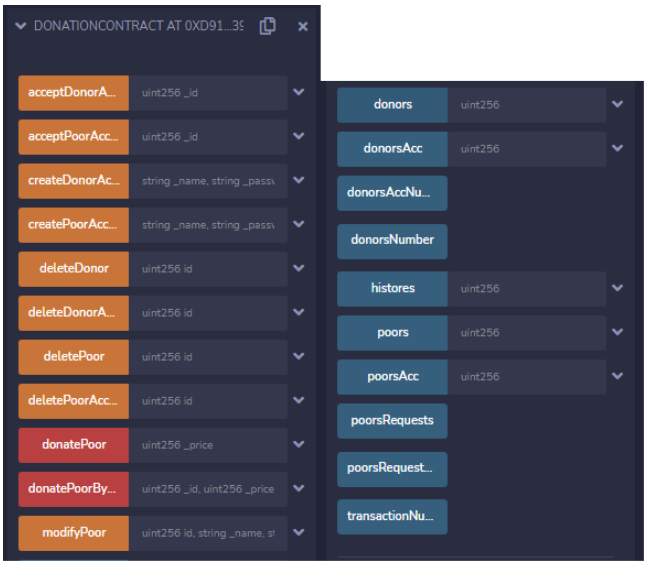

Figure 4.15: Smart contract in remix

- Constant or pure functions are represented by blue buttons. It does not initiate a new transaction when you click it. Therefore clicking will not affect the status - it will just return the value saved in the contract file, so it will not cost anything in gas fees [9].
- Non-payable functions are those that affect the status of the contract but do not receive Ether and feature an orange button. By clicking on it, you will initiate a transaction and so increase the cost of gas [9].
- Payable functions are those with red buttons. By clicking on it, a new transaction is created, and this transaction can accept a value [9].

4.11 Front End Client-side

In this section we will see linking the smart contract (Blockchain) to the client application (react js) by importing web3, And we add connectivity to DonationContract. We import the ABI smart contract. Into the main App.js component like this:

```
import Web3 from "web3";
import DonationContract from "./build/DonationContract.json";
```

The function below detects the presence of an Ethereum provider in a web browser, allowing our app to be connected to the Blockchain.

```
async loadWeb3() {
    if (window.ethereum) {
      window.web3 = new Web3(window.ethereum);
      await window.ethereum.enable();
    } else if (window.web3) {
      window.web3 = new Web3(window.web3.currentProvider);
    } else {
      window.alert(
        "Non-Ethereum browser detected. You should consider trying
           MetaMask!"
      );
    }
  }
```

The second function, loadBlockchainData(), loads Blockchain data. The following commands are defined in this function:

```
const web3 = window.web3;
const accounts = await web3.eth.getAccounts();
const Balance = await web3.eth.getBalance(accounts[0]);
const BalanceEth = await web3.utils.fromWei(Balance, "ether");
const networkId = await web3.eth.net.getId();
const networkData = DonationContract.networks[networkId];
```

The first line is to connect the web3 connection to a variable.

The second line is we fetch the accounts from Metamask and record them in the console. The third and fourth lines bring the accounts balance.

The fifth and sixth lines read the "networkID" to determine which Metamask network, i.e. Ganache, we're connected to. This network ID will be used to connect to the smart contract on the Ganache network. Web3.js and web3.eth.Contract() are then used to create the smart contract . We'll need two pieces of information to do so: the smart contract's ABI and the address. Both of them are taken from the imported file.

To fetch all the poor of the blockchain inside the loadBlockchainData() function in the process below, The same process applies to all mapping existing in the smart contract.

```
for (var i = 1; i <= poorsRequests; i++) {
  const poor = await donationContract.methods.poors(i).call();
  this.setState({
    poors: [...this.state.poors, poor],
  });
}
```

In the code below, we call one of the smart contract functions, which is the donatePoor() function, which receives the price donated by the donor. In the same

way all smart contract functions are called inside react code.

```
this.state.donationContract.methods.donatePoor(price).send({ from: this.
    state.account , value: price})
```

4.12 Presentation of the system interfaces

We open the app, connect our web browser and web app to the Blockchain, and begin communicating with the DonationContract smart contract. Before we begin, we will start our development server with the following command:

$ npm start

The figure 4.16 represents the home page

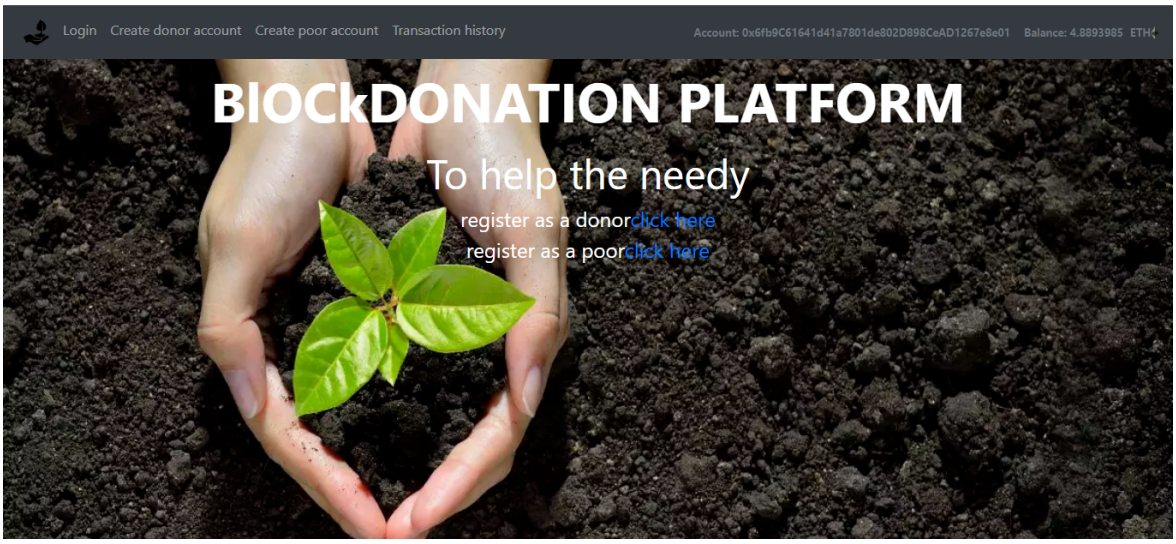

Figure 4.16: Home page

The figure 4.17 depicts three distinct pages: the login page, the poor account creation page, and the donor account creation page.

When press to create an account for the poor or the donor, the request is sent to the admin, and the admin is the one who decides whether to add or refuse. As for the authentication in login, it is by entering the correct full name and a correct password, and it must be connected to the correct address that he registered with through the wallet.

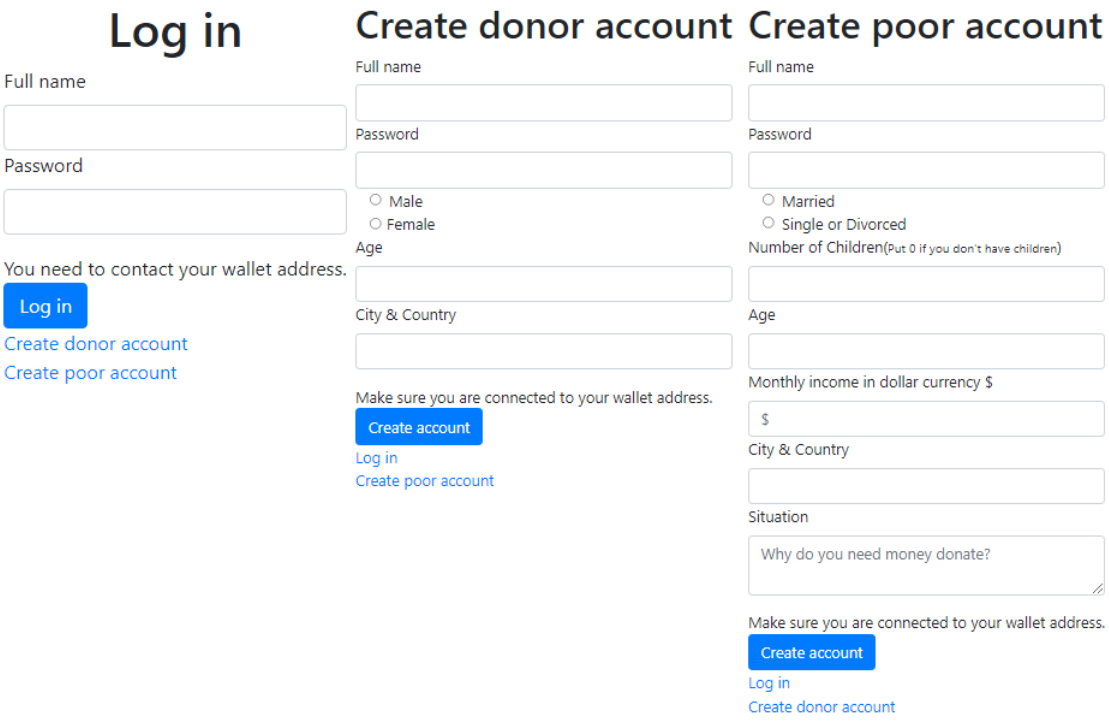

Figure 4.17: Login and create accounts pages

Admin pages:

The figure 4.18 shows the admin pages where the admin can accept or refuse the accounts, and when they are accepted, he also can delete them.

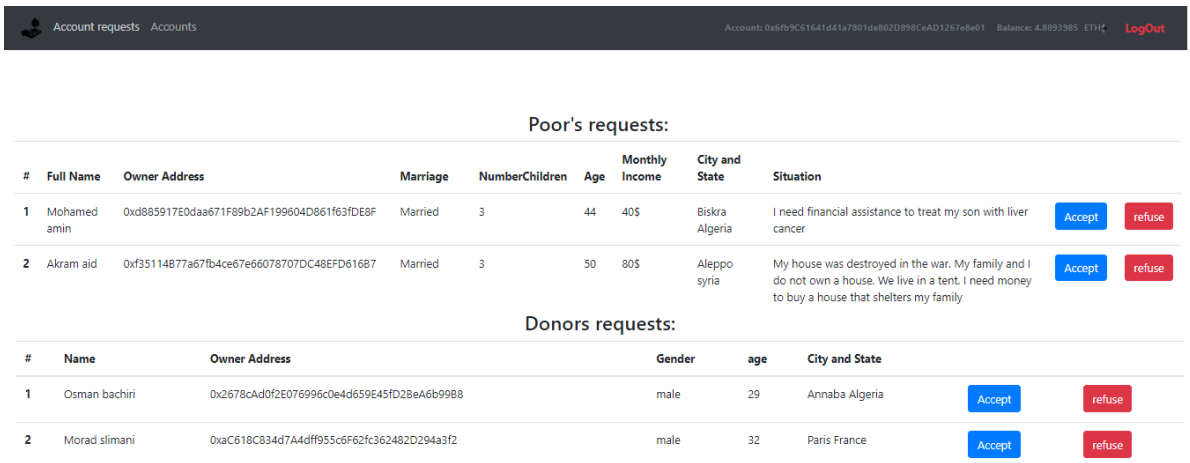

Figure 4.18: Accounts acceptance page

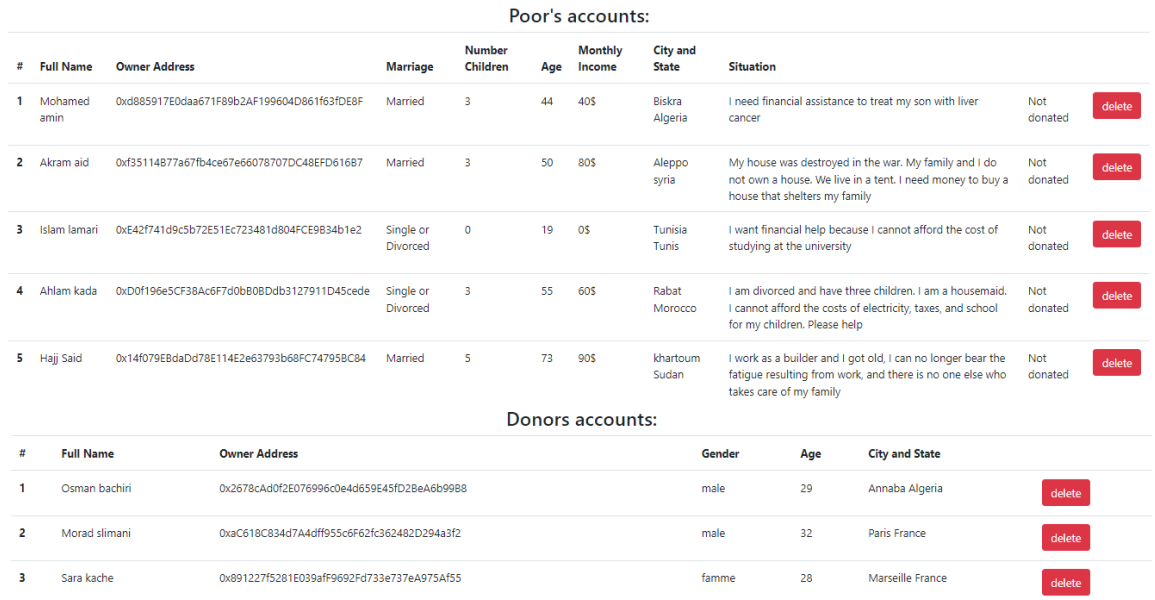

Figure 4.19: Accounts delete page

Poor pages:

The figure 4.20 shows the pages of the poor, where the poor can modify or delete his account If he has benefited from a donation, then if he modifies his account, he can benefit again.

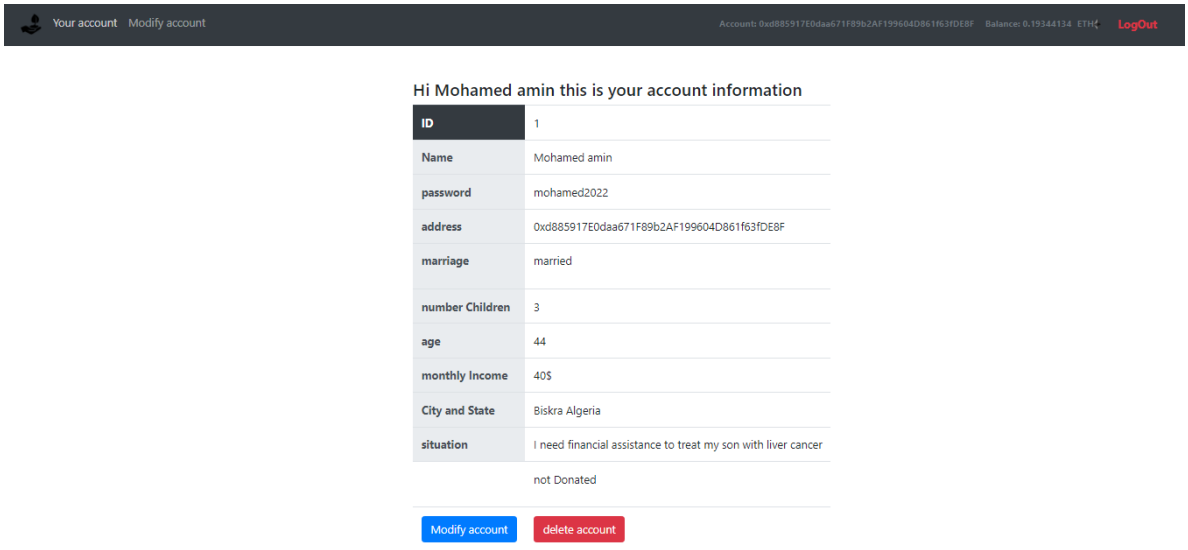

Figure 4.20: Poor account page

Donor pages:

The figure4.21 shows the donor account page, he can delete his account and the figure 4.22 shows the donation page. The donor can put the amount and press the

donate button and the figure 4.23 shows the donation page by choice, where the donor can choose the poor person to donate to and indicate the amount in a box for the poor and press donate. All operations donate this money using the MetaMask wallet.

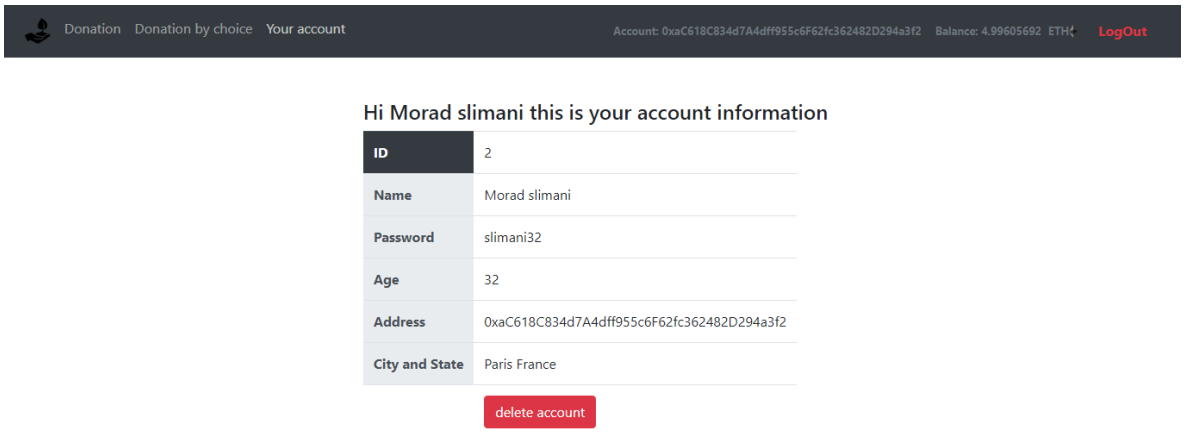

Figure 4.21: Donor account page

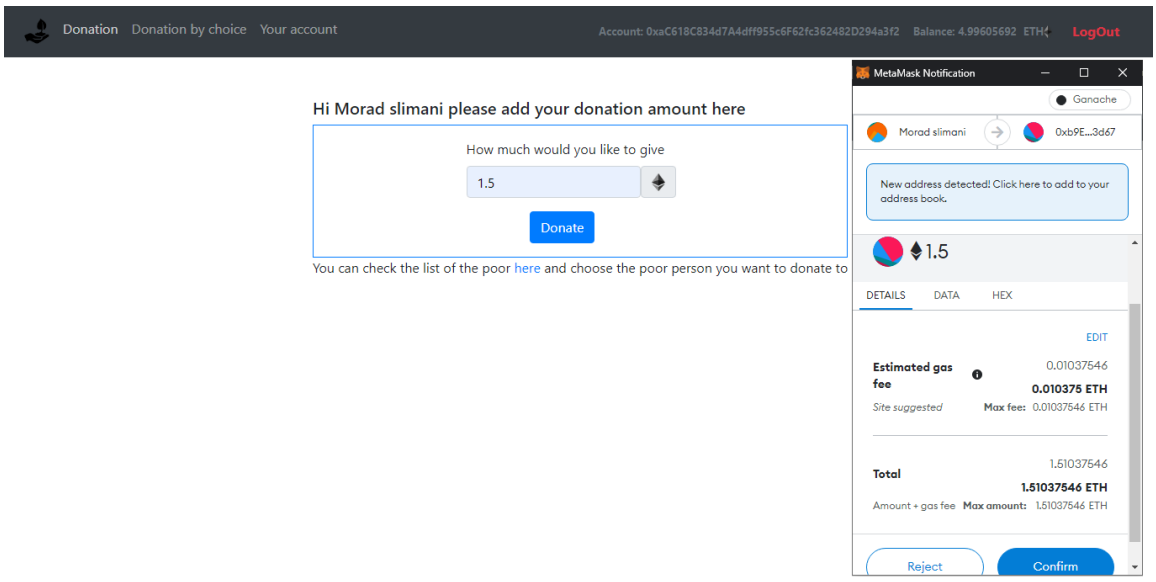

Figure 4.22: Donation page

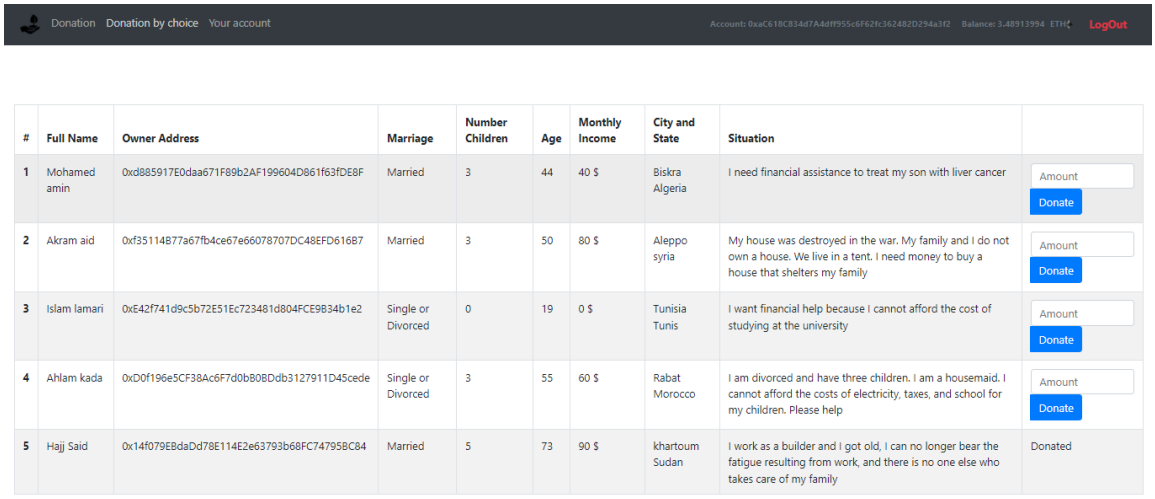

Figure 4.23: Donation by choice page

Transaction history page The figure 4.24 shows the transaction history page, where it shows all the donations that were made in the application, and everyone can access this page.

The page displays the donor's address, the poor's address, and the amount donated.

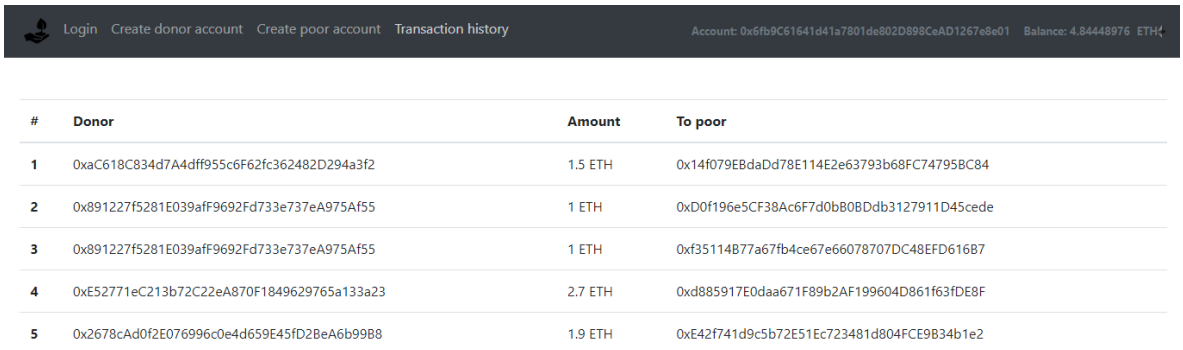

Figure 4.24: Transaction history page

4.13 Conclusion

In this chapter, a donation application was created that achieved transparency using the Blockchain without the need for a central entity by implementing a model using Ethereum and smart contracts, where the donor could easily track funds and know the details of each donation. The poor beneficiary could receive the donation directly through cryptocurrency without the need for centralized management, proving that the Blockchain can be used to make secure financial payments.

General Conclusion

In several domains, Blockchain technology has demonstrated its efficacy in terms of security and decentralization, making dependence on centralized old systems unattractive because there is no guarantee that data would not be altered with in traditional centralized systems.

This is a project that sought to find a solution to the problem of managing financial donations, in which donors have lost faith in charities due to a lack of transparency. A solution has been introduced using Blockchain technology that is characterized by transparency. This latter ensures that donors have confidence in how the donated money is spent, and the ability to trace where the donor can easily track the money and know the details of each donation, and the beneficiary can receive the donation directly without the need for central management. These two characteristics are the most prominent characteristics of this technology which has a lot of advantages. This project's strengths include the fact that the information will not be falsifiable as long as the Blockchain exists, as well as the fact that the Blockchain is a large distributed record that is replicated in multiple locations, making it impossible to tamper with the data and all the information recorded in the system.

Bibliography

[1] Ethereum. *coinmarketcap*. https://coinmarketcap.com/currencies/ethereum/.

[2] Ethereum's proof of stake protocol under review. https://cryptoslate.com/ethereums-proof-of-stake-protocol-in-review/.

[3] History of blockchain. icaew.com.

[4] What is cause-related marketing? https://learning.candid.org/resources/knowledge-base/cause-related-marketing/.

[5] Blockchain against hunger: Harnessing technology in support of syrian refugees. *World Food Programme*, 2017.

[6] Definition - what is a donation? https://www.finance.admin.cam.ac.uk/policy-and-procedures/financial-procedures/chapter-14-accounting-donations-and-grants/scope-2.

[7] The difference between blockchain and bitcoin. *UK Tech News*, 2020. https://www.uktech.news/the-difference-between-blockchain-and-bitcoin.

[8] Pursuing innovative approaches for access to personal identity. *World Identity Network*. https://win.systems/about-win/.

[9] Aniket. Remix's documentation! https://remix-ide.readthedocs.io/en/latest/.

[10] Andreas M Antonopoulos. The blockchain. mastering bitcoin, 2014.

[11] Michael J Barone, Anthony D Miyazaki, and Kimberly A Taylor. The influence of cause-related marketing on consumer choice: does one good turn deserve another? *Journal of the academy of marketing Science*, 28(2):248–262, 2000.

[12] Vangie Beal. Public-key encryption. *URL http://www. webopedia. com*, 2021.

[13] Vangie Beal. Symmetric-key cryptography. *URL http://www. webopedia. com*, 2021.

[14] Kim-Kwang Raymond Choo. New payment methods: A review of 2010–2012 fatf mutual evaluation reports. *Computers & Security*, 36:12–26, 2013.

[15] Parizo Christine. *What are the 4 different types of blockchain technology?* 2021.

[16] Kevin Curran. E-voting on the blockchain. *The Journal of the British Blockchain Association*, 1(2):4451, 2018.

[17] Chris Dannen. *Introducing Ethereum and solidity*, volume 1. Springer, 2017.

[18] HAMILTON DAVID. What is a blockchain transaction anyway. DECEMBER 2018. `https://coincentral.com/what-is-a-blockchain-transaction-anyway/`.

[19] Dwane Hal Dean. Consumer perception of corporate donations effects of company reputation for social responsibility and type of donation. *Journal of advertising*, 32(4):91–102, 2003.

[20] Christian Delgado-von Eitzen, Luis Anido-Rifón, and Manuel J Fernández-Iglesias. Blockchain applications in education: A systematic literature review. *Applied Sciences*, 11(24):11811, 2021.

[21] MDN Web Docs. Html (hypertext markup language). *Recuperado desde: https://developer. mozilla. org/es/docs/Web/HTML*, 2017.

[22] MDN Web Docs. Css: Cascading style sheets.[online] available at: https://developer. mozilla. org/en-us/docs/web. *CSS [Accessed 28 Nov. 2019]*, 2019.

[23] Muhammad Shoaib Farooq, Misbah Khan, and Adnan Abid. A framework to make charity collection transparent and auditable using blockchain technology. *Computers & Electrical Engineering*, 83:106588, 2020.

[24] J Frankenfield. Decentralized applications—dapps. investopedia, 2021.

[25] Jake Frankenfield. Cryptocurrency. *Accessed from Investopedia Website: https://www. investopedia. com/terms/c/cryptocurrency. asp*, 2019.

[26] Daniel G and Green Amanda. Ifrs (#) accounting for crypto-assets. 2018.

[27] Cory Gackenheimer. Understanding node. js. In *Node. js Recipes*, pages 1–26. Springer, 2013.

[28] DBA Geraldo Vasquez. An introduction to blockchain what does it mean for the accounting profession? 2021. https://www.cpajournal.com/2021/08/18/an-introduction-to-blockchain/.

[29] Alexander Grech and Anthony F Camilleri. *Blockchain in education*. Luxembourg: Publications Office of the European Union, 2017.

[30] McCubbin Gregory. How to build blockchain app - ethereum todo list 2019. https://www.dappuniversity.com/articles/blockchain-app-tutorial.

[31] Manav Gupta. Blockchain-ibm limited edition, 2017.

[32] Iredale Gwyneth. What are the different types of blockchain technology. January 2021. https://101blockchains.com/types-of-blockchain/.

[33] Campbell R Harvey. Cryptofinance. *Available at SSRN 2438299*, 2016.

[34] Robby Houben and Alexander Snyers. *Cryptocurrencies and blockchain: Legal context and implications for financial crime, money laundering and tax evasion*. 2018.

[35] Williams Hunter. Do you know how to accept donations online? 2014. https://www.littlegreenlight.com/blog/donation-processing-101/.

[36] DacNhuong Le, Raghvendra Kumar, Brojo Kishore Mishra, Manju Khari, and Jyotir Moy Chatterjee. Cyber security in parallel and distributed computing.

[37] Pratap Mayank. Blockchain technology explained: Introduction, meaning, and applications. *hackernoon.com*, 2018. https://tinyurl.com/ddwdjhx4.

[38] Bhabendu Kumar Mohanta, Debasish Jena, Soumyashree S Panda, and Srichandan Sobhanayak. Blockchain technology: A survey on applications and security privacy challenges. *Internet of Things*, 8:100107, 2019.

[39] Moralis. *Truffle Explained*. 2021.

[40] Harish Natarajan, Solvej Krause, and Helen Gradstein. Distributed ledger technology and blockchain. 2017.

[41] Guillermo Nicolas. How we can stop all voter fraud. 2016. https://followmyvote.com/how-we-can-stop-all-voter-fraud/.

[42] Jeffrey Owens. Blockchain 101 for governments. *Vienna: Wilton Park. Retrieved from https://www. wiltonpark. org. uk/wp*, 2017.

[43] Christof Paar and Jan Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[44] Prashant Pawar, Gaurav Rajukar, Nisha Gaikwad, Achal Bute, and Prof. Shradha Kirve. Tracking donations of charitable foundations using blockchain technology. *International Journal of Advanced Research in Computer and Communication Engineering*, 2021.

[45] Irene Pollach, Horst Treiblmaier, and Arne Floh. Online fundraising for environmental nonprofit organizations. In *Proceedings of the 38th Annual Hawaii international conference on system sciences*, pages 178b–178b. IEEE, 2005.

[46] Mahmood A Rashid, Krishneel Deo, Divnesh Prasad, Kunal Singh, Sarvesh Chand, and Mansour Assaf. Teduchain: A blockchain-based platform for crowdfunding tertiary education. *The Knowledge Engineering Review*, 35, 2020.

[47] D Harroch Richard and D. Bass Harris. 15 key steps to set up a charity. `https://www.forbes.com/sites/allbusiness/2017/04/16/15-key-steps-to-set-up-a-charity`.

[48] Marta Calsina Ruzafa. *Blockchain as a chain for humanitarian aid: transforming the lives of refugees*. PhD thesis, 2021.

[49] Sara Saberi, Mahtab Kouhizadeh, Joseph Sarkis, and Lejia Shen. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, 57(7):2117–2135, 2019.

[50] Richards Sam. Web2 vs web3. `https://ethereum.org/en/developers/docs/web2-vs-web3/`.

[51] Bikramaditya Singhal, Gautam Dhameja, and Priyansu Sekhar Panda. *Beginning Blockchain: A Beginner's guide to building Blockchain solutions*. Apress, 2018.

[52] P Rajan Varadarajan and Anil Menon. Cause-related marketing: A coalignment of marketing strategy and corporate philanthropy. *Journal of marketing*, 52(3):58–74, 1988.

[53] W3Schools. *What is npm?*

[54] Keke Wu, Bo Peng, Hua Xie, and Zhen Huang. An information entropy method to quantify the degrees of decentralization for blockchain systems. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 1–6. IEEE, 2019.

[55] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv preprint arXiv:1906.11078*, 2019.

[56] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. Ieee, 2017.