Thai Charitable Donation Platform Based on Distributed Ledger Technology

GROUP: 39

Korapong Mahahemmarat 59070503401 aeingkm99@gmail.com

Taratorn Kamjornmenukul 59070503474 mrwoodpecker1995@gmail.com

Advisor: Professor Sally E. Goldin

18 May 2020

I have read this report and approve its content.

Abstract (English)

Nowadays, social networks have come to play a big role in charitable activity. They offer a channel for donors and beneficiaries to communicate with each other more easily. In fact, social networks reduce the gap between donors and beneficiaries but they are a double-edged sword. In the real world, donors never know whether money that they transfer will reach the destination or the beneficiary. Moreover, the campaign that beneficiary created might be fraudulent so we never know which campaign can be trusted. Moreover, from CAF World Giving Index, it shows us that the rate of donation in Thailand tend to decrease in every year from the collected data by the end of 2016 and 2017.

From the above problem, we found that the environment that donors and beneficiary participate in seems to be an untrusted environment. Our plan is to create a platform or an environment that can be trusted by both donors and beneficiaries. Therefore, our main goals are to increase the donation rate and decrease fraudulent activity.

In the environment that we plan to create, all transactions will be transparent so this will encourage donors to donate money to the campaign. Furthermore, the history of all transactions will be immutable to prevent tampering and hiding evidence of fraud. Therefore, this will benefit both donors and beneficiaries.

Abstract (Thai)

ในขณะนี้โซเชียลเน็ตเวิร์กได้มามีส่วนรวมในวงการการบริจาคโซเชียลเน็ตเวิร์กได้ให้บริการต่างๆที่เป็นประโยชน์ต่อทั้งผู้บริจาคและผู้ตั้งรับการบริจาคเช่นให้บริการช่องทางการสื่อสารที่ง่ายขึ้นสำหรับทั้งสองฝ่ายแต่ในความเป็นจริงการที่โซเชียลเน็ตเวิร์กให้บริการที่ง่ายต่อการเข้าถึงนี่จึงทำให้โซเชียลเน็ตเวิร์กเป็นเหมือนดาบสองคมผู้บริจาคบริจาคเงินให้โครงการโครงการหนึ่งแต่ผู้บริจาคไม่สามารถรับรู้ได้เลยว่าเงินที่ผู้บริจาคนั้นถึงมือผู้รับบริจาคจริงหรือเปล่ามากกว่านั้นจากตัวเลขเชิงสถิติของ CAF World Giving Index ได้บ่งบอกถึงดัชนีการผู้บริจาคของประเทศไทยนั้นตกลงมาถึง 22 อันดับภายในปีเดียว(ข้อมูลที่ได้รับมาเป็นข้อมูลของปลายปี ค.ศ. 2016 ถึงปลายปี ค.ศ 2017)

จากปัญหาข้างต้นเราพบว่าสภาพแวดล้อมที่ผู้บริจาคและผู้รับผลประโยชน์มีส่วนร่วมดูเหมือนจะเป็นสภาพแวดล้อมที่ไม่น่าเชื่อถือแผนของเราคือการสร้างแพลตฟอร์มหรือสภาพแวดล้อมที่เชื่อถือได้ทั้งจากผู้บริจาคและผู้รับผลประโยชน์
ดังนั้นเป้าหมายหลักของเราคือการเพิ่มอัตราการบริจาคและลดกิจกรรมการฉ้อโกง

ในสภาพแวดล้อมที่เราวางแผนที่จะสร้างการทำธุรกรรมทั้งหมดจะมีความโปร่งใสดังนั้นสิ่งนี้จะสนับสนุนให้ผู้บริจาคบริจาคเงินให้กับแคมเปญนอกจากนี้ประวัติของการทำธุรกรรมทั้งหมดจะถูกเก็บไว้ในรูปแบบที่ไม่สามารถเปลี่ยนแปลงได้เพื่อป้องกันการดัดแปลงและซ่อนหลักฐานการทุจริต ดังนั้นสิ่งนี้จะเป็นประโยชน์ต่อทั้งผู้บริจาคและผู้รับผลประโยชน์

Acknowledgements

We could not complete this project without the help of many people. First of all is Dr. Sally E. Goldin, an advisor of our project, who shared her time to guide us, gave us knowledge and suggestions which helped our project run smoothly. She also corrected our document which has a lot of incorrect grammar and sentences.

We are also very pleased to thank Dr. Richard Piacentini, a DLT expert from Sharkaroo.com, for helping and giving us useful suggestions for our system. We have a lot of thanks for both of them. Moreover, we would like to thank our committee for a suggestion to improve our system.

Table of Contents

Chapter 3 Design and Methodology

Chapter 4 Results and Discussion

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement and Approach

Charitable activities are an activity focused on purposes in four main categories: 1) Relief of poverty; 2) Advancement of education; 3) Advancement of religion; 4) Certain other purposes that benefit the community in a way the courts have said is charitable. [1]

Nowadays, social networks such as Facebook have come to play a big role in charitable activity since they offer a channel for donors and beneficiaries to communicate with each other more easily. In fact, a social network reduces the gap between donors and beneficiaries but it is a double-edged sword. In the real world, donors never know whether money that they transfer will reach the destination or the beneficiary. Moreover, the campaign that the beneficiary created might be fraudulent, so donors never know which campaign is trustworthy.

Based on data from the CAF World Giving Index, collected in 2016 [2], Thailand was in rank 5 in terms of the proportion of people who donated money to charity. This information shows that Thailand was among the top 10 countries that prefer to donate money to charities. In contrast, Thailand's rank dropped from the top 10 countries to number 22 based on the collected data in 2017 [3]. The cause of this decline might be trust and identification issues.

Our project will focus on improving trust issues by using Distributed Ledger Technology (DLT) to make a record of every transaction that occurs in our platform. Every beneficiary must provide identification information and information about their campaign. Therefore, when donors file a report or think that there might be a fraud this will make it easier to investigate.

In the end, our project can be classified as addressing a documented social or environmental problem or benefiting society.

1.2 Objectives

1. Reduce the rate of fraud in charitable activity.
2. Build a Thai Charitable Donation platform based on Distributed Ledger Technology.
3. All transactions that go through this platform must be traceable, transparent, and every campaign can be investigated by all users.

1.3 Scope

The scope of our project is to create a web-based platform for Thai Charitable Donation. Our target group can be separated into 2 types: 1) Donors; 2) Beneficiaries. After donors donate money into our platform, our platform must provide the ability to record and store it in an immutable manner.

1.4 Tasks and Schedule

TERM 1

1. Find a topic of interest
    a. List technology to help solve the problem
    b. Present ideas to adviser
    c. Select topic
2. Research for related work and technology
3. Project IDEA report
4. Project Proposal
5. Project Design
    a. System requirements
    b. Use case diagram
    c. Data structure design
    d. Architectural design
    e. User interface design
    f. Plan system test
    g. Plan the deployment
    h. Project design complete
6. Experiment with tools
    a. Experiment on Stellar network
    b. Experiment on Angular framework
    c. Building a prototype
7. Proposal Presentation
    a. Writing and preparing PPT slide
    b. Presentation practice

    c. Present proposal

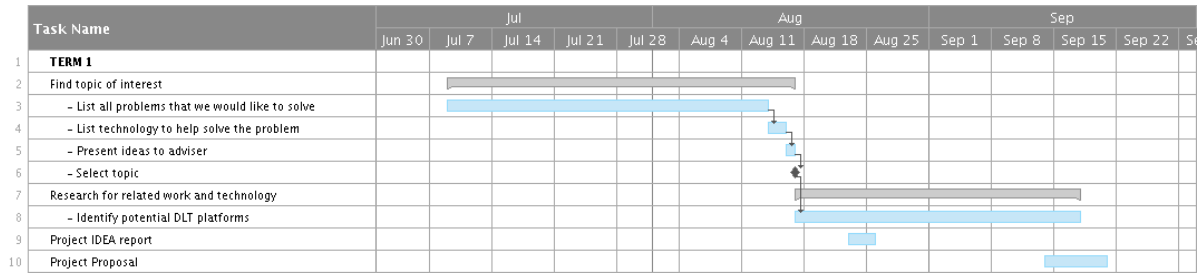
8. Semester Report
9. Final Presentation
    a. Writing and preparing PPT slide
    b. Presentation practice
    c. Present final report

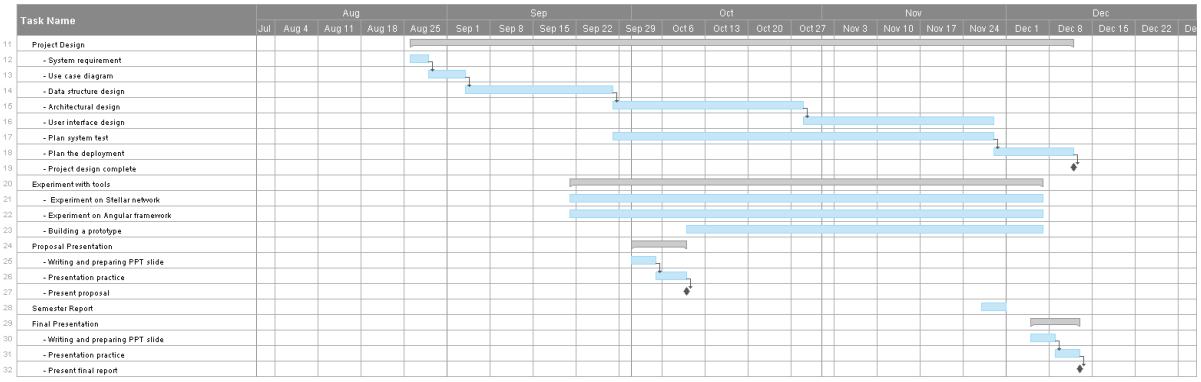
TERM 2

1. Project Development
    a. Develop system modules
    b. Integrate system modules
    c. Perform initial testing
    d. Development complete
2. Testing
    a. System testing
    b. Identify problems
    c. Fix bugs
3. Deploy Website

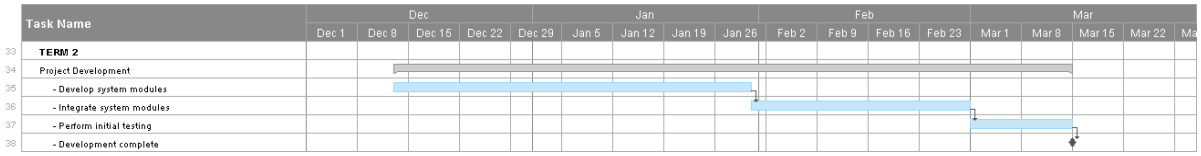
1.5 GANTT CHART

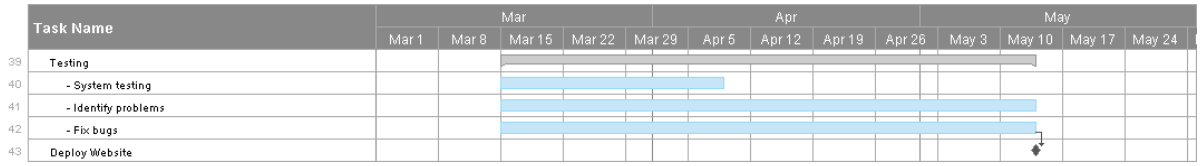
TERM 1





TERM 2





1.6 Deliverables

Term 1

1. User interface and site navigation design
2. Use case narratives
3. Architectural design
4. Website prototype
5. Blockchain database prototype

Term 2

1. Donation Website
2. Final Report

Chapter 2

Background, Theory and Related Research

2.1 Market Comparison

2.1.1 Taejai Thai Charitable Donation Platform (https://taejai.com/th/)

Taejai is a Charitable Donation Platform that is based in Thailand.

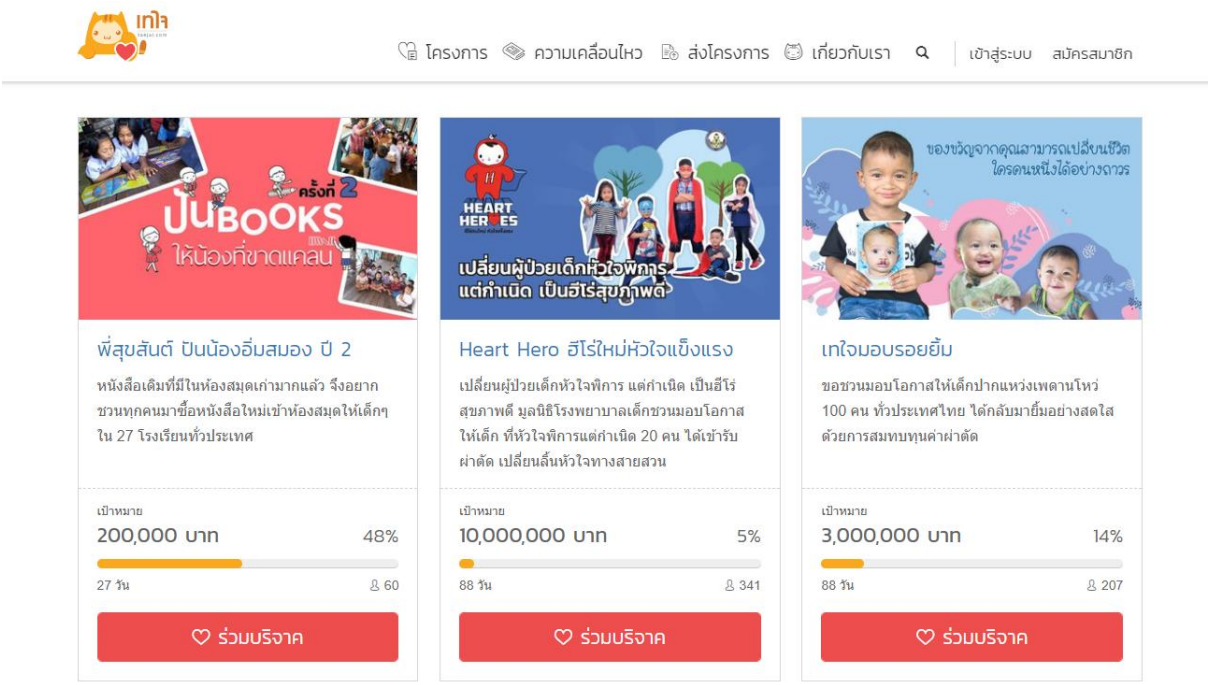


Figure 2.1: Homepage of Taejai.com

[Source: [44]]

Figure 2.1 shows a screenshot from this platform, which provides campaigns for donors to donate money. Moreover, the platform also displays the current amount of money donated into the campaign. In order to make a campaign on this platform user must provide a lot of information and campaign scope must be big enough. Furthermore, beneficiaries need to wait for around 3-7 days for their campaign to be verified by platform administrators.

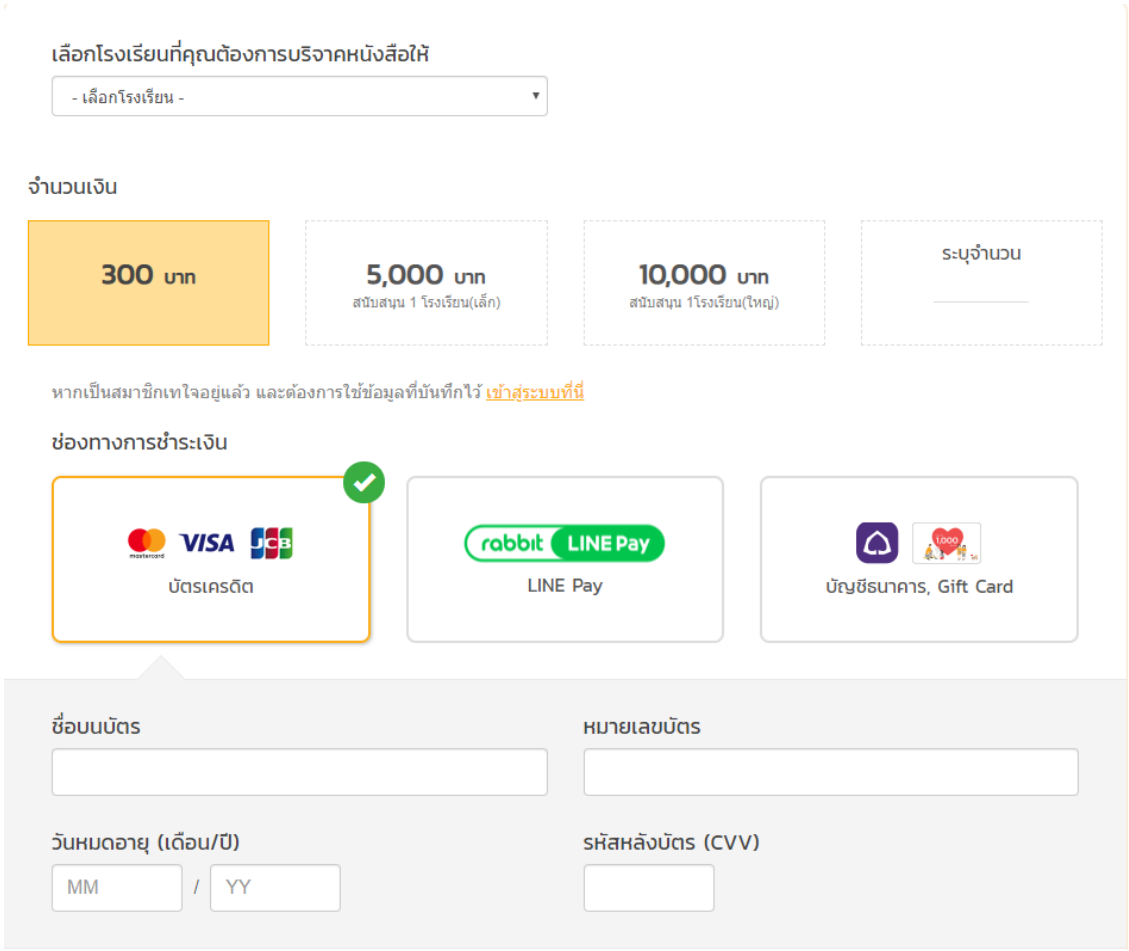


Figure 2.2: Taejai donation page from "พี่สุขสันต์ ปันน้องอิ่มสมอง ปี 2" campaign

[source:[45]]

Figure 2.2 shows that this platform allows users to donate even though they have not logged in to the system, but users need to provide credit card information and some personal information.

In contrast to the above, our project will allow any scope size of campaign. Moreover, our project will store all transactions data with DLT which is make transactions are traceable.

2.2 Asymmetric cryptography

Cryptography is the art of writing or solving codes through complex mathematics. The main advantages of Asymmetric cryptography are to increase the security of data. Moreover, Asymmetric cryptographic also provides *Digital Signature* that can ensure three characteristics: 1. Authentication 2. Non-repudiation 3. Integrity. (See section 2.3.2.1 for more information).

Asymmetric cryptography, also known as *public-key cryptography,* is one of the techniques that is used in cryptography field. 'Asymmetric' when applied to cryptography means unequal or non-equivalent. The term highlights the difference in visibility between the public key which everyone can see, and the private key which is secret. This cryptography technique generates a key pair in order to encrypt and decrypt the data which is linked by mathematical equation [4]. One is known as public key and another known as private key. One of those keys will be used to encrypt the data and the other will be used to decrypt the data. [5] This technique used to ensure integrity of data that passes through the insecure channel.

A *public ke*y is like a username or an ID of user. This key can be shared or available to be seen by everyone. The main role of public key is used to encrypt the data so that no one can open or tamper it.

A *private key* is like a password of user ID. This key should not be shared with anyone. The owner of the key must keep the key confidential. The main role of private key is used to decrypt the data encrypted by the public key.

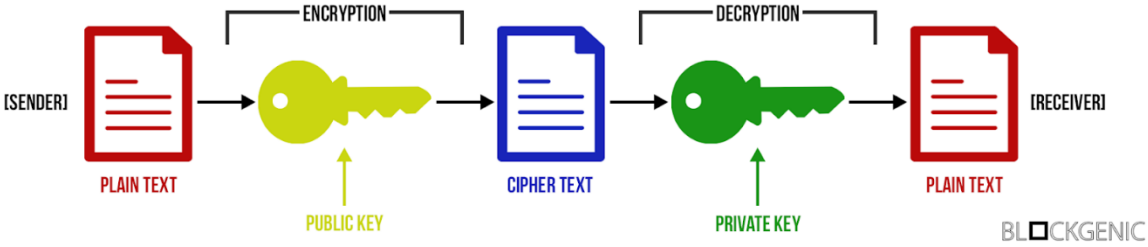


Figure 2.3: Process of how Asymmetric key works

[source: [46]]

Figure 2.3 shows the process of how the asymmetric cryptography works. First, the receiver needs to send the public key to the sender so that the sender can encrypt the data. The data that is encrypted by the public key of the receiver, so it cannot be read or tampered with by anyone due to the fact they need private key to decrypt the data. After the data has been encrypted, the data can be sent via an insecure channel. Next, after the receiver receives the data, the receiver needs to do an identity verification by using the receiver's own private key to decrypt it. [6]. This process ensures the integrity of the data. If the data have been corrupted, the private key will not produce a comprehensible plain text message.

2.3 Distributed Ledger Technology (DLT)

2.3.1 General concepts of DLT and benefits

Distributed Ledger Technology is a shared database that is based on a Peer-to-Peer network. *Peer-to-Peer* is a network where the computers are connected to each other and are able to transfer data directly to each other without any central authority. All participants or nodes in the network have their own identical copy of shared database. All items that are put in the ledger will be shared among all nodes in the network under the network consensus [7].

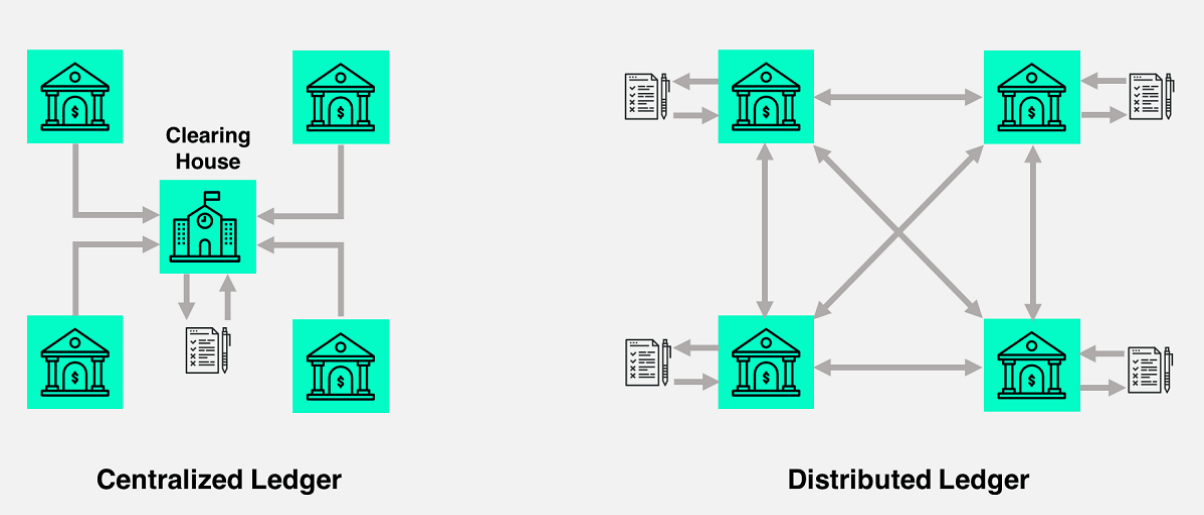


Figure 2.4: The difference between Centralized and Distributed ledger

[source: [47]]

Figure 2.4 illustrates how Distributed Ledger Technology stores data in different nodes (for instance, financial institutions or banks). There are a lot of benefits to Distributed Ledger Technology: 1) Cost saving: Transferring money or converting currency from finance institution (Bank), may cost a lot of money but DLT is based on Peer-to-Peer network so it helps each node to communicate with each other directly without the central authority which can reduce transaction fee from transferring money; 2) Reliability (no single point of failure): DLT is a distributed database. In fact, distributed database is a

database that spread around the network which means one failure of a database node won't affect the whole network.

DLT also has its disadvantages, in particular due to scalability problems. From the increasing number of transactions, each node needs more storage space in order to hold an identical copy of the ledger. Furthermore, the more nodes in the DLT network, the longer it may take to jointly validate and approve a ledger change and to send it to all the nodes.

2.3.2 Blockchain Technology

Blockchain Technology is a subcategory of Distributed Ledger Technology due to the fact Blockchain is a distributed database, and stores data in distributed way. In a Blockchain all nodes will hold an identical copy of the history of the ledger, which has the form of an immutable series of transaction blocks, where each block contains a cryptographic reference to the previous block. Blockchain is essentially a method for storing data among multiple parties with an immutable and transparent manner. Since Blockchain is a subcategory of DLT, a blockchain network is based on Peer-To-Peer network. The participants in the network can talk directly to each other without the control of central authority.

Every transaction or item of data that occurs inside the network will be signed by Digital Signature. After digital signature is signed, the transaction will be broadcast into the network and blockchain validator will gather it and bundle it into the block. The size of each block will depend on blockchain protocol. For example, in the Bitcoin protocol the size limit of block is 1 MB. However, a larger block capacity does not always benefit to the network. A larger block allows more transactions to be bundled in one block, but also consumes more storage space and makes the network slower [8].

The way that the block or ledger is updated will depend on the consensus algorithm of each blockchain protocol. From the above, all transactions can be done with a

transparent manner and verifiable even if there is no central authority to verify the transaction. This means trust is no longer an issue of the network anymore so this makes blockchain become a *trustless network*. [9]

A core concept of a blockchain is that each block has exactly one valid successor block. However, due to the distributed nature of blockchain, it is possible that two different blocks might be proposed as the successor block. This is a temporary situation, called a ***fork.*** *A fork* must be resolved by the network participants, by validating and choosing one successor block and discarding the other(s). This one important function of a blockchain's consensus protocol.

2.3.2.1 Architecture of Blockchain Network

There are 4 core components that are used to build a blockchain network: 1) Node; 2) Block; 3) Digital signature; 4) Consensus algorithm.

(i) A *Node* is simply a device or computer that participates in the blockchain network. *Node* can be separated into 2 types: 1. Partial Node 2. Full Node

1.*Partial Node* or *Node is* simply a device or computer that participates in the blockchain network. The main role of node is to share and send information of transactions and blocks between nodes.

2.*Full Node* or *Fully Validating node (Blockchain Validator)* is a node that holds a copy of the entire blockchain and helps the network validate transactions and blocks from the beginning block or *genesis block* to the most recent block [10]. The Full Node will validate transactions and blocks using the consensus protocol [11] to prevent invalid transactions, for instance double-spending in

Bitcoin Protocol. *Double-spending is* one type of attack where senders try to send the same digital currency more than once.

(ii) A blockchain stores its transaction or data in a concept of *block.* Blocks are chunks of data that are connected consecutively by the hash function. Each block contains a hash reference to all the data in the previous block, so that if the previous block is changed, this tampering can be detected. (The specific hash function depends on blockchain protocol). Block is where the network stores the data. As mentioned above, the first block created in the network is known as the *Genesis Block*. Genesis Block is a special block compared to other block, due to the fact that it is the first of the network so it has no previous hash or Parent Block Hash.[12]

(iii) Since blockchain does not have any control by the central authority, each node that participates in the network needs to create some kind of agreement to ensure the validity of transactions. The mechanics for the agreement of validity is called *consensus algorithm*. Consensus algorithm is an algorithm that is designed to keep the blockchain network secure and decentralized. Each consensus algorithm has their own unique characteristics. [13]

Table 2.1: Consensus Algorithm comparison

characteristics	Consensus Algorithms				
	Proof of Work	Proof of Stake	Delegated Proof of Stake	Practical Byzantine Fault Tolerance	Raft
Consensus fault tolerance (the ability of a distributed network of computers to correctly reach a sufficient consensus despite node failure)	50%	50%	50%	33%	N/A
crash fault tolerance	50%	50%	50%	33%	50%
verification speed	>100s	<100s	<100s	<10s	<10s
throughput (TPS)	<100	<1000	<1000	<2000	>10k
scalability	Strong	Strong	Strong	Weak	Weak

[Source: [48]]

As shown in Table 2.1, each consensus algorithm has its own advantages and disadvantages. For instance, in Proof of Work even Consensus Fault Tolerance is around 50% but the throughput from Proof of Work is less than 100 transactions per second compared to Practical Byzantine Fault Tolerance Consensus Fault Tolerance is only 33 percent they need around 66 percent of agreement between

node in order to confirm the transaction (see Section 2.3.5.3) but Practical Byzantine Fault Tolerance provide a very high rate of throughput. In order to consider which consensus algorithm is good or bad will depend on these characteristics like Consensus fault tolerance, throughput, etc. Therefore, in order to create a good consensus protocol all of these characteristics from table 2.3 need be considered.

(iv) A normal signature is a handwritten name that is put in a document for verification and prove the identity of the writer. A *digital Signature* works in the same way but it uses a hash function and encrypts the signature data with a private key from asymmetric cryptography to generate a Digitally Signed Document. From the use of asymmetric cryptography this can ensure the data integrity. [14] There are 3 main purposes of Digital Signature: 1) Authentication; 2) Non-repudiation; 3) Integrity.

Authentication - Digital Signature can guarantee that the message was actually created and sent by the sender.

Non-repudiation - Once the message has been sent with a Digital signature, the sender cannot deny he or she is the one who signed it.

Integrity - Digital Signature ensures the data sent from the sender cannot be tampered with.

Every participant in blockchain network will hold their own pair of public and private keys. Most of blockchain platforms use the Elliptic Curve

Digital Signature Algorithm (ECDSA) [15]. Digital Signature can be separated into 2 phases: 1. Signing Phase 2. Verification Phase. [12]

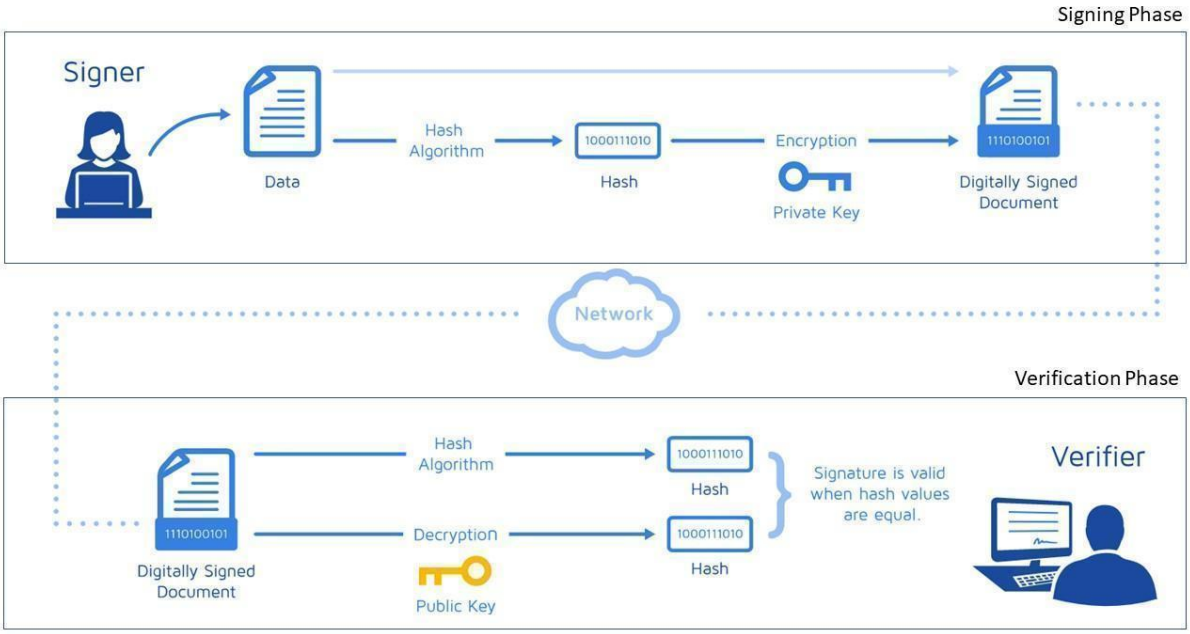


Figure 2.5: Signing and Verification phase

[source: [49]]

Signing Phase is phase of encrypt data or transaction with the sender's private key. As you can see from the figure 2.5, first the message data from the receiver will be hashed by hash function. The hash function can be changed depending on the protocol. For instance, Bitcoin use SHA-256. After that the hash value will be encrypted with the sender's private key so this step will produce a digital signature. After the sender creates a Digital Signature, this is combined with the original data to get a Digitally Signed Document. Then the Digitally Signed Document will be sent to the receiver.

Verification Phase this phase will begin after receiver receives a Digitally Signed Document from sender. The Digitally Signed Document will be separated in the original message data and the Digital Signature. The

original will go through the hash function and the Digital signature will go through the decrypt algorithm with the public key from the sender to decrypt it. Both of these functions will generate a hash output. Lastly, the receiver compares those two outputs. If both hash values are the same the receiver will know the message has not been tampered with and that it was actually sent by the claimed sender.

2.3.3 Types of Blockchain network

We can classify type of blockchain along two dimensions, based on who is allowed to join, and who is allowed to modify data.

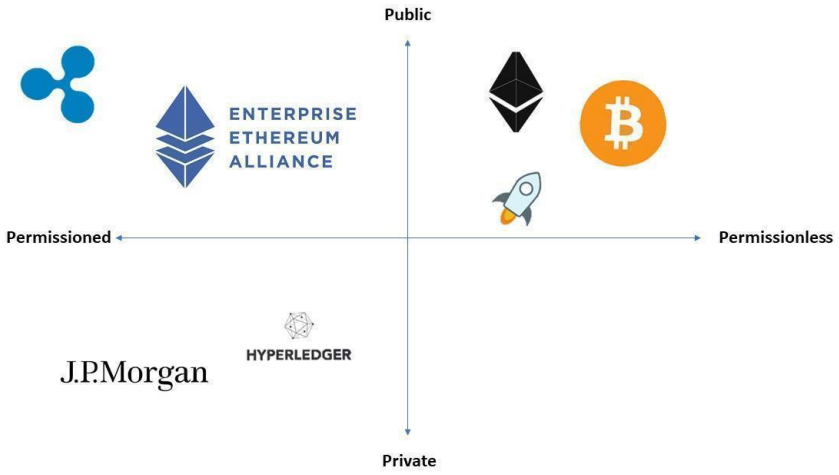


Figure 2.6: Blockchain classified into two dimensions type of network

[Source: [50]]

(i) *Public Permissionless network* - Anyone can join the network and become a node. By doing so they gain permission to read, write, verified and participate in that network [16]

(ii) *Public Permissioned network* - Anyone can access the record to read verified data in the network but only authenticated people can add/write a record. [16]

(iii) *Private Permissioned network* - Permission in the network are kept centralized to a certain extent by a company, authority or organization. These types of organizations may only let some of the known or trusted nodes to gain permission in the network. [17]

(iv) The concept of a private permissionless network does not really make sense.

(v) *Consortium network* - A consortium network is partially private and has many similarities to private network. But instead of letting one entity govern the network, multiple organizations come together to form a consortium network or some would say it is a semi-decentralized blockchain. [16]

Each type of blockchain network has its own advantages and disadvantages as we will discuss in the following paragraph

2.3.3.1 Benefits of public permissionless blockchain

Open for all readers and writers - Let anyone participate in the network and still remain anonymous.

Transparency - Provide full transparency of any data in the ledger.

Decentralize - Provide decentralize network. [17]

2.3.3.2 Benefits of private permissioned blockchain

Privacy and Scalability - Controlled by a single entity and let only invite and trusted node to access the network to read and write the data. [17]

High performance - Since the private blockchain has fewer nodes than the public, which makes data verification much faster.

2.3.3.3 Benefits of consortium blockchain

Have many advantages as a private blockchain **-** Provide privacy, scalability, and high performance.

Govern as a group - Instead of governed by one entity like a private one, consortium is governed by a group of collaborating organization to form an operating rule. [18]

2.3.3.4 Example of Blockchain protocols from each of the above blockchain type.

Bitcoin Protocol (Public Permissionless) - Bitcoin protocol is in a public permissionless category. Anyone can be a node, and participate in the network but still remain anonymous. Because anyone can read, write, validate the transaction, all transactions are transparent.

Ripple Protocol (Public Permissioned) - Ripple protocol is in a public permissioned category. Anyone can join the network but only chosen sets of validators can validate the transaction [19]

Hyperledger Protocol (Private Permissioned) **-** Hyperledger protocol is in the private permissioned category. It only allows known or trusted identity to become a node and gain access to the network. Therefore, using this protocol will gain scalability and privacy in the network.

2.3.4 Bitcoin

Bitcoin was the first system that implemented blockchain successfully. From the characteristics of blockchain, bitcoin protocol is a distributed ledger with a Peer-to-Peer network. Bitcoin use Proof of Work as its consensus protocol. Moreover, in order to keep information secure Bitcoin, use SHA-256 as the protocol hash function. SHA-256 is a cryptographic hash function designed by the NSA (National Security Agency) of the United States. The input of this function can be any size but after it goes through the hash function the output will be digested into 256 bits or 64 characters. The main characteristic of this

hash function is it cannot be traced back to plain text once it converted to cipher text. This is also known as a one-way hash function.

Bitcoin is also known as the first cryptocurrency that based on Blockchain Technology. The purpose of the Bitcoin network is to allow participants to exchange digital tokens that have real world value, as well as to create new tokens (thus acquiring additional value).

2.3.4.1 Node

Bitcoin supports the typical blockchain node types, partial node and full node, but there is one more type of node which is a Miner node.

Miner is a full node that has an ability to create a new block. First, the Miner downloads the entire blockchain like full node. Second, Miner needs to verify incoming transactions to determine whether they are valid or not. If a miner finds an invalid transaction the miner will not include it into the block. Third, the miner will create a block into which they will bundle all transactions. Fourth, in Bitcoin, the miner needs to do Proof of Work, which consists of finding a valid nonce. This will be discussed in greater detail in section 2.3.4.2. Fifth, after miner finds a valid nonce, miner will broadcast the new block to the whole network and gain a profit from creating a new block. The process of building a block is called *mining*.

2.3.4.2 Block

The Bitcoin blockchain stores its transaction or data into *blocks* that connect consecutively by the hash function. Block is where the network stores the data. .[9]

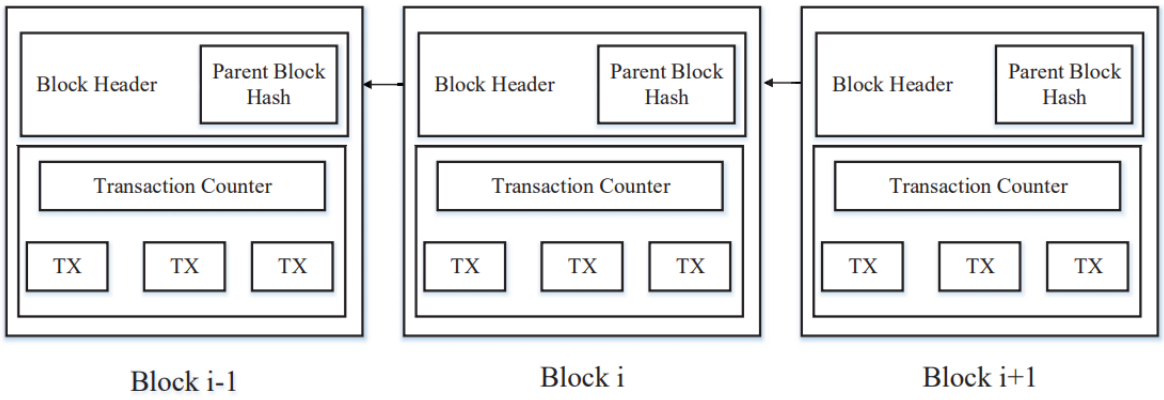


Figure 2.7: Block structure connected by hash function

[Source: [51]]

As shown in Figure 2.7, each block in the figure is connected by a hash pointer that points to the previous block. The structure of block can be separated into 2 main parts which are block header and block body. The body of block will hold the count of transactions and the actual transaction information that has been collected by the Miner. Each transaction will be hashed together by hash function called *SHA-256.* The hashed value from all transactions is called the *Merkle root.*

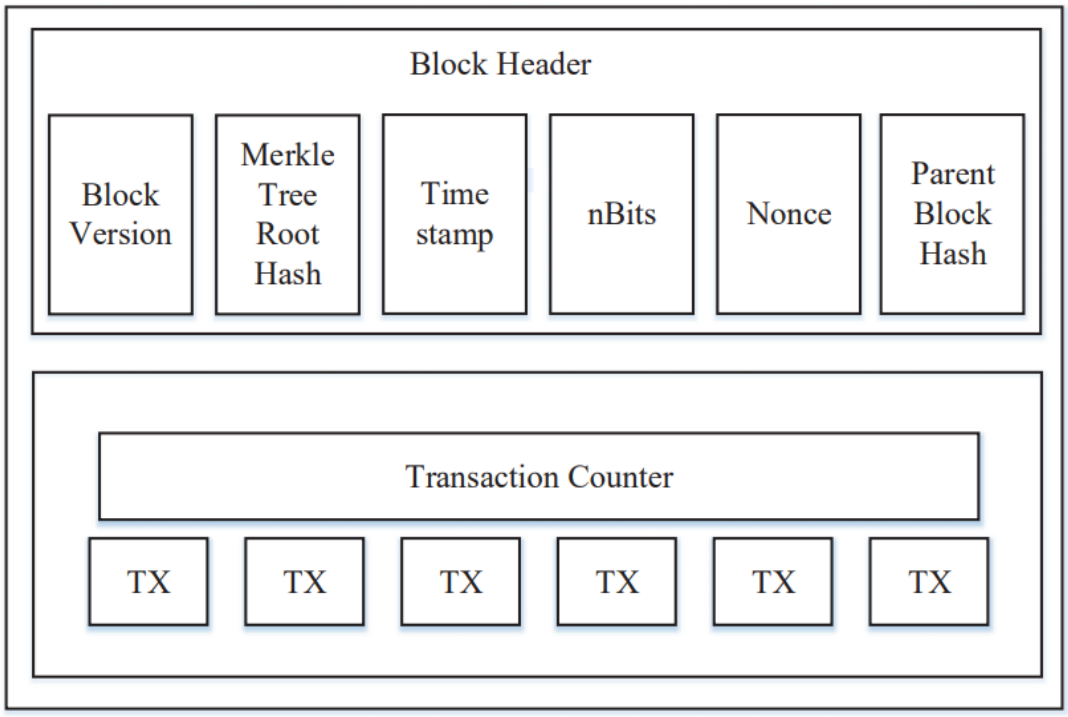


Figure 2.8: Block header with more detail

[Source: [52]]

Figure 2.8 illustrates the block header in more detail. The header of the block consists of Block Version, Merkle Tree Root Hash (Merkle root), Timestamp, nBits (Target), Nonce, and Parent Block Hash or Previous Block Hash with the overall size of 80 bytes. [20]

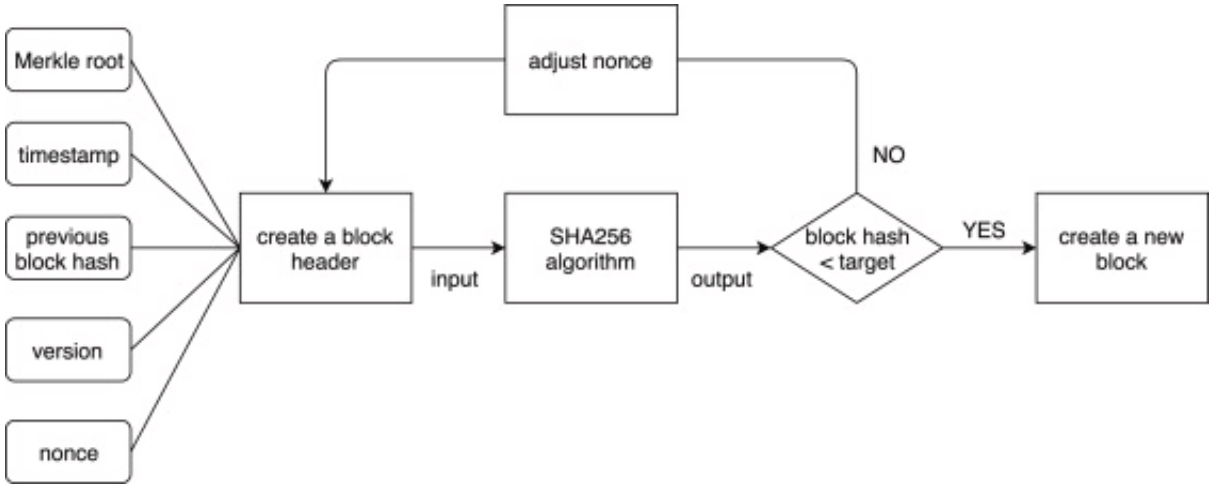


Figure 2.9: Block creation process

[Source: [53]]

Figure 2.9 illustrates the process of creating a Bitcoin block. In order to create, first miners need to propose a new block and solve a puzzle from Proof of Work consensus. This process will consume a large amount of computational power of miner so this is why this consensus is called Proof of Work. The process works as follows: first miner will listen to the network and gather incoming transactions that pass through the network with the limited amount of 1 MB. In this process, most of miner will gather only transactions with a high fee and hash every transaction with SHA-256. After every transaction hashed, the output hash will be

stored in Merkle root. Second, miners need to combine all data together by SHA-256 in order to create a block header. (See Table 2.2 for more information) Third, the miners need to hash the output from second step by SHA-256 hash function again. Fourth, after miners got an output from hash function, they need to compare it with the target value or nbits (SHA (SHA256(Merkle root | timestamp | prev | version | nonce)) < target). If the hash value is greater than the target value, miners need to increment nonce and do the hash function again. In contrast, If the hash value is less than the target or nbit this mean the Proof of Work solved. After that, miners will broadcast a new block to the network and begin to mine a new block again.

Table 2.2: Description of block header data

[Source: [54]]

Name	Size	Description
Block Version	4 bytes	number that use to indicate the version of block
Merkle root	32 bytes	An output value from hash every transaction that miner gathered.
Timestamp	4 bytes	Block closed Timestamp
nBits(target)	4 bytes	nBits or target is a threshold of the target hash that miners need to find. The hash value must be below the threshold in order to reach the consensus. The threshold can be adjusted in order to control the number of new blocks that are created. The ideal rate block creation is around 10 minutes per 1 block. This value will be adjusted every 2 weeks or 2016 blocks. If blocks created more than 2016 blocks in past 2 weeks the target value or nBits will be adjusted lower than the previous nBits.Which mean the block creation will be more difficult. In contrast, if the block creation rate tends to be lower than 2016 blocks this value will be adjusted higher in order to increase the block creation rate.

Nonce	4 bytes	nonce is a number that use to combine with other block header data. The way it combines with other data is by using SHA-256. The value of nonce starts from 0 and will increment every time after miner do a hash in order to find the hash output that lower than target.
Previous Block Hash	32 bytes	a hash value that points to the previous block hash

From table 2.2, the data in the block header are very important in order to create a new block which will describe below.

2.3.4.3 Consensus Protocol

In a network with consensus, there are many kinds of cyber-attack that can harm the bitcoin network such as Double Spending Attack which use to trick the server and create new amount of copied currency that did not previously exist. For an example, Sam wants to buy an apple from Jane from the online shop. The apple is costs around $5 but Sam didn't want to spend his own money so he decides to do a Double Spending Attack on Jane. First, Sam will send create two transactions with the same amount of money and send it at the same time. One of transactions will be sent to Jane (Transaction A) and another one will be sent to Sam another account (Transaction B). If Sam has a multiple accounts and votes for Transaction B and most of validator believe that B is the valid transaction, Transaction A will be invalid and Jane won't get $5. In order to prevent cyber-attacks, Bitcoin needs some kind of consensus algorithm which is known as *Proof of Work. Proof of Work* is an algorithm for creating a new block. In order for nodes to participate in voting system in Bitcoin network, that nodes must proof that have enough potential by finish the Proof of Work consensus algorithm. Proof of Work requires a large amount of computational power due to the fact that miners need to execute a hash function that can only be done by brute force. This process is like throwing a dart

while blindfolded. The output of hash function is pretty much random so to finish the Proof of Work algorithm is much like winning a lottery.

2.3.5 Stellar Network

Stellar is a public permissioned blockchain network [21] due to the fact that the network allows only some level of nodes to validate transactions. The purpose of this network is to move money or any kind of assets across borders with improved speed and reliability, and low cost. The transaction of assets or money will be transferred through the participating financial institutions and recorded as money in the participants' accounts. The currency of Stellar network is known as XML or Lumens. When it went live in November 2015 the network contained a total of 103.78B XLM due to the design of the protocol.

Figure 2.10: Overall view of Stellar network

[Source: [55]]

This type of protocol does not require miners to validate the incoming transactions like Bitcoin. The validation in Stellar achieved by Stellar Consensus Protocol or SCP which will be discussed in detail in *Section 2.3.5.3*.

2.3.5.1 Stellar Core

Stellar core is a backbone or core component of the Stellar network. Each node in Stellar network contains its own Stellar core. Stellar core uses C++ as a

language for implementation in order to construct a chain of ledger. The main process of Stellar core is validating and agreeing with other Stellar cores and communicate with other nodes on the network. This process will happen through SCP or Stellar Consensus Protocol. Stellar Core consists of 7 main components 1. SCP 2. Herder 3. Overlay 4. Ledger 5. History 6. Bucket List 7. Transactions. Moreover, inside Stellar core it contains an SQL Database for storing necessary data

1. *SCP* is an implementation of Stellar Consensus Protocol and it is fully abstract from the rest of the system.
2. *Herder* is component is an interface that connects between SCP component and the rest of the Stellar core in the network. Inside Herder component there is a HerderSCPDriver which is a concrete implementation of the SCP protocol that allow SCP to do its method.
3. *Overlay* is a component that is used to flood messages to the network and fetch needed data in order to accomplish consensus from its peers.
4. *Ledger* is an output transaction set from SCP process. This will be stored in SQL Database that located in Stellar core.
5. *History* is the historical data that come from transaction set such as transaction fee history, transaction history. This data will be stored in SQL database.
6. *Bucket List* is a component for storing data and come to help SQL Database to serve these 2 operations ‘Efficiently calculating a cryptographic hash of the entire set after each change to it’ and ‘Efficiently transmitting a minimal "delta" of changes to the set when a peer is out of sync with the current

ledger state and needs to "catch up'. From the above, SQL database is inconvenient or intractable in its SQL storage form.

7. *Transactions* is component is used to implement all the various types of transaction [43]

2.3.5.2 Stellar Nodes

A Node in Stellar can be any device that connects to the network. For anyone running a node in the Stellar network, Stellar will provide benefits such as providing a Horizon instance (see section 2.3.5.4) and allowing for customization of business logic or APIs, etc. [22]. Nodes in the Stellar network can be classified into four levels, Watcher, Archiver, Basic Validator and Full Validator. Each node will connect to network through Horizon and Stellar core.

Table 2.3: Role of each level of node

[source: [56]]

	watcher	archiver	basic validator	full validator
description	non-validator	all of watcher + publish to archive	all of watcher + active participation in consensus (submit proposals for the transaction set to include in the next ledger)	basic validator + publish to archive
submits transactions	yes	yes	yes	yes
supports Horizon	yes	yes	yes	yes
participates in consensus	no	no	yes	yes
helps other nodes to catch up and join the network	no	yes	no	yes

Increase the resiliency of the network	No	Medium	Low	High

Table 2.3 displays the benefits and restrictions on each level of node. Each level has their own responsibility and benefits. *Watcher node* is a node that simply watches the activity of the network. *Archiver node* is responsible for recording the activity in the work in long term storage, plus it can do whatever a watcher node can do. *Basic validator* is a node that can participate in the voting system of the network, as well as function as a watcher. *Full validator* is a node take a full responsibility of maintaining the network and participate in consensus. [22]

Anchor is an architecture concept specific to the Stellar blockchain system. An anchor is a node that functions like a bridge to connect the real-world currency to Stellar network currency. The main role of anchors is to convert real money into Stellar credit and vice versa. Mostly, the anchors are financial organizations like banks, etc. [23]

2.3.5.3 The Stellar Consensus Protocol

Stellar Consensus Protocol: First we need to talk about the idea behind *Stellar Consensus Protocol*. Byzantine Fault Tolerance (BFT) is the ability of a distributed network of computers to correctly reach a sufficient consensus despite node failure. BFT is derived from the Byzantine Generals Problem, which is a situation where there are parties that must agree on a single strategy to avoid failure and also assume that some of the involved parties might be corrupt or malicious. Practical Byzantine Fault Tolerance (pBFT) is an algorithm that optimizes the characteristics of BFT. In a pBFT system there must be a

recommended validator list that is defined by a central authority. pBFT can reach a consensus by broadcasting the agreement message to the others. Systems such as Hyperledger uses pBFT.

pBFT was replaced with Federated Byzantine Agreement (FBA) in Stellar Consensus Protocol because FBA provide a decentralized alternative way to pBFT[24]. FBA describe a way to achieving agreement across many validators in the network. Transactions are verified by a group instead of one node broadcasting to each other like pBFT. The following are key features of FBA:

i. *Decentralized control of the network*: there is no central authority to dictate whose approval is required for the consensus. So, anyone can participate in the consensus
ii. *Low latency:* consensus can be reached in a few seconds at most.
iii. *Flexible trust*: each validator in the network is free to decide which other validator they trust without restriction.
iv. *Asymptotic security*: as stated in the Stellar Consensus Protocol white paper [60], safety relied on using hash families and digital signature which can be tuned to protect against adversaries with high computing power.

Stellar uses SCP to reach consensus. Unlike other consensus protocols such as Bitcoin's proof of work that use Bitcoin generation as an incentive for nodes, Stellar Lumen (XML) is independent from its consensus protocol. Running a validator node in Stellar does not generate income but the node will have other benefits such as the node can participate in consensus thus make the network more secure.

There are three properties that a consensus protocol needs, which are safety, liveness, and fault tolerance

A consensus protocol can provide **safety** if

- All the outputs produced to have the same value
- Output value equals one of the nodes' inputs values

A consensus protocol can provide **liveness** if

- Eventually all the non-faulty nodes will actually output a value

A consensus protocol can provide **fault tolerance** if

- Protocol can recover from the failure of a node at any point in the execution of a protocol
- There are two different kinds of fault tolerance which are
  - Fail-stop model: In this model, it assumes that nodes fail by crashing and never send any more messages
  - Byzantine-fault-tolerant model: In this model, it assumes that nodes fail by behaving arbitrarily

In theory (FLP impossibility result) [61], no deterministic consensus protocol can guarantee all three characteristics of safety, liveness, and fault tolerance in an asynchronous system

Basically, all consensus protocols choose **fault tolerance** property as one of their properties. This leaves the last two properties for a protocol to choose and SCP favors safety over liveness

At the core of SCP are quorums and quorum slices. Node in the SCP are free to choose whichever nodes they trust for information and a sub group of node that trust each other is called a quorum slice. A quorum is a set of nodes that contain

at least one slice of each of its members. For example, a set of nodes that are trusted by node **n,** plus node **n** itself defines the quorum slice of **n**. Let's say the set of nodes contains **m** and **p**. Node **n** will agree to a statement only if nodes **m** and **p** agree on the statement. If nodes **m** and **p** agree on the statement then node **n** will also agree to it as well. Now, if **m** and **p** also have other nodes in their own slices these nodes are necessary to form the quorum

To be able to see a clearer picture, the following figures are an example of quorums and quorum slices

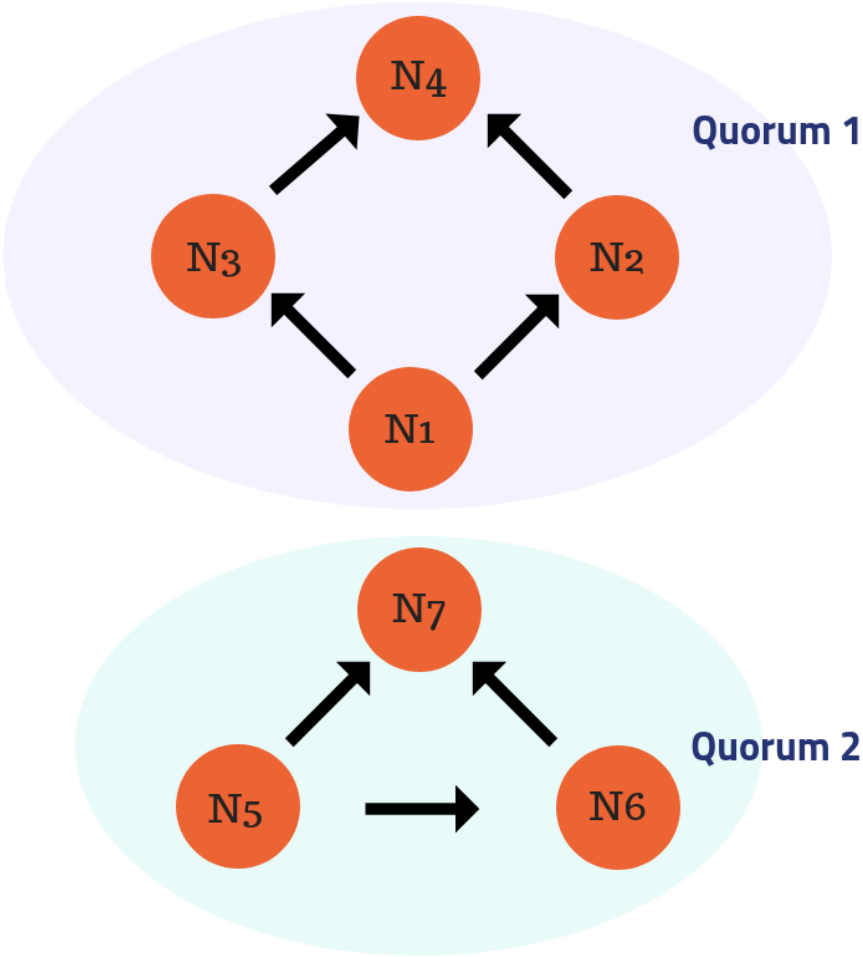


Figure 2.11: Disjoint Quorums

[Source: [57]]

The above figure consists of: N(Node) = N1, N2, N3, N4, N5, N6, N7 and the arrows represent quorum slices for example, quorum slice of N1 is N1, N2, N3

Quorum 1 composed of N1, N2, N3, and N4

Quorum slices (Q):

Q (N1) = N1, N2, N3

Q (N2) = N2, N4

Q (N3) = N3, N4

Quorum 2 composed of N5, N6, N7

Quorum slices (Q):

Q(N5) = N5, N6, N7

Q(N6) = N6, N7

From the above example, in quorum 1: N4 can determine the agreement of quorum 1 because N1 will agree to a statement only if N2 and N3 agree to it. N2 will also agree to a statement only if N3 and N4 agree to it so does N3. So, N1 is never to be able to accept any statement unless N4 agrees to it. So, the smallest quorum that contain N1 is a set of all nodes in the system N1, N2, N3, N4.

In this case, quorums will not interfere with each other since there is no nodes that are shared between quorum 1 and quorum 2. We called two separate quorums that don't intersect with each other "disjoint quorum". With disjoint quorums, there is no way to guarantee the two consistent statements. So, in order to guarantee safety that the two quorums will consistently validate a statement, the SCP requires quorum intersection because a node that intersect is the one that two quorum will accept statement so they can all agree on the same statement otherwise the network could reach separate agreement and a statement might get stuck in a permanently indeterminate state.

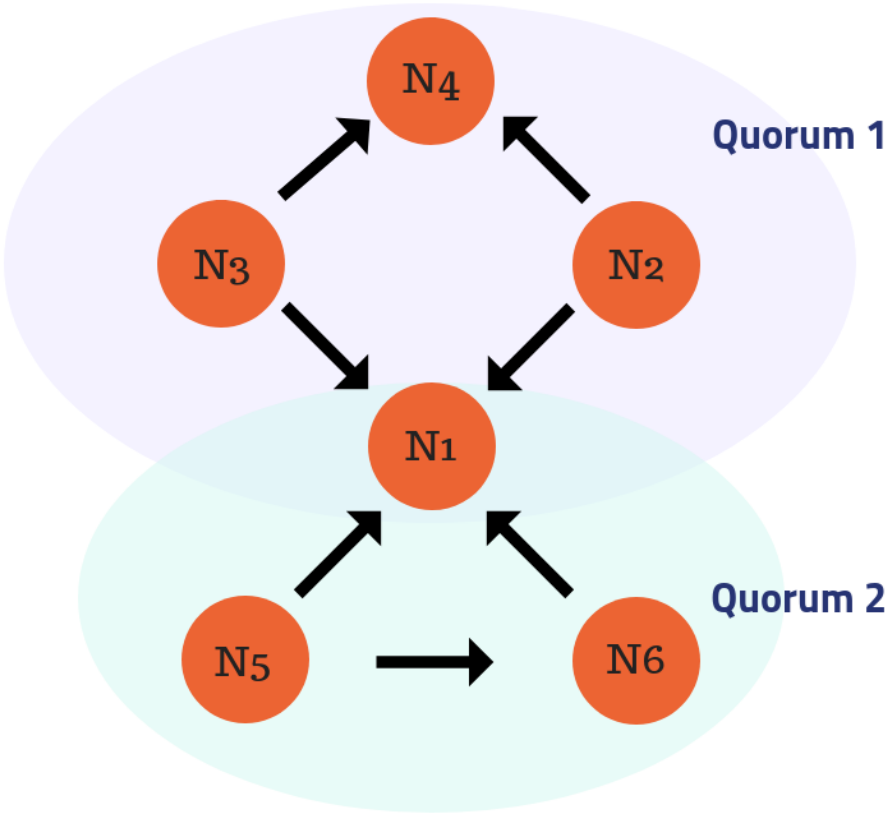


Figure 2.12: Intersect Quorum

[Source: [58]]

The above figure consists of: N(Node) = N1, N2, N3, N4, N5, N6 and the arrows represent quorum slice for example, quorum slice of N5 is N5, N1, N6

Quorum 1 is composed of N1, N2, N3, and N4

Quorum slices (Q):

Q (N2) = N2, N1, N4

Q (N3) = N3, N1, N4

Quorum 2 is composed of N1, N5, and N6

Quorum slices (Q):

Q (N5) = N5, N1, N6

Q(N6) = N6, N1

From the above example, when two quorums share a node that essential for its own agreement, we called it “quorum intersection”. In this case, N1 is at the point of intersection and is needed for determining an agreement of quorum 1 and quorum 2. In this case, the two quorums will externalize consistent statements.

From the diagrams above we can also see that each quorum slice also has tier of its own. We can rank from top to middle to leaf tier in order of importance for consensus. For example, in the disjoint consensus diagram, we can rank N4 is a top tier because it is the most important node in the consensus, followed by N3 and N2, then followed by N1.

When reaching the consensus and by agreeing on what updates to apply to the ledger, the SCP white paper says that “For instance, slots may be consecutively numbered position in a sequentially apply log” in this sentence we can interpret that “Slot” in blockchain is basically a block. All validator in the network must agree upon the validity of that block and then update their personal ledgers to include the agreed block once it is published After that we can say the consensus is achieved.[25]

Node Failure

The situation discussed above is just an ideal scenario. In fact, it is not easy for nodes to reach an agreement because of node failures. So, to create a system that is fault tolerant, the system must prepare for the worst. In this case we can classify node behavior into types as follows.

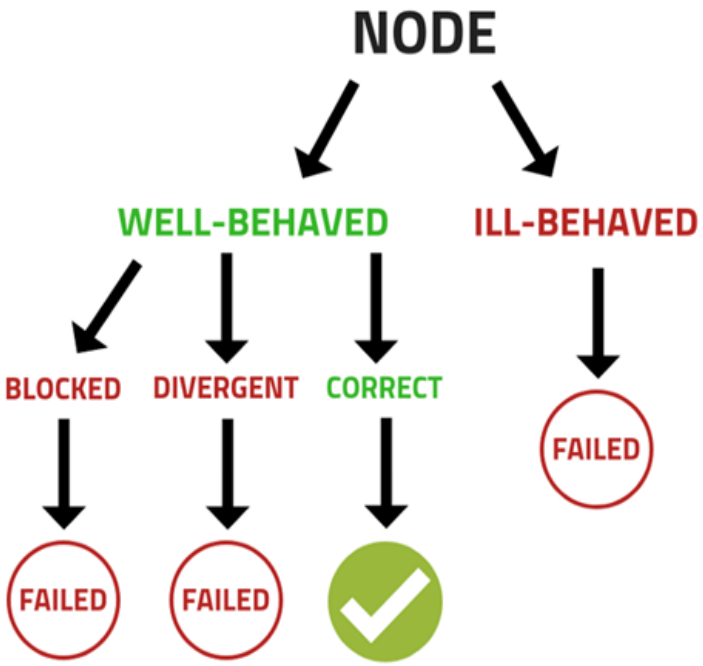


Figure 2.13: Node behavior types

[Source: [59]]

Ill-behaved node

Ill-behaved nodes are nodes that don't obey the protocol rules and doesn't care about network's best interest. They don't choose sensible quorum slices, may have crashed and may intentionally send false messages to the other nodes. Ill-behaved nodes suffer Byzantine failure, meaning they can behave arbitrarily

Well-behaved node

Well-behaved nodes are nodes that obey the protocol rules and choose sensible quorum slices. However, a well-behaved node can fail too., There are two ways that well-behaved node can fail:

Blocked node: Well-behaved nodes that wait indefinitely for messages from ill-behaved node, so they can't output a value for agreement

Divergent node: Well-behaved nodes that get false messages from ill-behaved node, so they output a value that diverges from other nodes

In summary the behavior of these types of node will have an effect on the system. As a first example, if every quorum in the network depends on an ill-behaved node (failed node) that is a quorum intersection, the ill-behaved node can send one message to one quorum and another to another quorum which will drive the system into divergent states. This will cause well-behaved node to failed (become a divergent node) and can't guarantee safety. As a second example, suppose each of a node's slices in quorums

contains ill-behaved nodes which might not agree to anything or crash. and These ill-behaved nodes can drive the system into blocked states. This will cause well-behaved node to become failed (blocked node) because the well-behaved node would never be able to hear unanimously from one of its quorums and never make progress

2.3.5.4 Horizon

Horizon is an API server that connects clients with the Stellar ecosystem. Moreover, Horizon also provides a RESTful API for clients to exchange data between application and Stellar network. The Horizon acts as an interface between Stellar and application to communicate together. Horizon provide two types of API library: 1) Libraries maintained by Stellar.org such as JavaScript, Java, Go; 2) Community-maintained libraries for interacting with Horizon in other languages such as Python, C#, C++, Scala, Ruby, iOS.

2.3.5.5 Stellar Ledger Structure

2.3.5.5.1 Ledger

The *Ledger* is the state of the distributed database at a given point of time. A ledger can be separated into two main parts, the ledger header and the ledger entries.

i. *Ledger header* consists of these following fields:
ii. *Version:* the protocol version of given ledger
iii. *Previous Ledger Hash:* A hash value from previous ledger. This hash value will form a ledger that chained together.

iv. *SCP value*: a particular output value that come from an agreement between nodes in the network. This value also stored in these 3 fields

v. *Transaction set hash*: Hash of the transaction set that was applied to the *previous ledger.* Transaction set structure consists of previousLedgerHash (Hash value from previous ledger) and Transaction envelope (Transaction is ready to be signed, the transaction object is wrapped in an object called a Transaction envelope).

- *Close time*: Time that ledger closed. UNIX timestamp
- *Upgrades:* How the network adjust itself and moves to a new protocol version
- *Transaction set result hash*: Hash of the results of applying the transaction set.
- *Bucket list hash:* Hash of all object in this current ledger or hash of the ledger state.
- *Ledger sequence:* number of sequences of this ledger. This number will be incrementing every time the ledger is close

v. *Total coins:* total number of lumens in existence

vi. *Fee pool:* The overall fee that has been paid. This number also added to the inflation pool and reset to 0 the next time inflation runs. Inflation operation is an operation that distributes lumens that happen in every week. [40]

vii. *Inflation sequence:* The sequence number of Inflation has been run.

viii. *ID pool:* The last used global ID. These IDs are used for generating objects.

ix. *Maximum Number of Transactions:* The maximum number that validator can put in to this ledger

x. *Base fee:* This field use to define the minimum fee that user needs to pay for each transaction

xi. *Base reserve:* number that use to calculate a minimum balance of lumen. Minimum Balance = (2 + # of entries) * base reserve [41]

xii. *Skip list:* Hash of ledger in the past. This allow user to jump back in time without going through all ledger. [42]

2.3.5.5.2 Transaction operation types

Actions that change things in Stellar, like sending payments, changing your account, or making offers to trade various kinds of currencies, are called *operations*.

Here are the possible operation types:

i. Create Account

ii. Payment

iii. Path Payment

iv. Manage Buy Offer

v. Manage Sell Offer

vi. Create Passive Sell Offer

vii. Set Options

viii. Change Trust

ix. Allow Trust

x. Account Merge

xi. Inflation

xii. Manage Data

xiii. Bump Sequence

Operations within each transaction will be classified into four main types of entries: *1) Account entry; 2) Trustline entry; 3) Offer entry; 4) Data entry*.

i. *Account entry.* Account entry represents a transaction that performed by accounts, and accounts control the access rights to balances.

ii. *Trustline entry.* Trustlines are lines of credit the account has given a particular issuer in a specific currency.

iii. *Offer entry.* Offers are entries that an account creates which provide a way to automate simple trading inside the Stellar network.

iv. *Data entry.* Data entries are key value pairs attached to an account. They allow account controllers to attach arbitrary data to their account. It provides a flexible extension point to add application specific data into the ledger.

2.3.5.5.3 Transaction mechanism

i. *Creation (Transaction Creator)* - User create a transaction

ii. *Signing (Transaction Signers.* - The transaction will be wrapped in the form of envelope and send to required signer(s) to sign the transaction

iii. *Submitting (Transaction submitter*) - After signer sign an envelope the Stellar core will check the validity of transaction by using validity rules of transaction. If the envelope of transaction is invalid it will be immediately rejected by Stellar core. If the transaction is invalid the account's sequence will not be consumed and the fee will not be charged.

iv. *Propagating (Validator)* – Once Stellar-core sure that transaction is valid, this transaction will be flooded into the network.

v. *Crafting a candidate transaction set (Validator*) – At this stage, all validator will take all valid transactions that it has received since the last ledger close and collect them to create a transaction set. For the Later transactions are put aside for the next ledger close.

2.3.5.5.4 Validity of Transaction

i. Source Account – Source Account must exist on the ledger

ii. Fee – Number of fees must be greater or equal to the network minimum base fee.

iii. Sequence Number – Sequence number must be *1* greater than the sequence number stored in the source account. This use to prevent user to perform a double spend attack.

iv. List of Operations – operation must be valid in the current protocol network. Operation parameter must be in a valid form.

v. List of Signatures – The signature must be in a valid form '[Network Name]; [Month of Creation] [Year of Creation]'.

vi. Memo – The memo must be a valid type

vii. Time Bounds – The summited transaction must be in set time bound of the transaction or else the system will consider this transaction is invalid

viii. *Nominating a transaction (Validator)* – Once a candidate transaction set created, the set is nominated to the network.

ix. *Stellar Consensus Protocol (SCP)* – determines the final transaction set (Validator Network) – After the validator nominate the transaction set, SCP will select only one set of transaction to apply.

x. *Transaction apply order is determined (Validator Network)* – Once the transaction set is determined, the apply order will be computed. This will both shuffles the order of the set-in order to create uncertainty for competing transactions, and maintains the order of sequence numbers for multiple transactions per account.

xi. *Fees are collected* – all fees are collected from all transactions in transaction set.

xii. *Application (Validator)* – Each transaction will be applied by order previously determined. Account sequence will be consumed by *1*, transaction's validity is checked again, and operation in transaction will be applied.

xiii. *Protocol Upgrades (Validator)* – upgrades are if an upgrade took place

2.3.6 HTTP Protocol

WWW (World Wide Web) is all about communication between client and server which is possible via HTTP request and HTTP response. Client are most often web browsers (Chrome, Firefox). Servers are computer that hosting the website data or applications that generate such data [26]. HTTP (Hypertext Transfer Protocol) is a protocol that allow clients to send various requests to the server via **HTTP requests** and the server will respond to the request via **HTTP response** (the response contain hypertext data). For example, when client wants to open a web page it will send an HTTP request to the server then the server will send back the HTTP response, in this case the requested web page to the client [27].

HTTP is a stateless protocol. Being a stateless protocol means each request-response cycle is unrelated to other cycles [28]. The server doesn't have any representation of a continuing interaction or session

When a client sends an HTTP request to the server it also has a method to determine its intention. So, type of request method determine what operation will get executed. These are some of the basic HTTP request methods

GET: getting data

POST: creating data

PUT: updating data

DELETE: deleting data

To clarify the confusion, HTML (Hypertext Markup Language) and HTTP are two different things but related. **HTML is a language** that marks up normal text and converts it to hypertext. Hypertext is a text that contains links or references to other documents or resources. When a client computer receives an HTML document from a server, the browser

on the client will use the HTML markup to determine how it should display the information.

In contrast**, HTTP is a protocol** that tells client and server how to handle the data transmission.[29] HTTP identifies desired resources on the server and transmits them to the client, but does not control how the information in the resources is displayed (rendered).

2.3.7 RESTful Service

Representational State Transfer (REST) is an architectural style of network systems. REST is a set of design constraints and it uses a stateless communication protocol for building HTTP web services. In REST architectural style, a REST resource is the data on which client want to perform operations. This resource can be accessed by using Uniform Resource Identifier (URI) of the HTTP request. The type of HTTP request method will determine what kind of operation (GET, POST, PUT, DELETE) will be performed to the resource. Then the response from the server is a confirmed result of the command that client sent with the HTTP request URI. The following principles define REST constraints:

- *Resource identification through URI*: all resources from the server can access by using only URI
- *Client-server*: client and server application must be able to evolve separately and not depend on each other
- *Stateless Protocol*: all interaction between client and server must remain stateless
- *Cacheable*: caching shall be applied to resources when possible. If a client request has access to the cache response, it can avoid repeating the same request and use the cache copy instead. This can help relieve some of the server work
- *Layered system*: there might be some further system layers at work in the system. So, there can be more servers in the middle and some of the servers might provide for example, a caching layer

- *Code on demand (optional)*: REST often uses formats like XML or JSON to represent a resource state to client. But if you need to return executable code for some reason, you also can do that.

A RESTful Service means it must follow all these REST constraints [30]

2.3.8 Web Application

A web application (Web app) is an application program that executes on a server and allows a client to gain access to it over the internet via a web browser. Web applications don't need to be downloaded because clients can access them through the network [31].

These are some benefits of a web app

- Multiple clients can access the same version of an application
- Web app don't need to be installed
- Can access from various platforms for example, desktop, mobile

Web applications can be structured as one tier, two tier, three tier and N-tier architectures. The definition of the application architecture is based on the existence of and location of different functional layers:"

- *Presentation layer*: the main functionality in this layer is to handle formatting of output and delivery of input data to the application layer for further processing. Because of this, Presentation layer can relieve some of the concerns in application layer regarding syntax difference.
- *Application layer or Business Logic Layer*: Once the Presentation layer passes down the information to this layer, Application layer then interact with Database layer and sends required information back to the Presentation layer. This layer acts as a mediator between Presentation layer and Data layer. Simply put, it performs the computational operations of the application

- *Data* layer: The data will be stored in this layer. It contains methods that connects the database and performs required actions such as insert, update, delete, read. Simply put, it is to share and retrieve data
- *There are four common types of web application architecture*
- *One Tier Application or Standalone Application*: The one tier architecture has all the layers contained in one tier which is all Client tier. The data is stored in the local system.
- *Two Tier Application or Client-Server Application*: The two-tier architecture is divided into two parts which is Client tier, Database tier. Client tier will handle both Presentation layer and Application layer. Database tier will handle Data layer. There will be a communication that takes place between Client tier and Database tier. Client tier will send request to the Database tier, then Database tier will process the request and sends back the requested data back to Client tier
- *Three Tier Application:* The three tier is divided into three parts which is Client tier, Application tier, Data tier. Client tier handles Presentation layer, Application tier handles Application layer, and Database tier handles Data layer
- *N Tier Application:* N tier application or distributed application is similar to three tier architecture but the number of application servers in application tier is increased and represented in individual tiers so that business logic may be distributed [32]

2.3.9 Development tools

2.3.9.1 JavaScript

JavaScript is a programming language for the web and supported by most browsers. It is used to provide a more user-friendly experience for browsers. JavaScript supports many modern user interface features including interactive maps, animated graphics, etc. In addition, there are two ways in JavaScript that will

make interaction smoother, faster, and more interactive. One is that, JavaScript can change the organization of the page (the DOM, or Document Object Model). The other is the ability to send an HTTP request to the server and get specific data, that we want to show in a page, and get the data back without having to re-render the whole page.

2.3.9.2 Hypertext Markup Language (HTML)

It is a markup language that is used to give structure and meaning to web content, for example, defining paragraphs, headings, etc. HTML is in the first layer of standard web technologies.

2.3.9.3 Cascading Style Sheets (CSS)

It is a language that describes how HTML are to be displayed on screen, for example, setting background colors and font. CSS is in the second layer of standard web technologies. [33]

2.3.9.4 Java

Java is an Object-Oriented Programming Language. It means we can organize a software as a combination of objects.

Notable features of Java:

- *Write once, run anywhere*: the compile code can run on all platforms because compilation happens in byte code. Byte code is platform independent which can be run on any machine
- *Object Oriented:* Since everything is an object, we can inherit attribute and methods from another class. This feature aim to provide reusability in your application for example, we can inherit class “Car” that has the attributes and methods from base class “Vehicle” which in “Vehicle” every vehicle can start and stop its

engine. Since car is also a vehicle, we can just inherit the base class and add more specific attributes like car has 4 wheels to that class instead of building it from scratch.

- *Robust:* highly structured error handling by emphasizing only on compile time error checking and run time checking and there is automatic garbage collection in Java which will help get rid of the objects that are not being use
- *Distributed:* Java was built to be used and distributed environment of the Internet
- *Native internationalization*: Internet has become a global network and that means an increase demand for global software. Java uses Unicode by default, Unicode is a character encoding standard. It is capable of representing almost every well-known character in multiple languages. So, we can encode foreign characters and display them properly and clear to any computer what the text should look like
- Can be used for client-side, server-side, mobile and embedded applications.

For our project, we will use Java to develop our Spring Boot back-end

2.3.9.5 TypeScript

TypeScript is a programming language developed by Microsoft; it is a superset of JavaScript that compiles to plain JavaScript. TypeScript is basically JavaScript plus a few new features. The main benefit of TypeScript is it offers the ability to add static typing. Static typing will check the types and look for errors before running the program. In contrast JavaScript naturally uses dynamic typing.

Dynamic typing will check the types and look for errors while running the program. For example, suppose we have a function that accepts an argument of a number and then adds 5 to it. If we pass an argument like “hello” which is of type String, JavaScript will see that we try to add String and number together then it will convert the number to its equivalent which is String and return it as if no error has happened even though the function is not working as intended. We will get a return value as “hello5” which is of type String. With TypeScript which offers static typing, if we call the above function, we will catch the error instantly. The types are checked in order to prevent assignment of invalid values in terms of their types, which might allow us to save more time debugging error. The benefit of static typing is optional, if you are already familiar with dynamic typing you can also use it as well.[39]

2.3.9.6 Angular

Angular, supported by Google, is an open source software and one of the most popular TypeScript-based frameworks. It is used for developing front-end mobile and desktop web applications. You can think of a framework like a software that has already been developed to help you create a web application. So, the intention of a framework is for you to don’t have to build everything from scratch. It is basically a reusable code that provide you with a template and tools that you need for building a web application. Angular is a component-based framework, inside an Angular web application, our website can divide up into pieces that called “Component” and each one acts as its own isolated entity. A component includes a template (HTML file) that gets rendered into the things you see on the screen, Style sheet (CSS file) for styling the template, and the logic part of the component (TypeScript file). Angular 8 is the latest version and supports TypeScript 3.4 which is required to run Angular 8 project [35]. The most notable

feature which is two-way data-binding, is the synchronization between view and model. The view is the HTML container which an application is displayed. The model is a collection of data that an application need. This binding can help developers save time and reduce workload [36]. The one we are using in this project is Angular8.

Benefits

- *Single Page Applications:* Unlike other web application where when you click a link and you have to wait for it to load, in an Angular web application will give an experience like the web application reside in a single page.
- *Simplified Unit-Testing*: By separating each element in a page to components make unit testing much easier
- *Client-side routing*: Angular use client-side routing rather than server-side routing.[37]
- *Angular Command Line Interface:* In short Angular CLI is a tool that use to initial, develop, and maintain Angular applications. We can easily start a new Angular project by simply typing in a command “ng new [project name]” or create a component by simply typing in command “ng generate component [component name]”

2.3.9.7 Spring Boot

Spring Boot is an open source Java-based standalone framework, meaning that in order to deploy an application the developer does not need to deploy it on a web server or in any special environment due to the fact that Spring boot already has Tomcat web server embedded inside of it. All a developer needs to do is give out the run command, and it will start. Spring boot will be used as a server-side component of this project.

Spring Boot offers a variety of benefits such as, it reduces lots of development time and increases productivity.

2.3.9.8 PostgreSQL

PostgreSQL is a free and open source object-relational database management system (ORDMS). It is designed to handle various kinds of platforms such as Mac OS X, Solaris and Windows. In addition, PostgreSQL has been around for nearly two decades and that it supports standard SQL. PostgreSQL also has an active community to support when other users need help. Moreover, PostgreSQL provides less cost for maintaining due to its stability compared with other database management systems.

2.3.9.9 Docker

Docker is the OS-level virtualization platform service that allows applications to be packaged with all the dependencies installed and run wherever wanted. The applications and its dependencies will be packaged inside a container. A *container* is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.[63]

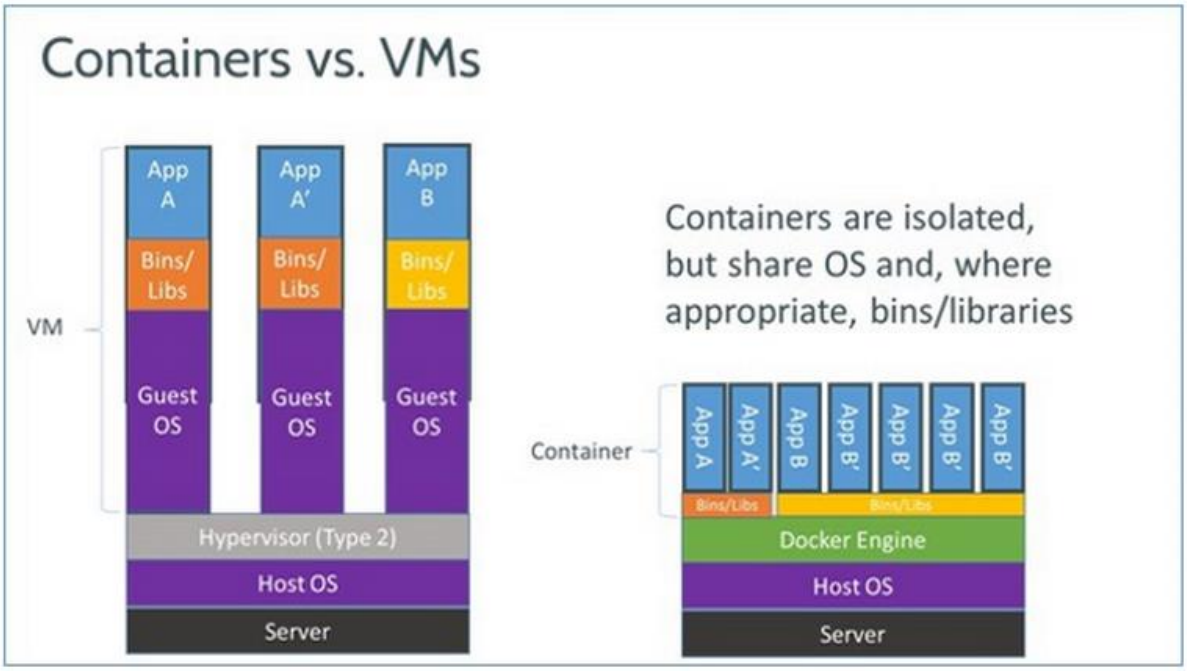


Figure 2.14: Docker vs Virtual Machine [62]

From figure 2.14, it illustrates the architecture of Docker container versus other virtual machines. As you can the Docker container use shared operating system which mean this is much more efficient and save a lot of space than the virtual machine.

Chapter 3

Design and Methodology

3.1 System Requirement and System Constraint

3.1.1 System requirements

- The system must be able to display the full history of his or her transactions to an authorized user
- The system must preserve the anonymity of donors if the donor requests this
- The system must allow donors to send money to beneficiaries
- The system must allow beneficiaries to create and manage campaigns requesting money
- The system must distinguish between (but allow both) government-registered charities and individual or non-registered beneficiaries

3.1.2 System constraint

- The system only available via website (no app)
- For Version 1, the version will allow only transaction in lumens

3.2 Use cases

3.2.1 Use case diagram

Figure 3.1: Overview use case diagram

Figure 3.1 displays all use cases that each actor can do with the system (this use case diagram is the updated version). These are the main goal of each use case (For detail please see Appendix A)

1. “View Donation history” - to view donation history of user transaction.

2. “View donation campaign” – to view campaigns that are currently active.
3. “Search campaign” – to search campaign that are currently active.
4. “Donate Money” – to donate money to the campaign and store transaction in Stellar network.
5. “Login” – to login or access into the system.
6. “Logout” -to logout from the system.
7. “Account setting” – to edit or update account information.
8. “Signup” – to participate as a user inside the system.
9. “Create campaign” – to create campaign in order to receive donation from user.
10. “View owned campaign” – to view campaign that owned by the user.
11. “Approve user identity” – to approve user identity request.
12. “Campaign Management” – to manage all campaign inside the system like inactivate campaign or activate suspended campaign
13. “View fraud report” – to view fraud report that has been send by user.

3.3 Software Detailed Design

3.3.1 Donation

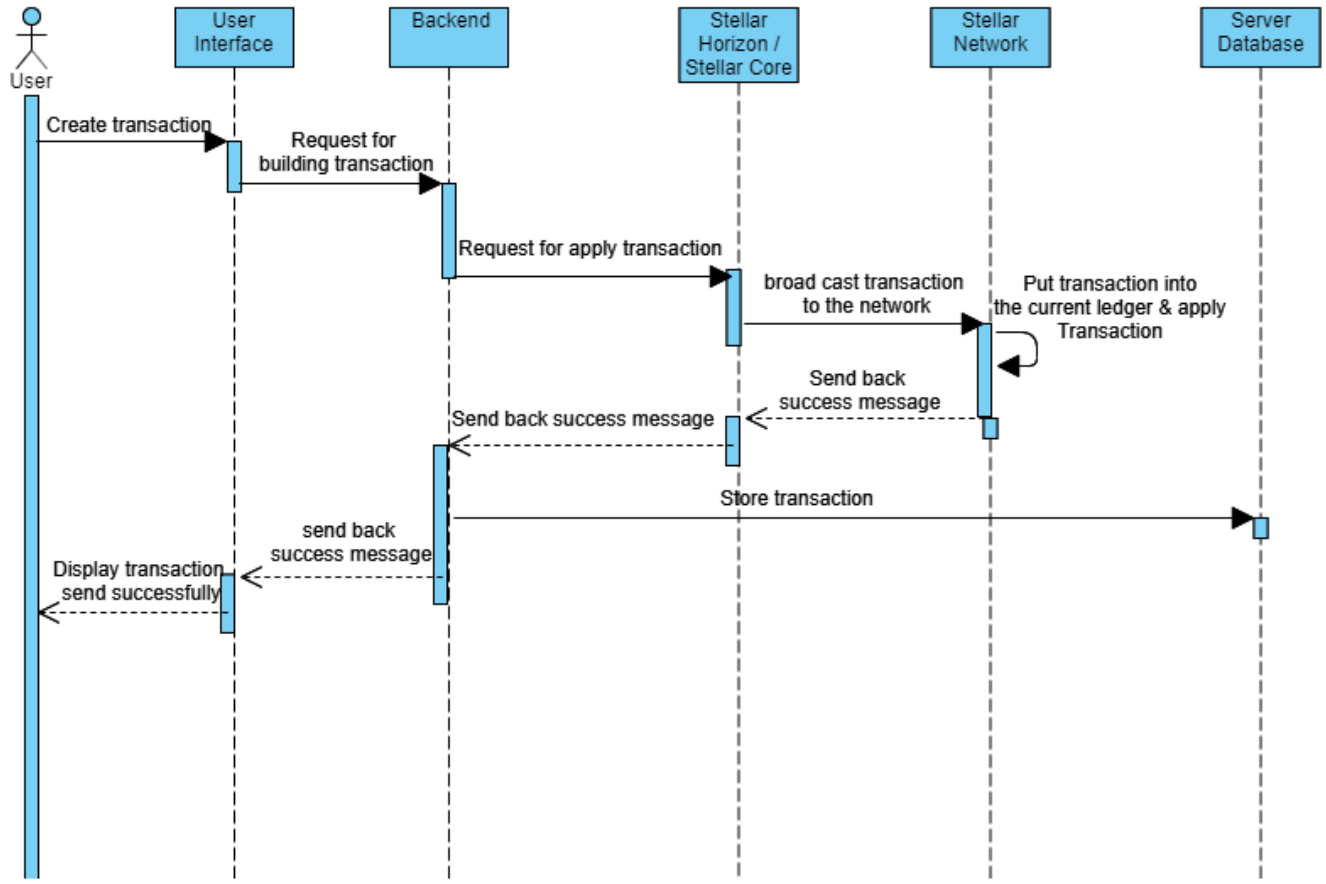


Figure 3.2: Donate money through Stellar network sequence diagram

From Figure 3.2, it displays the process of how system creates a transaction and send through the Stellar network. First, user will create a transaction by filling in a financial information. Second, after user fill in the Backend will create a transaction operation and send to Stellar core through Horizon. After Stellar core received the transaction operation it will broadcast this transaction to the network and the validator node put this transaction into the ledger. After Validator node include this transaction into the ledger, the transaction will be applied. At this step, the sender balance will be deducted and the receiver balance will be increased by Stellar network. After transaction

applied it will send a success message back to our Stellar core. Next, our Stellar core will success message back to our backend through Horizon. After backend received success, it will store transaction data into database send success message back to user interface and display a success message.

3.3.2 Signup

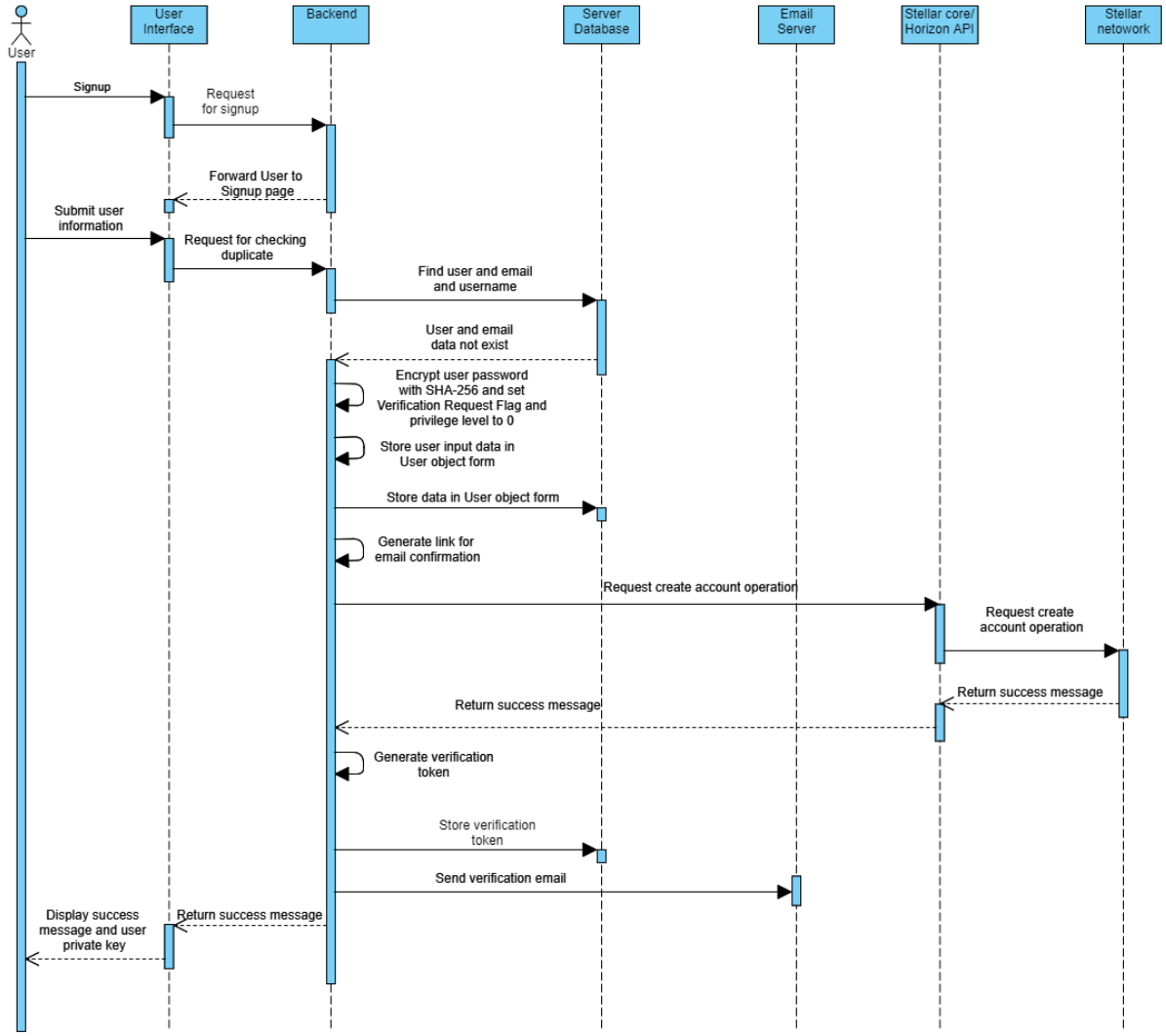


Figure 3.3: Signup sequence diagram

Figure 3.3 displays the process of signup in our system. First user request for signup through user interface. After that user interface will request for backend to forwarding user to signup page. Next, user need to fill in some necessary information like username, password and submit it to our backend. Then, backend will check username

and email in database. If there is no duplicate username or email in database, the system will encrypt user password with BCrypt hash function then set user verification request flag and user privilege level to 0 (0 mean user to need to activate account first). Next, the System will store input information into User object form and store it in server database. After system store user into database, the system will generate a Public and Private key for user and send it to Stellar network in order to create a Stellar account. Next, both of Public and Private key will attached with the verification email. Next after the system generate the key, the system will generate a verification token for user and attached in the email. After that system will generate an email confirmation link and display confirmation message through user interface. After user click on email confirmation link located in email system sent, the system will increase user privilege level to 1 (Allow user to login to the system).

3.3.3 Campaign Transaction history from Stellar

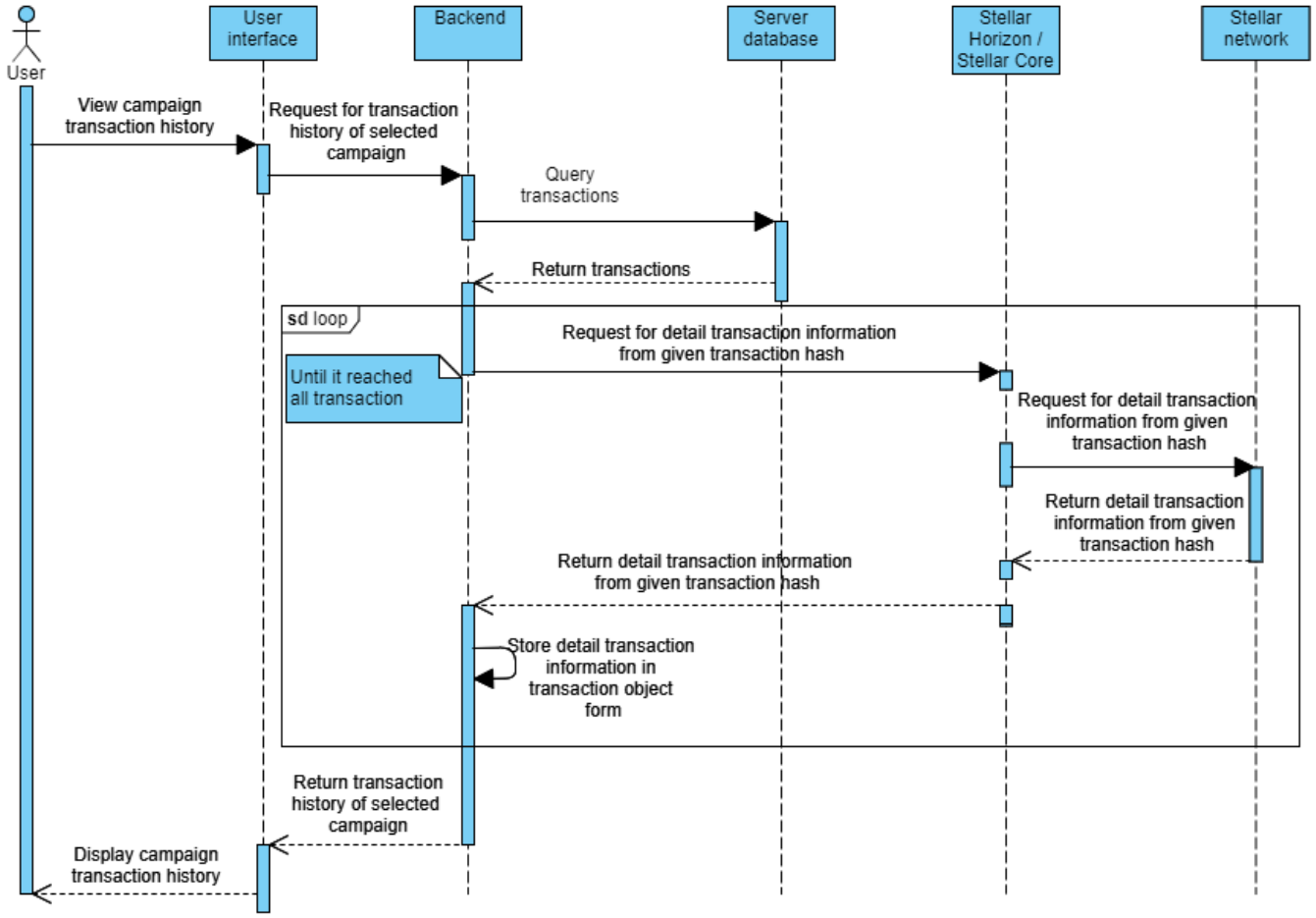


Figure 3.4: Display transaction history

From figure 3.4, it illustrates the process of how system display transaction history. First user will request to view a campaign transaction history. Next System will query all transactions that were sent to this campaign. After system query all history transactions it will start a loop that sends a transaction hash to the Stellar network in order to retrieve a transaction object from Stellar network. The loop will keep running until it reached all transactions. Finally, the system will display every transaction to user.

3.3.4 Create campaign

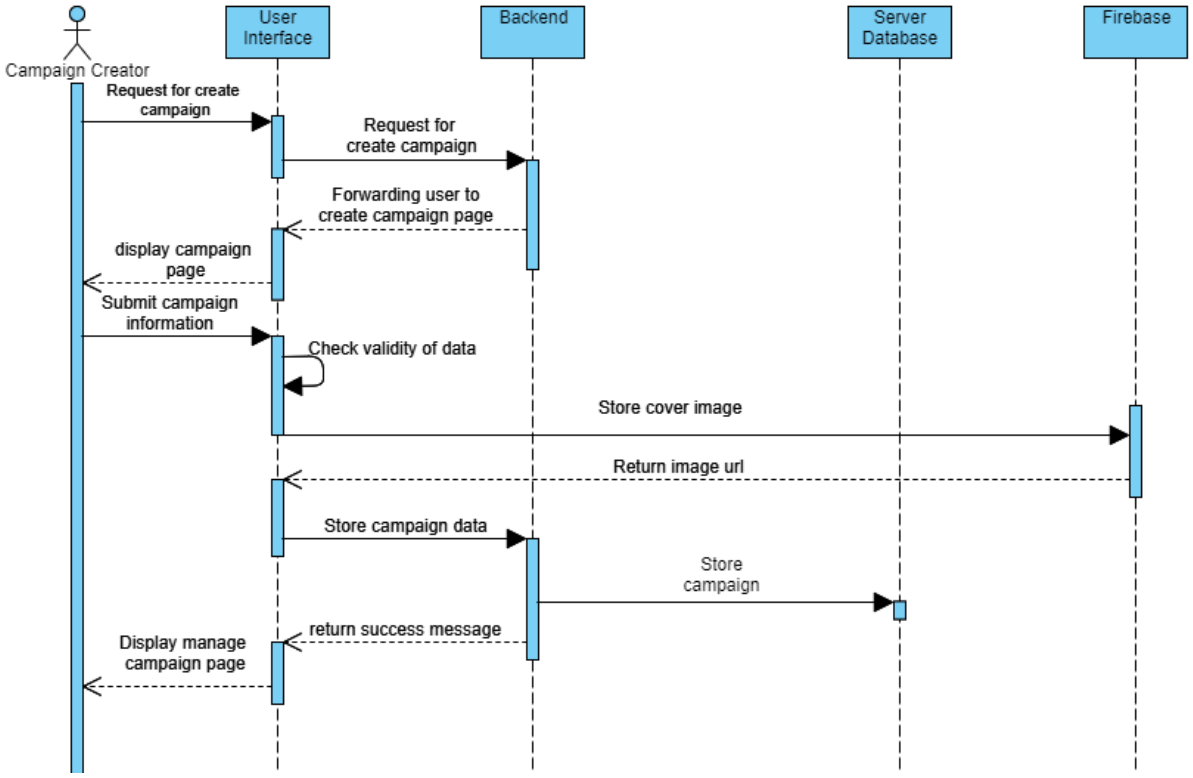


Figure 3.5: Create campaign sequence diagram

From figure 3.5, it displays how create campaign process work. First user will request for create campaign. Next the system will forward user to create campaign. Then user fill out information about the campaign and submit it the system. Next, the system will the validity of data whether user fill in all required field or not. After that the system will store campaign cover image into the firebase then fire base will return the URL path of campaign cover image. Next, after the system received URL path the system will send campaign data to the backend. Backend will store all campaign data into the database and return success message to the UI. Next, after UI received success message from the backend it will forward user to the manage campaign page.

3.4 System architecture

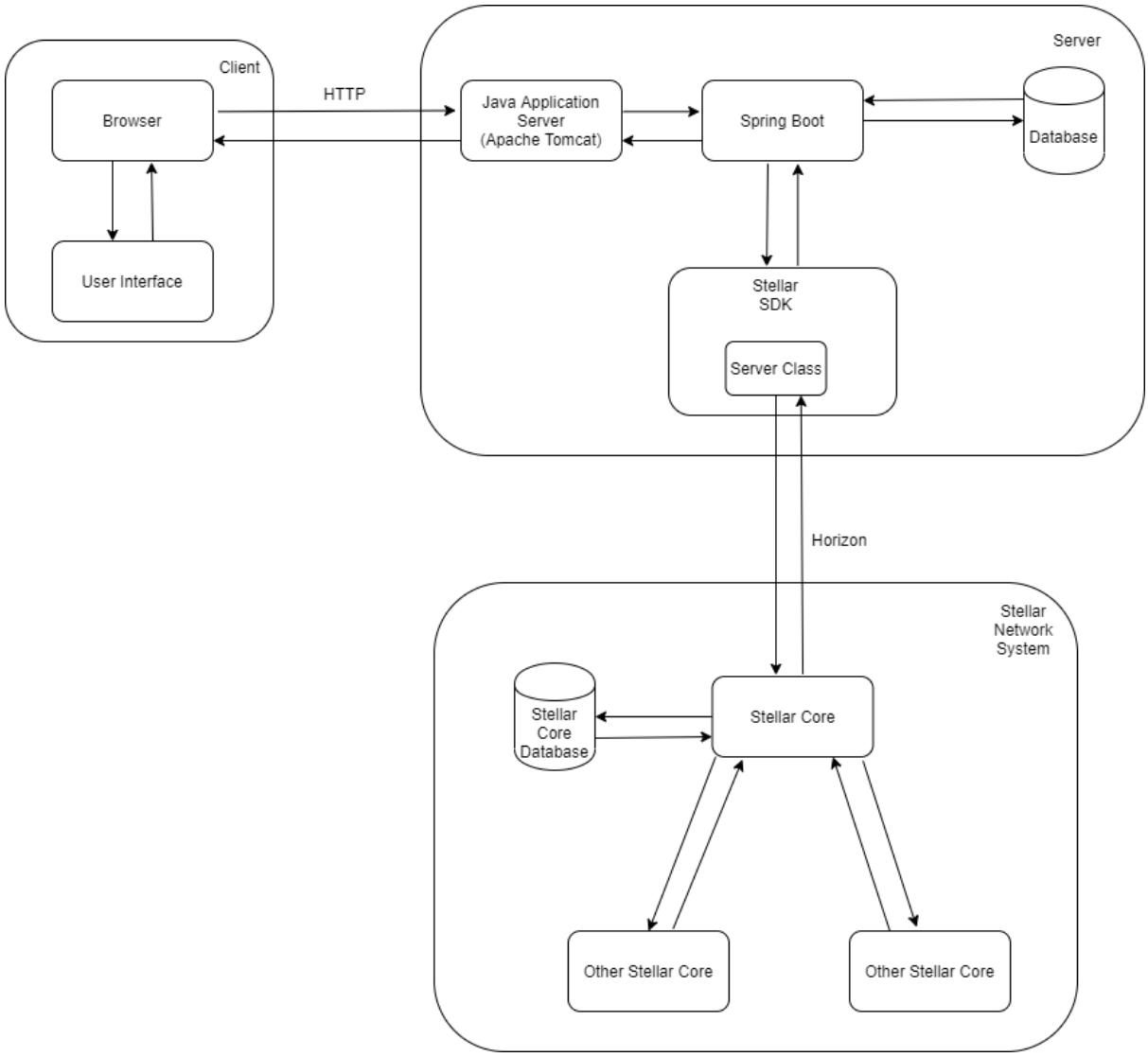


Figure 3.6: System architecture

Figure 3.6 displays an architecture diagram for this platform which can be separated into three main parts. The Client side contain browser and user interface that communicates to server side by HTTP. The server will communicate to Client side by request and response through HTTP with Apache Tomcat. Apache Tomcat is our Java application server and MySQL is our database which is used to store and retrieve data. Spring Boot communicates with Server Class which is provided by Stellar SDK in order to send and retrieve transaction information to a Stellar Network with Horizon API. Stellar Network consists of our Stellar core instance and

other nodes' Stellar Core instances. Stellar Core is used for communicating with peers and the network and store communicated data in Stellar Core database.

3.5 Web page structure

This will show all the route for all the web pages that we have for unverified users

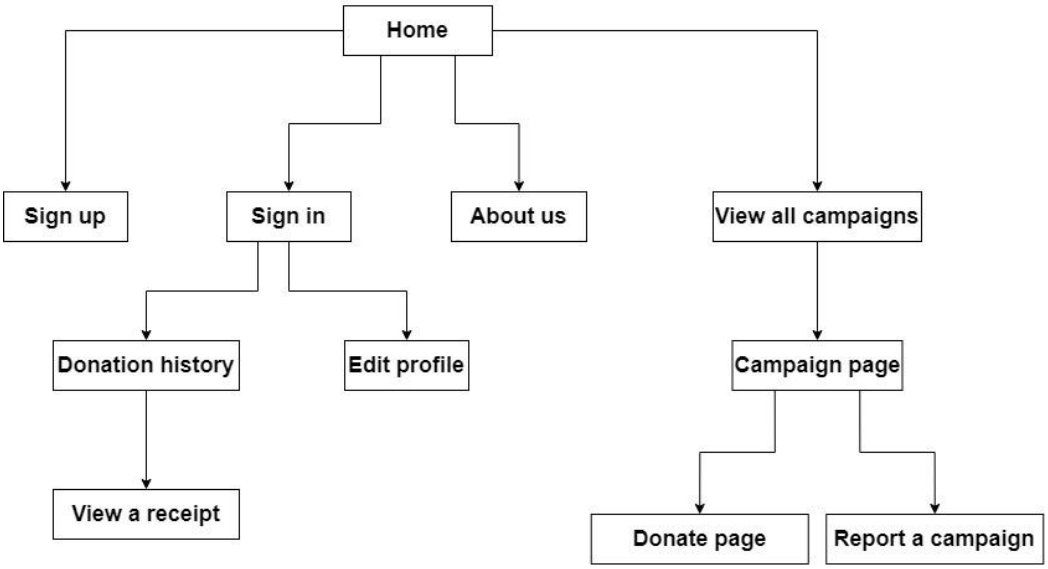


Figure 3.7: Unverified user Interface Page Structure

This will show all the route for all the web pages that we have for verified users (Beneficiary)

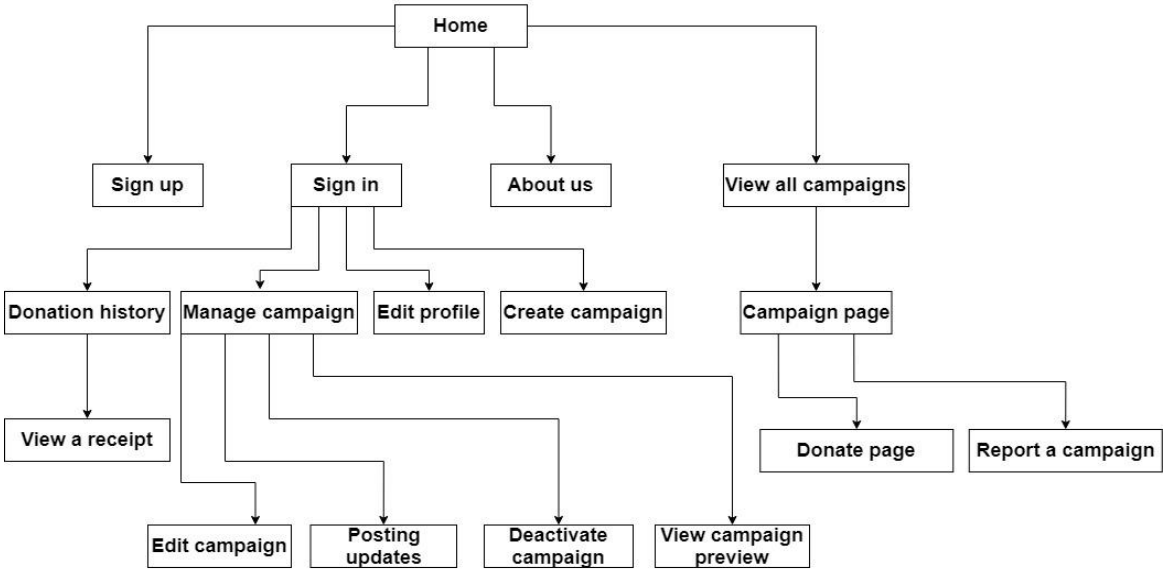


Figure 3.8: Beneficiary Interface Page Structure

This will show all the route for all the web pages that we have for admin

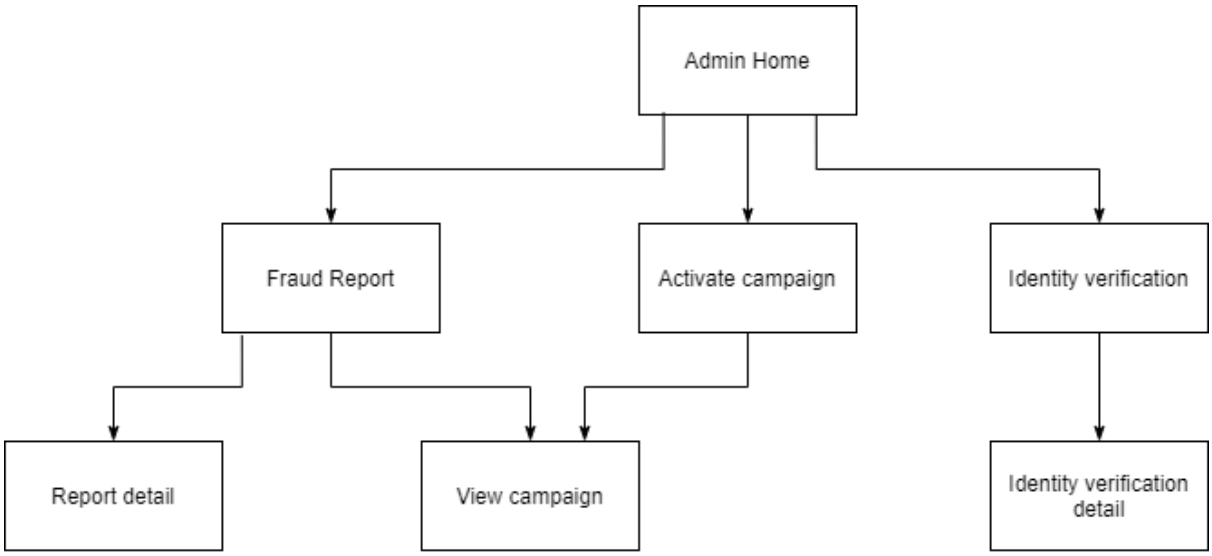


Figure 3.9: Admin Interface Page Structure

For a further detail of each UI please see *Appendix B*

3.6 Database schema

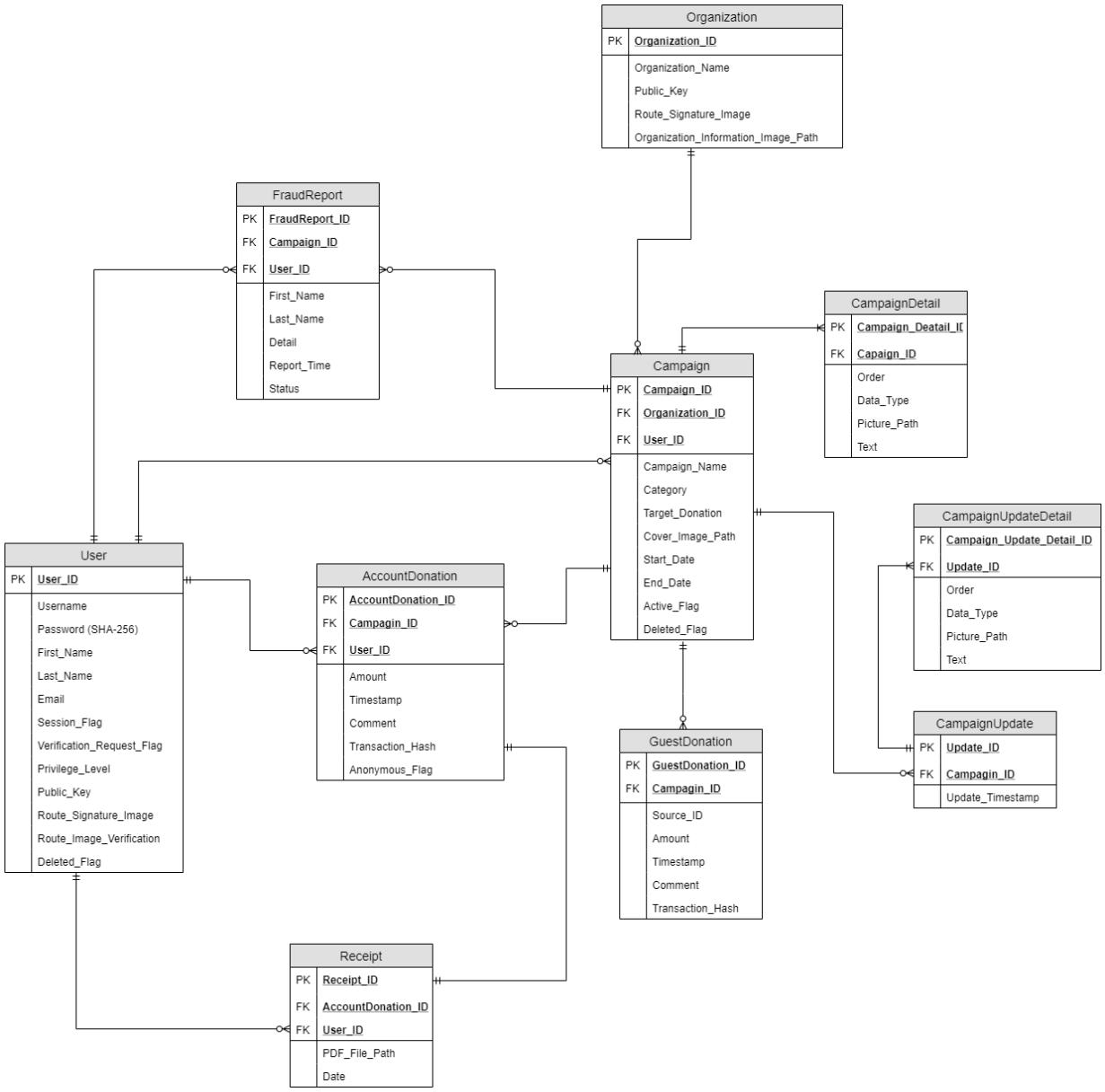


Figure 3.10: Database schema

Figure 3.10 displays the overview of the system database. Overall, the system database consists of 10 tables and each table will be described below. this is the schema as originally designed, and that some changes have been made during implementation (For the final design see *Section 4.2*).

Table 3.1: User table

Entity Name	Attribute	Data Type	Constrain	Definition
User	User_ID	Integer(8)	NOT NULL	Identification number for each user
	Username	Varchar(32)	NOT NULL	Nickname for each user
	Password	Varchar(64)	NOT NULL	Password for each user. Password must be hash with SHA-256(one way hash) in order to keep the password secure.
	Previlege_Level	Integer(1)	NOT NULL	An integer that use to restrict action of each user
	First_Name	Varchar(32)	NOT NULL	First name of each user
	Last_Name	Varchar(32)	NOT NULL	Last name of each user
	Email	Varchar(32)	NOT NULL	Email of each user
	Public_key	Varchar(56)	NULL	A string that act like an address. It uses as an address for other user to transfer money to.
	Session_Flag	Integer(1)	NOT NULL	Flag that use to determine user is logged in to the system or not(0 = sign out, 1 logged in)
	Route_Signature_Image	Varchar(90)	NULL	Route to find image for signature for each user
	Route_Image_Verification	Varchar(90)	NULL	Route to find image for identity verification for each user
	Verification_Request_Flag	Integer(1)	NOT NULL	Verification request flag is used to tell that this user request for identity verification

	Deleted_Flag	Integer(1)	NOT NULL	Flag that set user out of the system instead of delete user from database

Table 3.2: Fraud Report table

Entity Name	Attribute	Data Type	Constrain	Definition
Fraud Report	FraudReport_ID	Integer(8)	NOT NULL	Identification number for each Fraud report
	Campaign_ID	Integer(8)	NOT NULL	ID of campaign ID
	User_ID	Integer(8)	NOT NULL	ID of user that send report
	First_Name	Varchar(32)	NOT NULL	First name of user that send report
	Last_Name	Varchar(32)	NOT NULL	Last name of user that send report
	Detail	Text	NOT NULL	Detail of report
	Report_Time	Timestamp	NOT NULL	Publish report time
	Status	Integer(1)	NOT NULL	An integer that use to tell whether the report is success or not

Table 3.3: Receipt table

Entity Name	Attribute	Data Type	Constrain	Definition
Receipt	Receipt_ID	Integer(8)	NOT NULL	Identification number for receipt
	AccountDonation_ID	Integer(10)	NOT NULL	AccountDonation_ID that user did a transaction

	User_ID	Inter(8)	NOT NULL	User_ID that own this receipt
	PDF_File_Path	Varchar(90)	NOT NULL	Path where the PDF file is located. PDF File is used to store the receipt information
	Date	Timestamp	NOT NULL	Time that this receipt create

Table 3.4 Account Donation table

Entity Name	Attribute	Data Type	Constrain	Definition
AccountDonation	AccountDonation_ID	Integer(10)	NOT NULL	Identification number for transaction that publish by user from this platform
	Campaign_ID	Integer(8)	NOT NULL	Campaign that user create a transaction
	User_ID	Integer(8)	NOT NULL	User that create this transaction
	Amount	Integer(20)	NOT NULL	The amount of money in this transaction
	Timestamp	Timestamp	NOT NULL	Time that campaign creator receive money
	Comment	Text	NULL	A message from the user to the campaign creator
	Transaction_Hash	Varchar(64)	NOT NULL	Hash that can be used to query transaction information from Stellar network
	Anonymous_Flag	Integer(1)	NOT NULL	This flag is used to determine whether user want to donate

				anonymously or non-anonymous

Table 3.5: Campaign table

Entity Name	Attribute	Data Type	Constrain	Definition
Campaign	Campaign_ID	Integer(8)	NOT NULL	Identification number for each campaign
	Organizarion_ID	Inter(8)	NULL	All of the money that donate to this campaign will go to this Organization
	User_ID	Integer(8)	NOT NULL	User who owned the campaign
	Campaign_Name	Varchar(100)	NOT NULL	Name of the campaign
	Target_Donation	Integer(20)	NOT NULL	Target amount of money that campaign creator want to raise
	Cover_Image_path	Varchar(80)	NOT NULL	Path that stored this campaign cover photo
	Start_Date	Timestamp	NOT NULL	Published time of this campaign
	End_Date	Timestamp	NOT NULL	Date that this campaign will be closed
	Active_Flag	Integer(1)	NOT NULL	Flag that use to tell campaign is still active or not
	Deleteed_Flag	Integer(1)	NOT NULL	Flag that use to tell that this campaign is deleted by owner

Table 3.6: Guest Donation table (Donate via third party Stellar wallet)

Entity Name	Attribute	Data Type	Constrain	Definition
Guest Donation	GuestDonation_ID	Integer(8)	NOT NULL	Identification number for each transaction from outside the system
	Campaign_ID	Integer(8)	NOT NULL	Campaign that guests donate to
	Source_ID	Varchar(56)	NOT NULL	Public key of guest who owned the transaction
	Amount	Integer(20)	NOT NULL	Amount of money that guest donate
	Timestamp	Timestamp	NOT NULL	Time that campaign creator receive the money
	Comment	Text	NOT NULL	A message from the user to the campaign creator
	Transaction_Hash	Varchar(64)	NOT NULL	Hash that can be used to query transaction information from Stellar network

Table 3.7: Campaign Detail table

Entity Name	Attribute	Data Type	Constrain	Definition
CampaignDetail	Campaign_Detail_ID	Integer(8)	NOT NULL	Identification number for each campaign detail
	Campaign_ID	Integer(8)	NOT NULL	Campaign that owned this detail
	Order	Integer(3)	NOT NULL	Order for query campaign detail in order to create web page

	Data_Type	Varchar(10)	NOT NULL	Data type of this campaign detail
	Picture_Path	Varchar(90)	NULL	Path of picture for campaign detail
	Text	Text	NULL	Campaign details in text

Table 3.8: Campaign detail update table

Entity Name	Attribute	Data Type	Constrain	Definition
CampaignUpdate Detail	Campaign_Update_Detail_ID	Integer(8)	NOT NULL	Identification number for each campaign update detail
	Update_ID	Integer(8)	NOT NULL	Campaign update that owned this detail
	Order	Integer(3)	NOT NULL	Order for query campaign update detail in order to create web page
	Data_Type	Varchar(10)	NOT NULL	Data type of this campaign update detail
	Picture_Path	Varchar(90)	NULL	Path of picture for campaign update detail
	Text	Text	NULL	Campaign update details in text

Table 3.9: Campaign update table

Entity Name	Attribute	Data Type	Constrain	Definition
CampaignUpdate	Campaign_Update	Integer(8)	NOT NULL	Identification number for each campaign update
	Campaign_ID	Integer(8)	NOT NULL	Campaign that owned this update
	Update_Timestamp	Timestamp	NOT NULL	Updated time

Table 3.10: Organization table

Entity Name	Attribute	Data Type	Constrain	Definition
Organization	Organization_ID	Integer(8)	NOT NULL	Identification number for each organization
	Organization_name	Varchar(100)	NOT NULL	Name of organization
	Public_Key	Varchar(56)	NOT NULL	Public key of this organization which is use to receive donation money
	Route_Signature_Image	Varchar(90)	NOT NULL	Route to find image for signature for each organization
	Organization_Information_Image_Path	Varchar(90)	NOT NULL	Route to find organization information PDF file for each organization

Chapter 4

Results and Discussion

Overview

This chapter provides a result and summary of what we have implemented for final 2020. Topics in this chapter include: 1) Web application UI update; 2) Database; 3) Transaction with Stellar blockchain; 4) Validation plan

4.1 Web application results

Here are pages that we have implemented: 1) Home page; 2) Donate page; 3) Campaign page; 4) Create campaign page; 5) Sign in page; 6) Signup page; 7) Account setting page; 8) Transaction history page; 9) Campaign transaction history page; 10) Admin page.

4.1.1 Home page

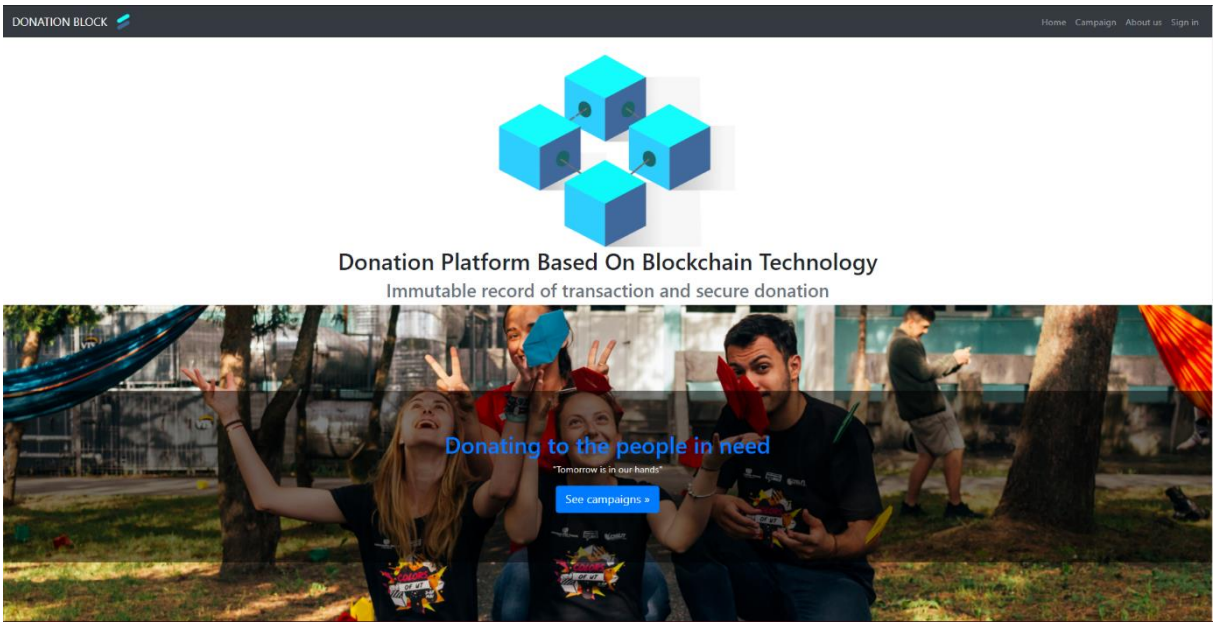


Figure 4.1: Home page overview

As shown in figure 4.1, this page can be separated into 2 main parts which are Navigation bar and See all campaign page navigation button

Figure 4.2: Home page navigation bar

Figure 4.2, shows that there are several options that users can operate with the navigation bar.

1. Home - This button will navigate users back to the home page.
2. Campaign - This button will navigate users to display all campaign page.
3. About us - This button will navigate users to the About page which is used to be describe overview of our web application.
4. Sign in - This button will navigate users to the login page which is used for user authentication.

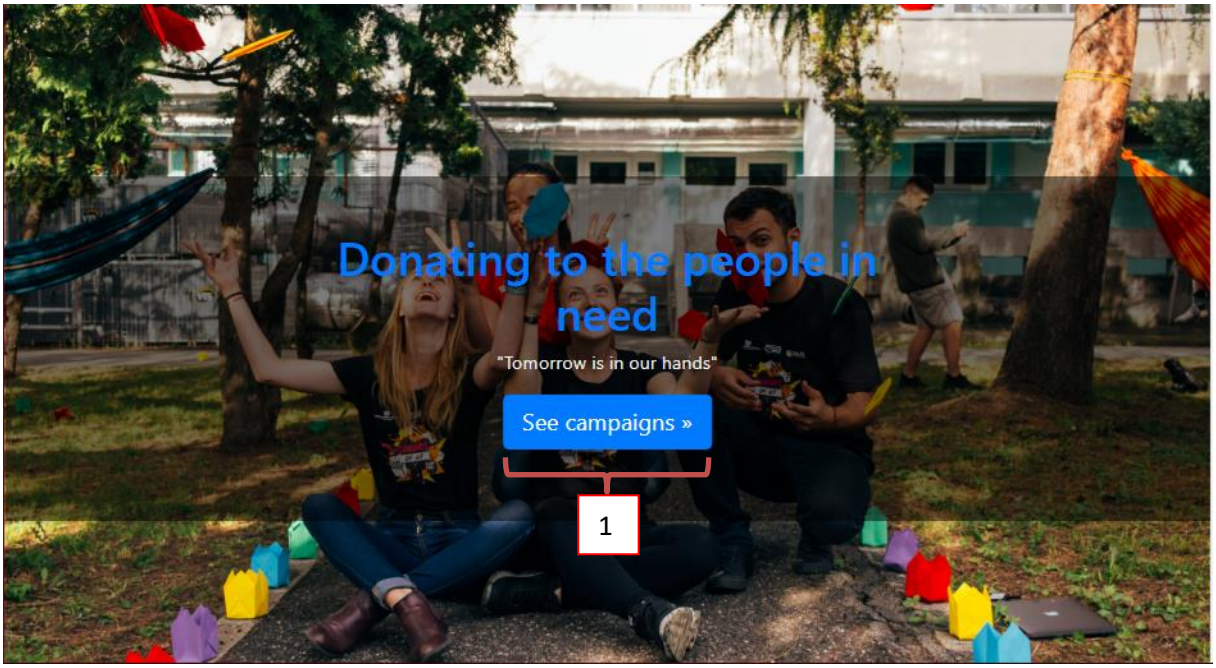


Figure 4.3: Home navigation to all campaign

Figure 4.3 is part of home page which is used for navigate user to page that display all campaign

4.1.2 Donate page

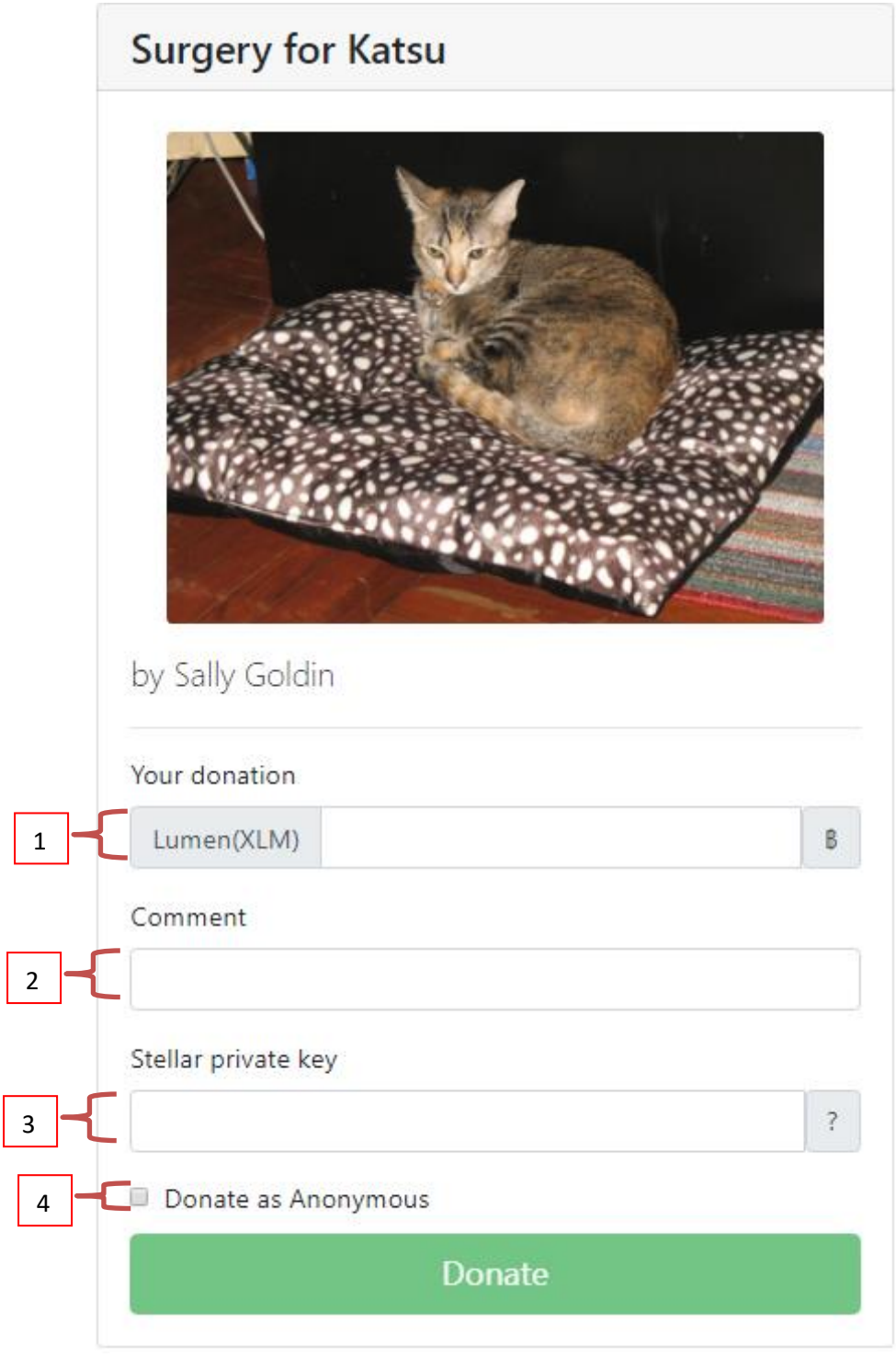


Figure 4.4: Donate form in donate page

Figure 4.4 displays a donation form of the selected campaign where users must specify information about their donation. All form fields must be specified except for the comment field.

1. Your donation - Donor must enter how much they want to donate in Lumen. User can also convert the amount to Baht by click the "฿" on the right
2. Comment - Donor can also choose to leave a comment for the campaign's owner (optional)
3. Stellar private key - We decided to send Private key along with the verification email when they first sign up and if they forgot their Private key there is a reminder for where to look at the right side by hovering the mouse to "?"
4. Donate as Anonymous - If a user wants to donate as anonymous (that is, donate without showing a name), the user can tick the checkbox.

4.1.3 All campaign page

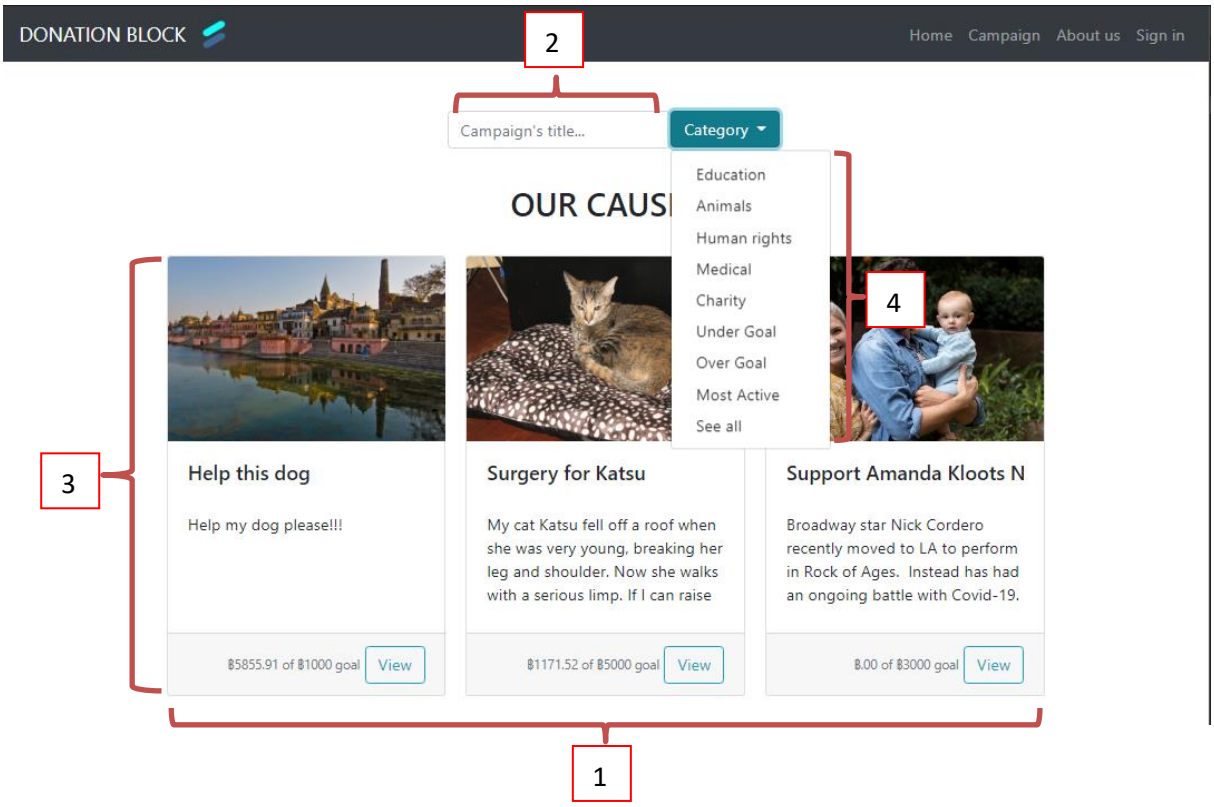


Figure 4.5: View all active campaign page

Figure 4.5 will show all campaigns that all beneficiaries have created.

1. All campaigns - This section will show every campaign from all beneficiaries
2. Search by campaign's title - Allow user to search for all campaigns by enter a campaign's title
3. Campaign card - The card provides brief information about a campaign. It includes a cover image, title, story, how much money it raised, and target donation. If a user is interested in a campaign, they can view full information about the campaign on the Campaign page (Figure 4.6) by simply clicking on the "View" button located at the foot of a card.
4. Search by various category - Allow user to search for all campaigns according to that category, there are 9 categories in total

a. Education: Show campaigns that when first created they select "Education" as their campaign's category
b. Animals: Show campaigns that when first created they select "Animals" as their campaign's category
c. Human rights: Show campaigns that when first created they select "Human rights" as their campaign's category
d. Medical: Show campaigns that when first created they select "Medical" as their campaign's category
e. Charity: Show campaigns that when first created they select "Charity" as their campaign's category
f. Under Goal: Show campaigns that haven't reached its target donation yet
g. Over Goal: Show campaigns that have reached its target donation
h. Most Active: Show campaigns that gets frequent donation, sorting in ascending order
i. See all: Show all campaigns

4.1.4 Campaign page

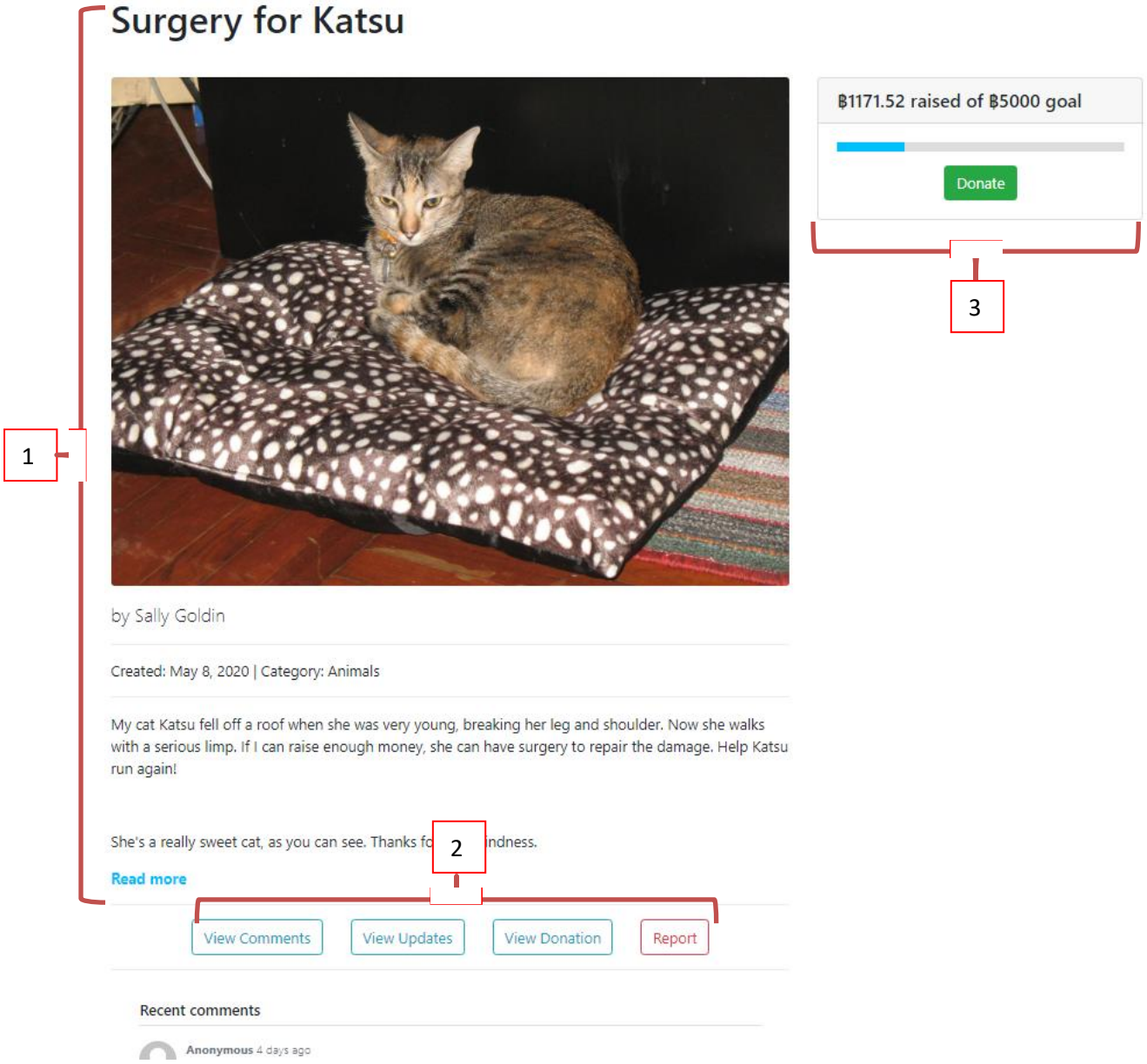


Figure 4.6: Overall campaign page

Figure 4.6 will display the campaign page and elements inside of the page.

1. Detail section - This section is used for describing some necessary parts of each campaign from top to bottom are title, cover picture, campaign's owner, created date, campaign's category and campaign's detail.
2. Navigation section - This section is used for displaying comments, campaign update, donation history of the campaign and filing a report

about the campaign. It will display selected navigation content below this section

3. Donation section - This section is used to show how much money the campaign has raised and the “Donate” button will navigate users to the donation page of this campaign.

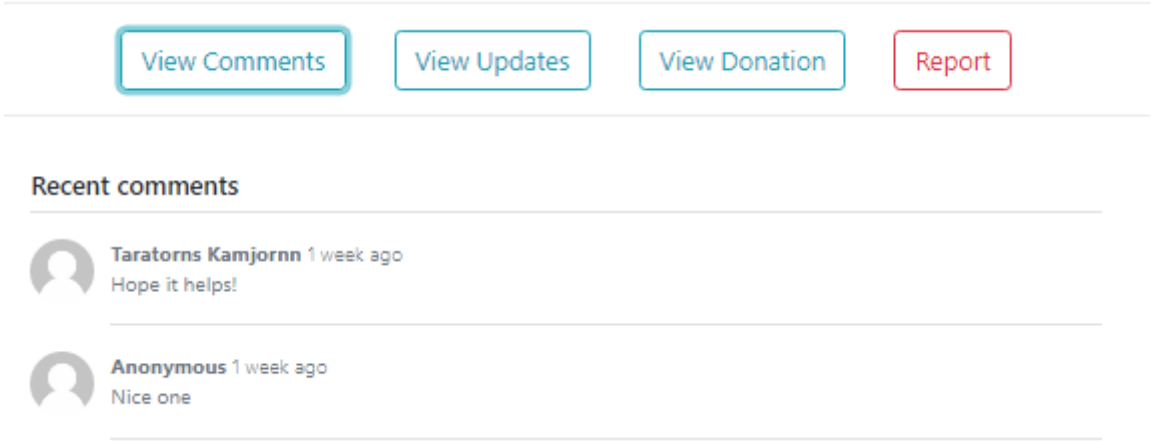


Figure 4.7: View comment section

Figure 4.7 will display comments from donor when user selected “View Comments” in the Navigation section from Figure 4.6

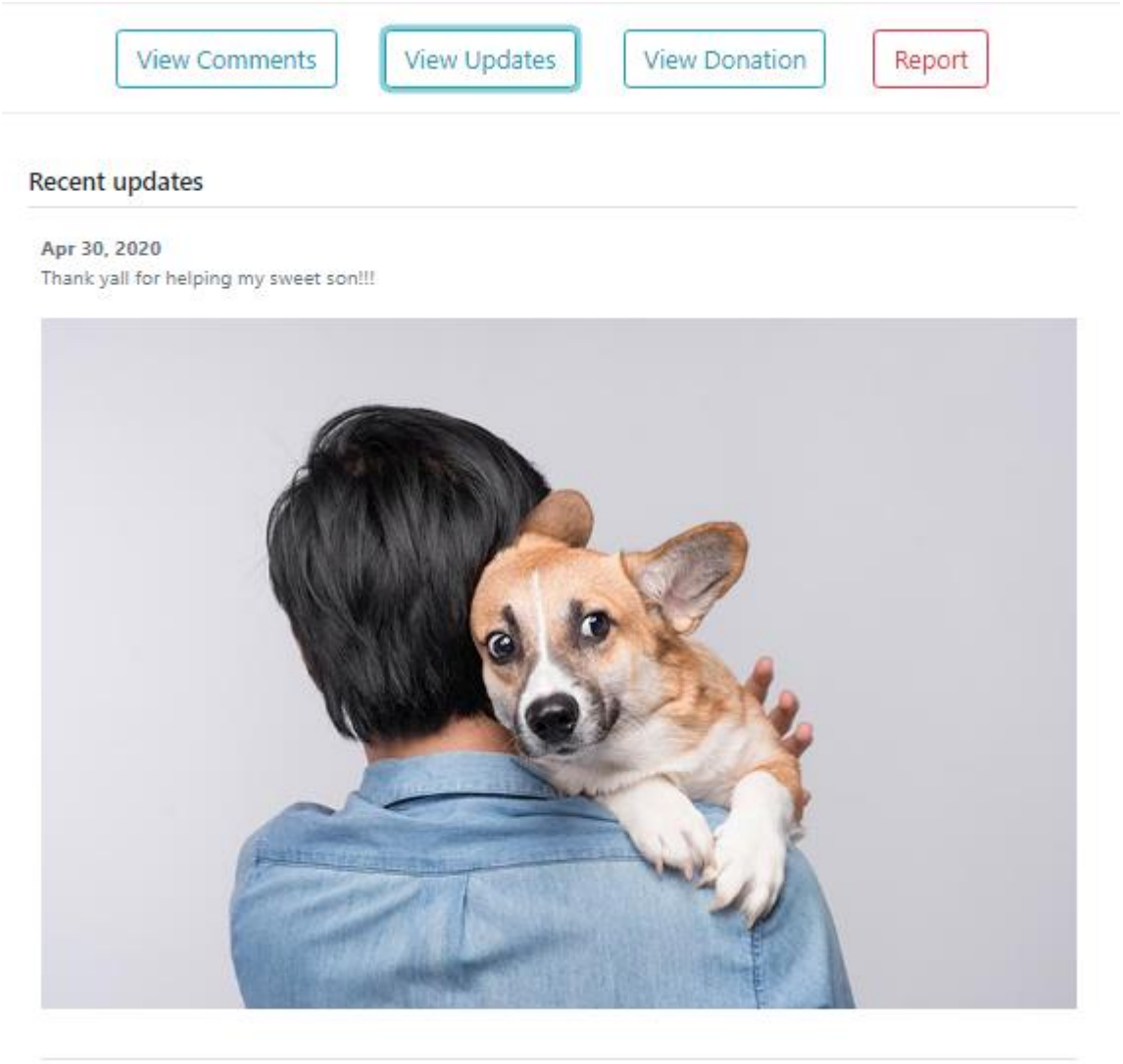


Figure 4.8: View update section

Figure 4.8 will display updates from campaign's owner when user selected "View Updates" in the Navigation section from Figure 4.6

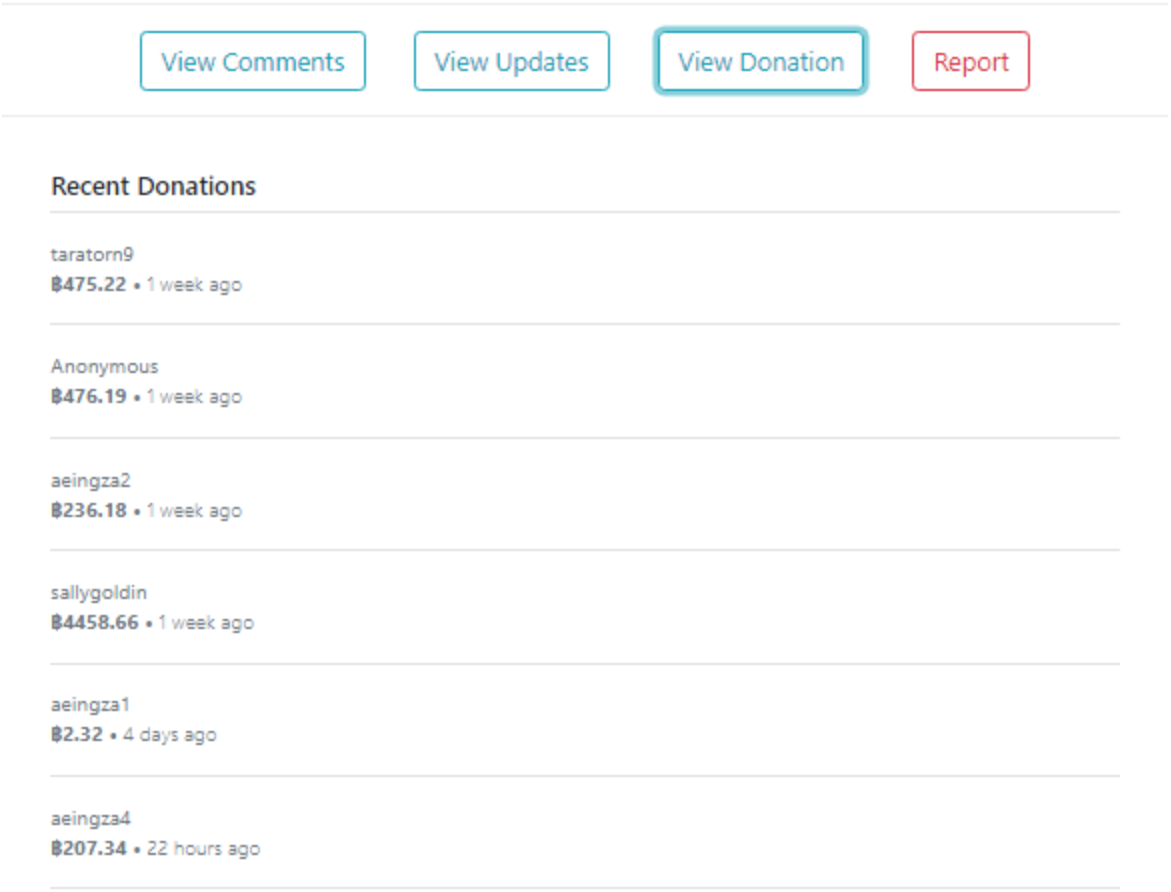


Figure 4.9: View campaign's donation history section

Figure 4.9 will display campaign's donation history when user selected "View Donation" in the Navigation section from Figure 4.6

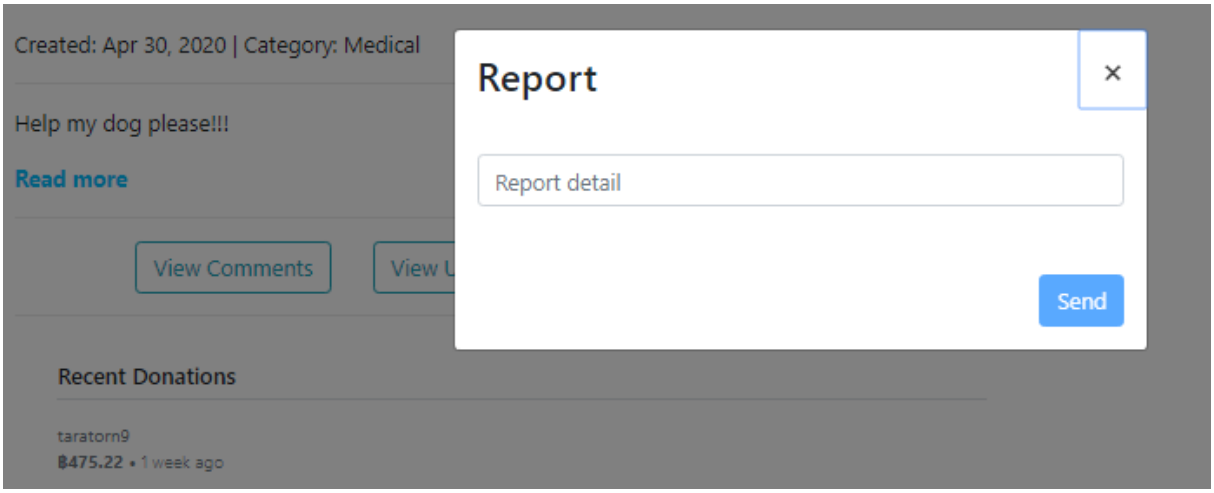


Figure 4.10: Report campaign section

Figure 4.10 will display a textbox for user to fill in detail about suspicious fraud campaigns when user selected “Report” in the Navigation section from Figure 4.6

4.1.5 Create campaign page

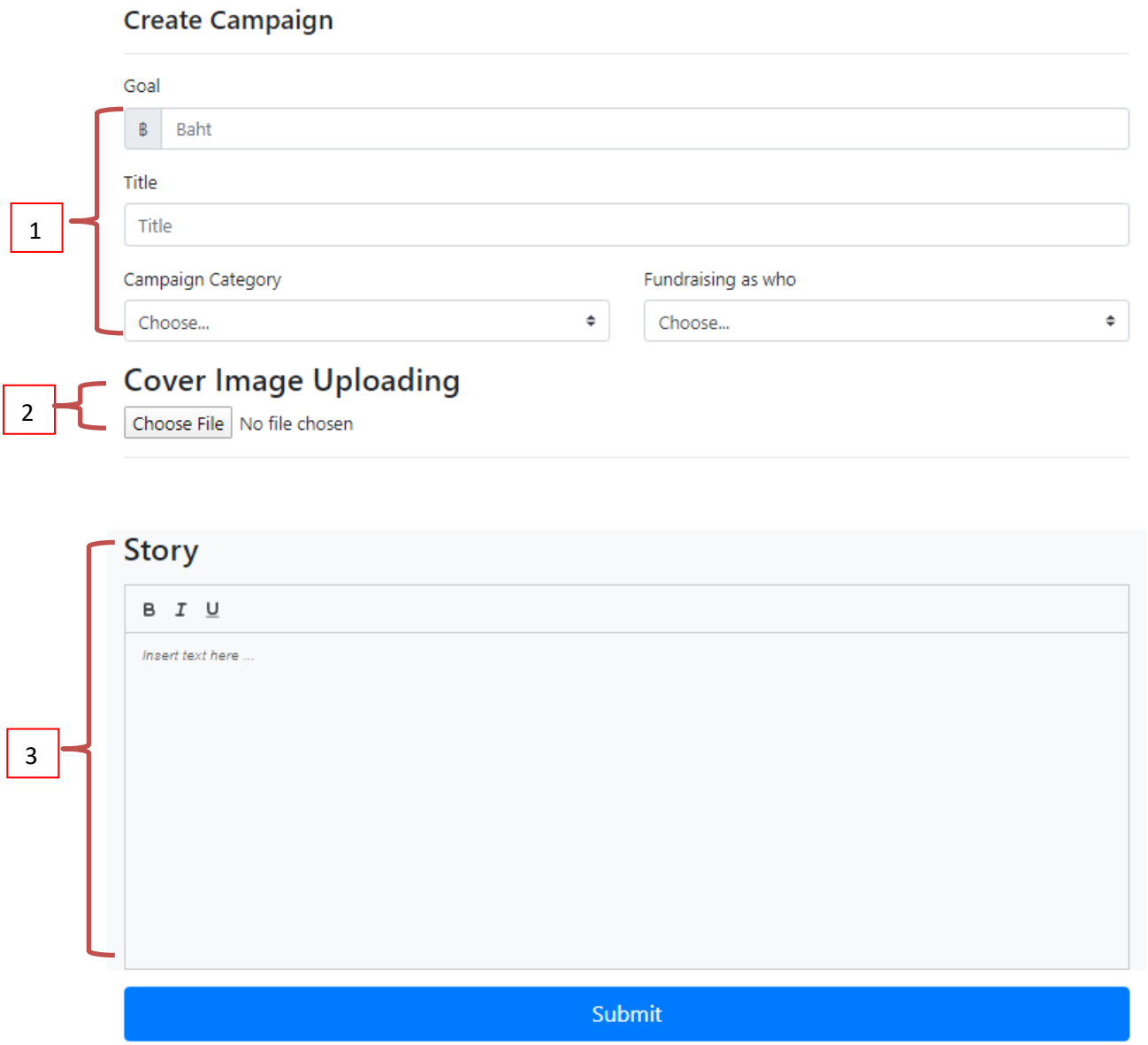


Figure 4.11: Overview of create campaign page

Figure 4.11 illustrates the overview of the create campaign page. shows us the form that collects information users need to specify in order to create a campaign.

1. This field is used to specify some basic information.
2. This part is used for upload cover image for each campaign.
3. This part is a part that users need to write down some more details about their campaign.

4.1.6 Sign in page

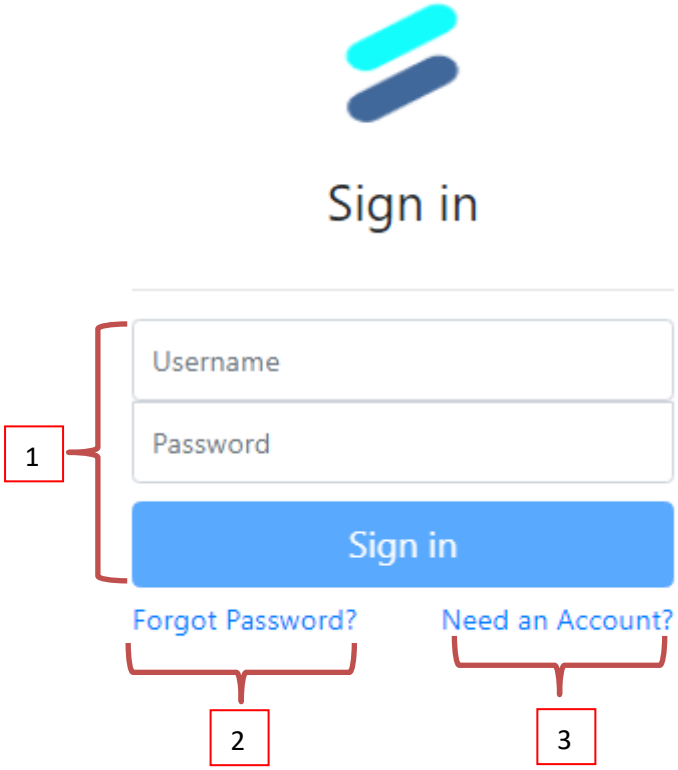


Figure 4.12: Sign in form

Figure 4.12 is the Sign In page of the system.

1. These 2-form fields must be specified in order to login to the system.
2. This section uses for sign up for the new account
3. This is section uses for recover username or password function if users forgot their username or password

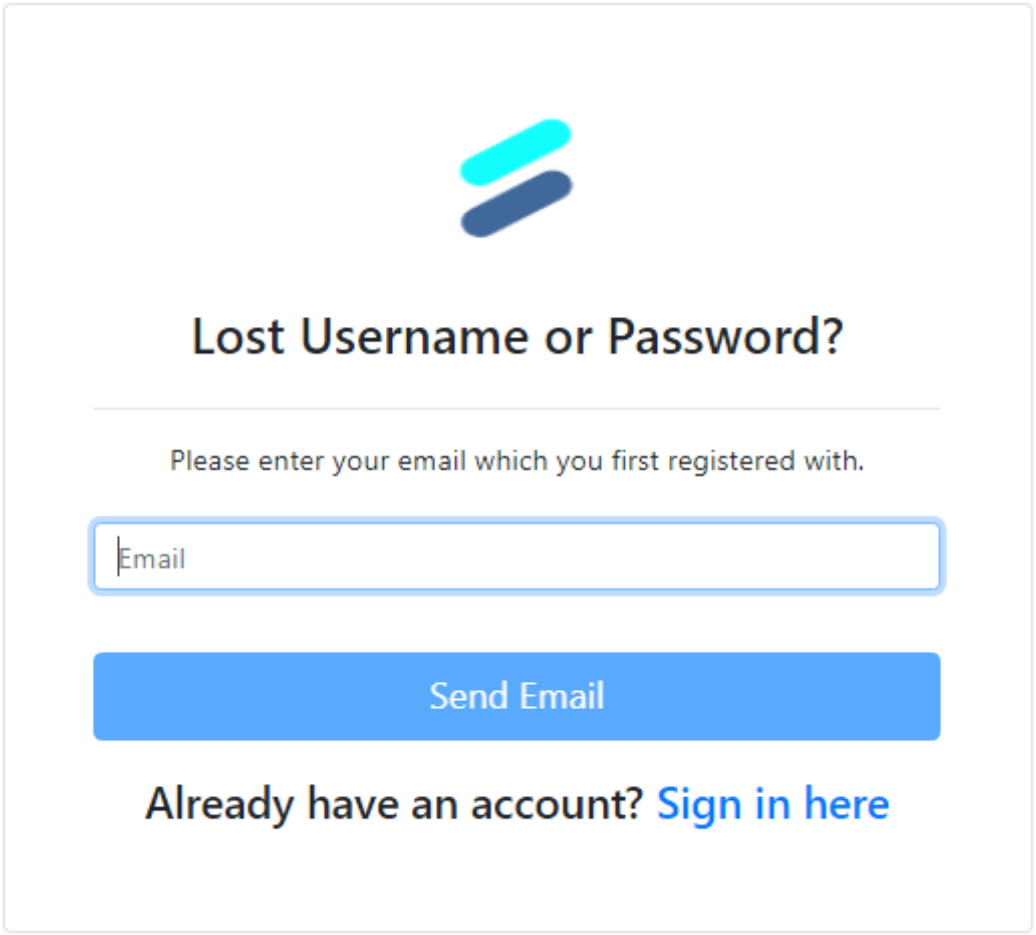


Figure 4.13: Recover username or password form

Figure 4.13 will display the recover username or password function. There will be a form for users to enter their registered email. After they enter their registered email then click “Send Email”, our website will send an email that contains their username and password to the provided address.

4.1.7 Signup page

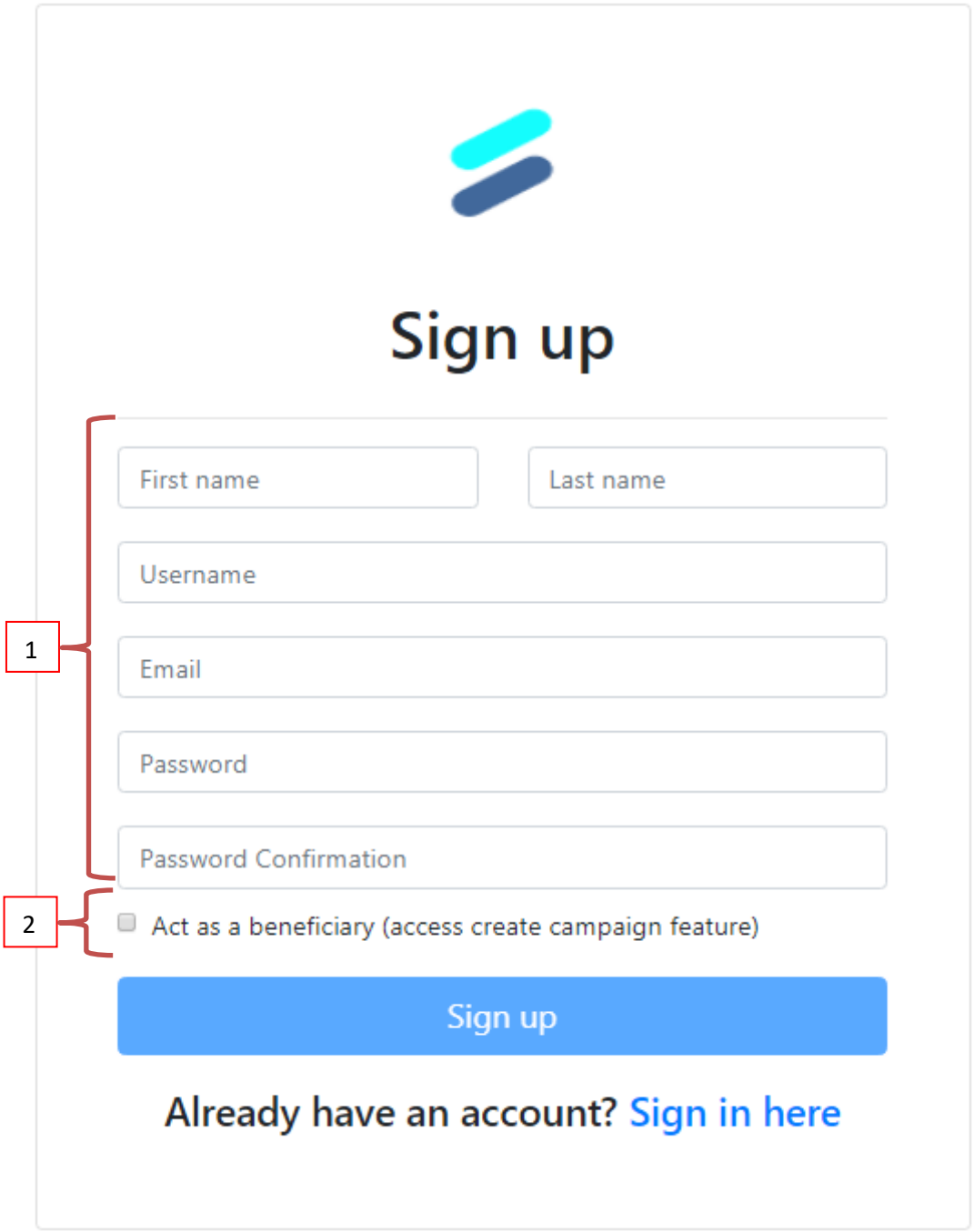


Figure 4.14: Sign up form

Figure 4.14 is a signup form.

1. Normal signup – All of these fields must be specified in order to create an account with our system. Each user account that signs up with our system, it will be sent to the Stellar network as well. From the fact that we also create an

account and link it to the Stellar network, these 2 accounts are linked by public key that is stored in our database and Stellar network.

2. If user want to access the create campaign feature, they can tick the box that state “Act as a beneficiary”

Figure 4.15: Sign up as a beneficiary form

Figure 4.15 is a sign up as a beneficiary form that if registered will give user access to create campaign and manage campaign features. All of these fields will show if the user tick the box that states “Act as a beneficiary”. Users must provide their information in given fields for our admin to verify. Specifically, they should provide an image or scan of their ID card or passport, and a scan of their signature. After the admin verifies the user then the user can access features for beneficiaries 2

4.1.8 Edit profile section

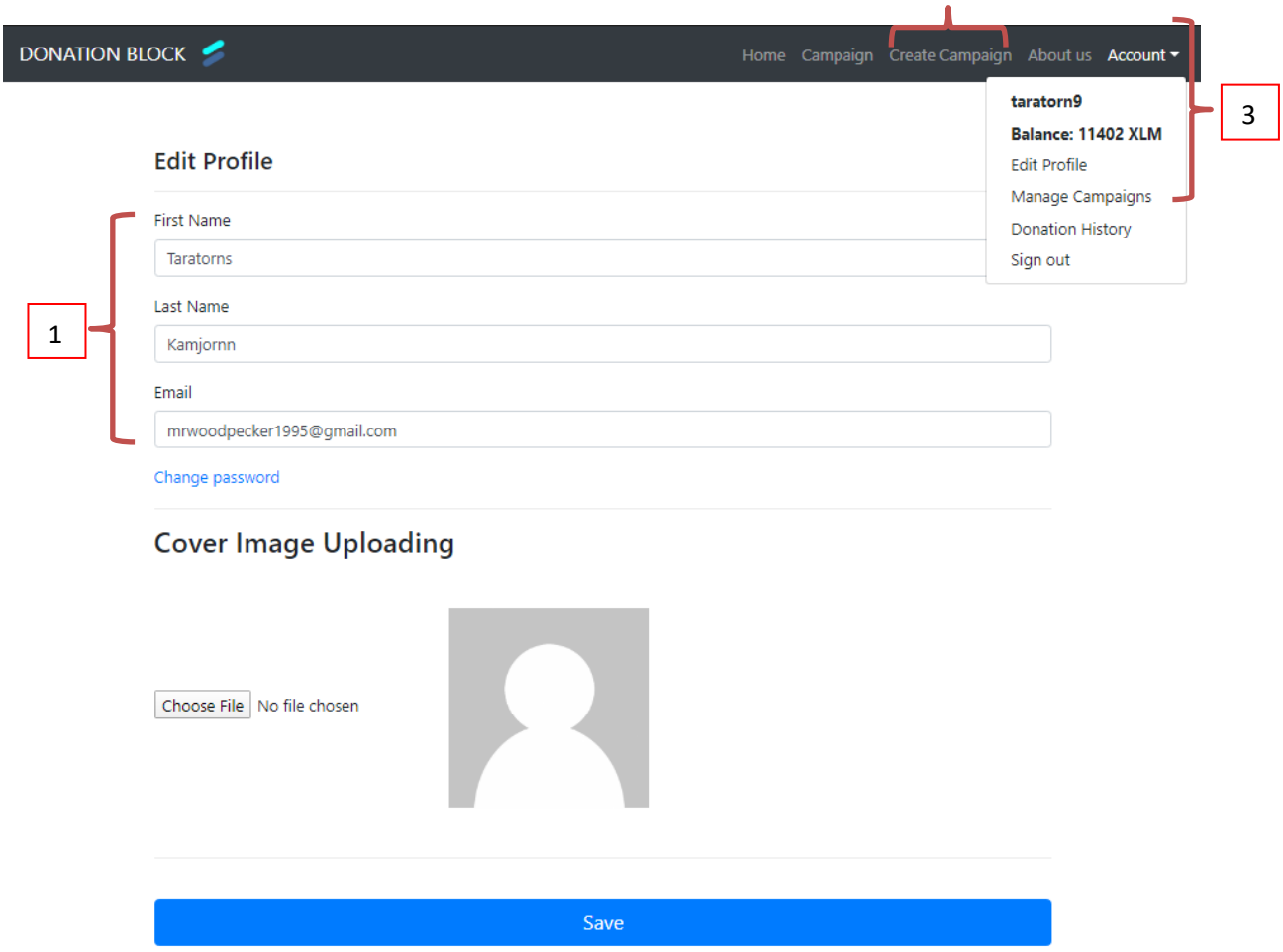


Figure 4.16: Edit user profile page

Figure 4.16 will display filled user information fields when the user selects “Edit Profile” in the “Account” dropdown menu on the top right corner.

1. Edit Profile - This section included first name, last name email and their profile picture. They can edit their information and upload their profile

picture. They also can change their password by click on “Change password”

2. Create Campaign Navigation - This navigation button will only be shown to users who have been verified as beneficiary.
3. User Dropdown Button - This dropdown will only be shown to users registered with our website. From top to bottom, it includes
    a. Username: username of current user
    b. User’s balance in Lumen: When user first registered with our website, they will get 10000 Lumen automatically
    c. Edit profile feature: For editing user’s information
    d. Manage campaign: For managing their created campaign, this will be shown only if they have been verified as beneficiary
    e. Donation history: For user to check their donation history
    f. Sign out: Sign out of the system

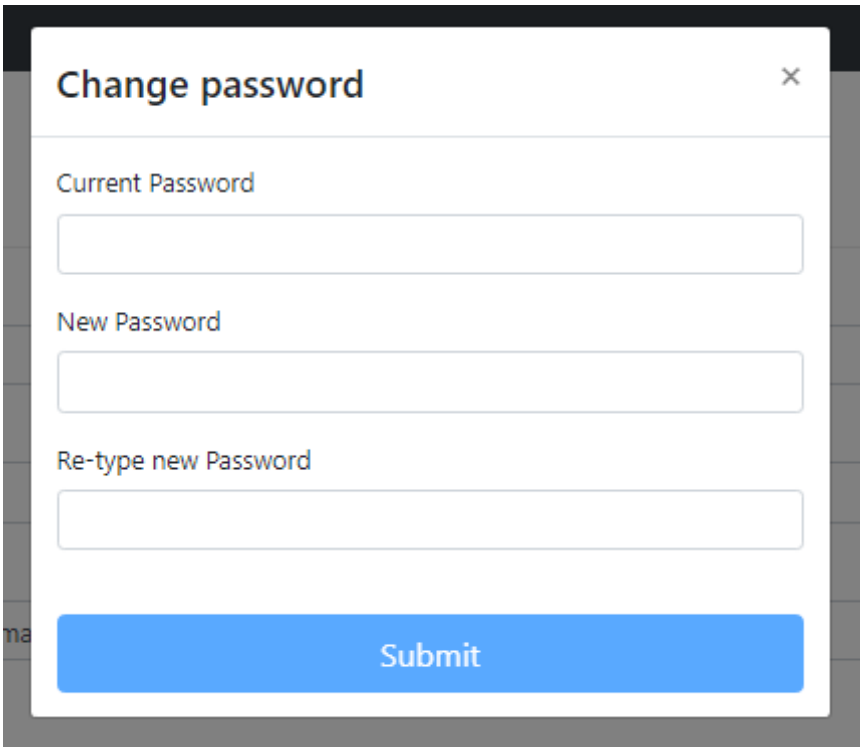


Figure 4.17: Change user’s password popup

Figure 4.17 will display when the user clicks on “Change password”. They need to fill in their current and new password in the given space.

4.1.9 Manage campaign section

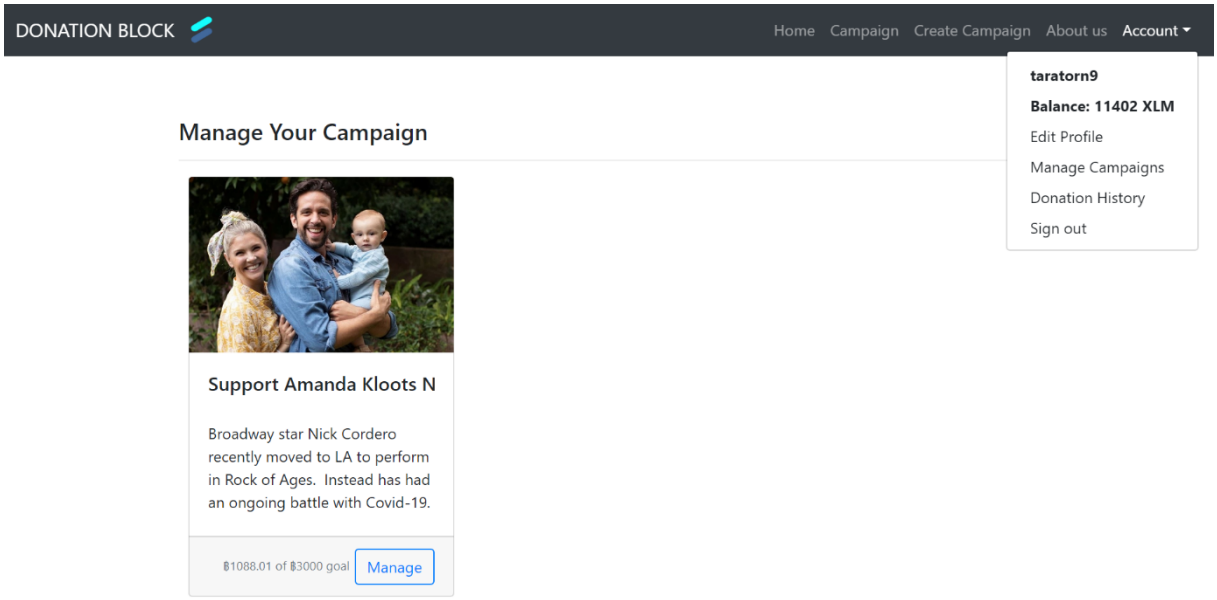


Figure 4.18: Manage Campaign page

Figure 4.18 will display when the user selects “Manage campaign” in the “Account” dropdown menu on the top right corner. It will display all campaigns that are created by the user. User can select a campaign that they want to manage by click on “Manage” button that locate at the foot of the campaign’s card

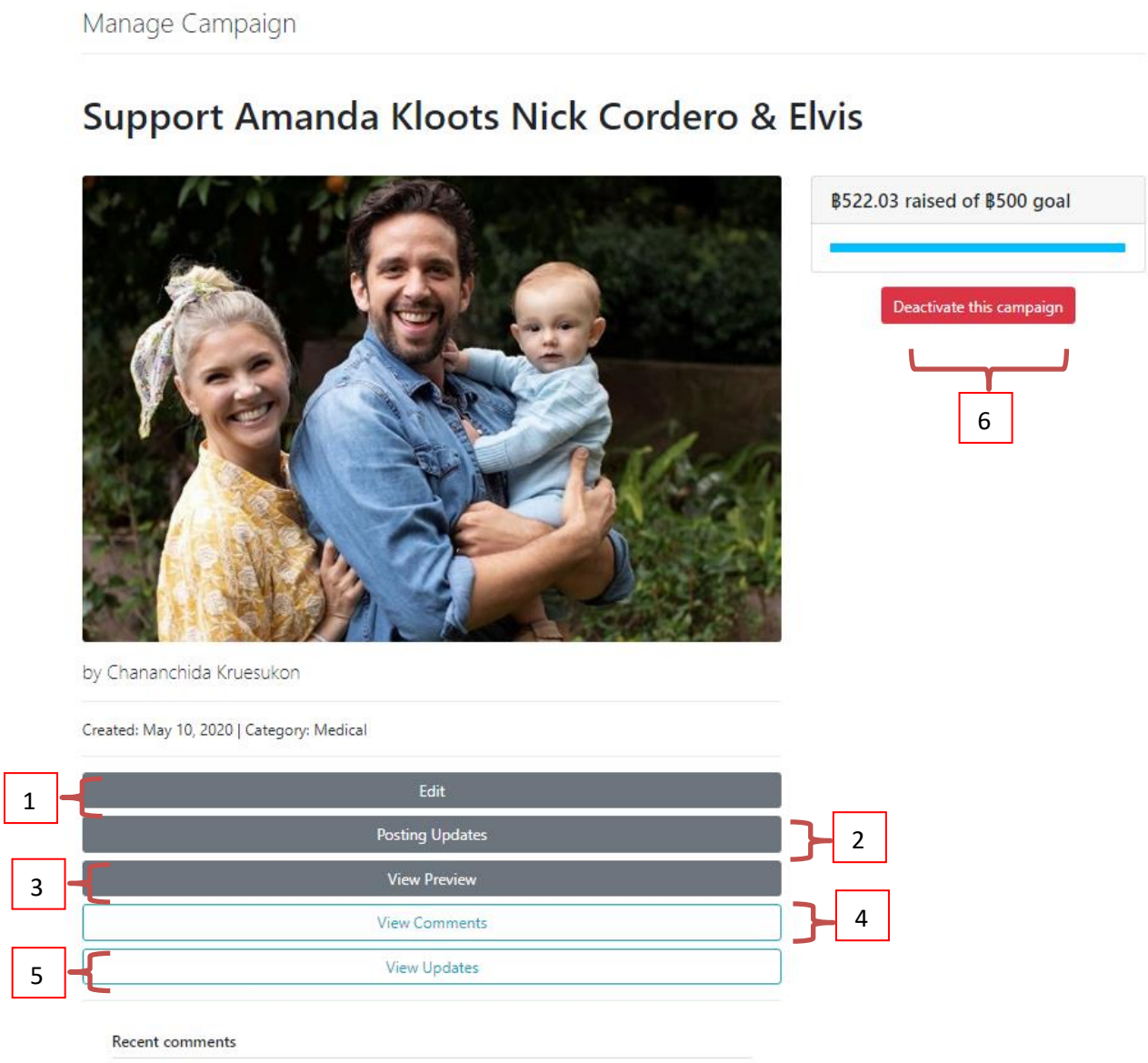


Figure 4.19: Manage Selected Campaign page

Figure 4.19 will display when the user selects a campaign on Manage Campaign page (Figure 4.18). This page provides options for campaign's owner to

1. Edit - Campaign's owner can edit their campaign's information by click on "Edit" button
2. Posting Updates - Campaign's owner can update the state of their campaign by click on "Posting Updates" button
3. View Preview - Campaign's owner can view their campaign in a perspective of other user by click on "View Preview" button

4. View Comments - This is the same as View comment section in Figure 4.7, by click on “View Comments” button
5. View Updates - This is the same as View update section Figure 4.8, by click on “View Updates” button
6. Deactivate this campaign - They can also choose to deactivate the campaign by simply clicking on “Deactivate this campaign” and there will be a confirmation pop up for the owner to make sure they really want to deactivate the campaign. A deactivate campaign works just like an inactive one, no data really gets deleted. All of the data of the deactivate campaign are still in our database but the campaign will no longer display to users.

« Go back to Manage Campaign

Edit Campaign

Goal

฿ 3000

Title

Support Amanda Kloots Nick Cordero & Elvis

Campaign Category

Charity

Fundraising as who

Individual

Cover Image Uploading

Choose File No file chosen

Story

B I U

Broadway star Nick Cordero recently moved to LA to perform in Rock of Ages. Instead has had an ongoing battle with Covid-19. It seems every time he takes a step forward he takes two steps back. He is hooked up to a ventilator, dialysis machine and ecmo while doctors are having trouble getting blood flow to his leg, and even waking him up. If he does wake up, he may never walk again.
But behind him, resiliently cheering him on from a distance - dancing and rallying the world to send him positive energy and love are his wife Amanda Kloots and son Elvis, who is 10 months. Amanda is always one of the first to offer assistance to those in need. She brings kindness to people every day through her fitness training videos and social media following, so we are asking you to contribute to help her now! She needs to pay for the hospital bills which are already starting to come in by donating here.
We also ask you to join us by singing and dancing every day at 3pmPST/ 6pmEST to #wakeupnick
Any support is appreciated. Amanda Nick and Elvis are a ray of light for so many – let's support them in their time of need.

Click for updates from Amanda

You can read more about Nick's story here: (please note that Nick's progress is being updated hourly and these articles may not reflect his current status): asdasd

https://twitter.com/cbsthismorning/status/1255845283973468162?s=21

Submit

Figure 4.20: Edit select campaign page

Figure 4.20 will display filled campaign information fields after the campaign's owner clicks on the "Edit" button on the Manage Selected Campaign page (Figure 4.19). This section included every field when the owner first created the campaign, the owner

can select any field to edit the data then click on the “Submit” button to save the edit data and go back to the Manage Select Campaign page (Figure 4.19). They also can choose to go back to the Manage Select Campaign page (Figure 4.19) manually by clicking on “<< Go back to manage campaign” button at the top.

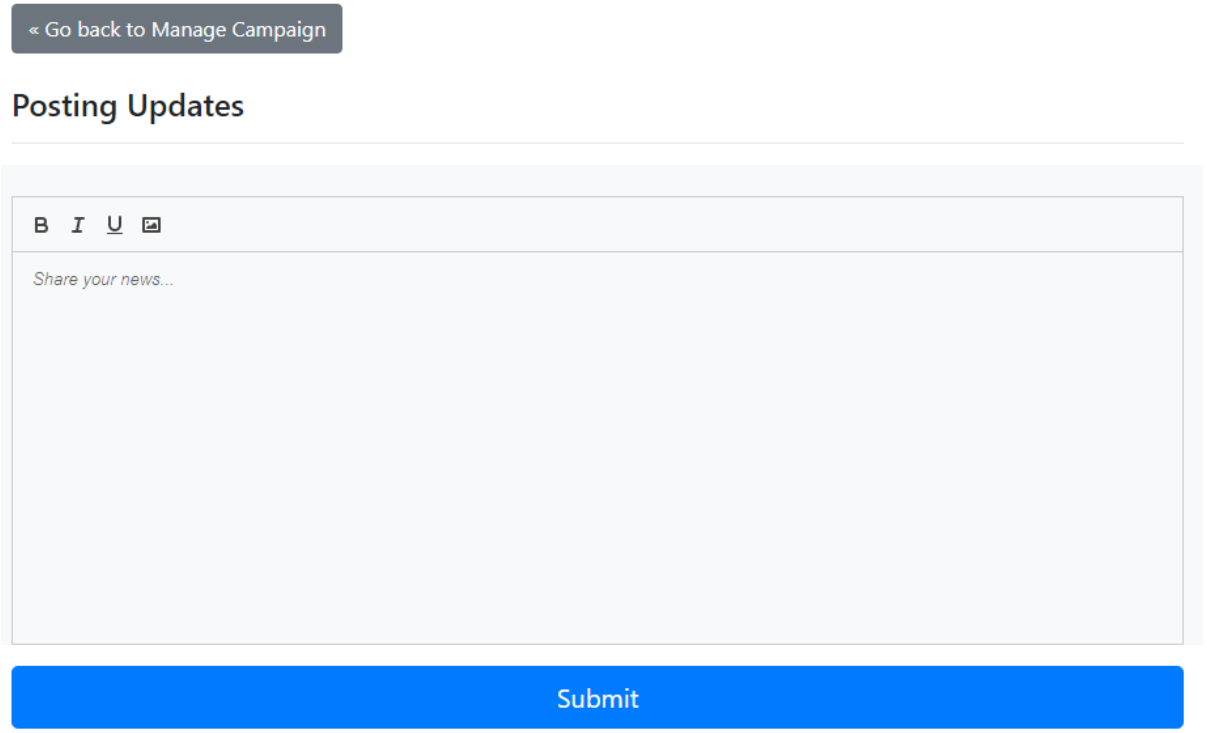


Figure 4.21: Update select campaign page

Figure 4.21 will display a text editor field that the campaign's owner can update the state of the campaign by filling in text or picture then click on “Submit” and go back to the Manage Select Campaign page (Figure 4.19). They also can choose to go back to the Manage Select Campaign page (Figure 4.19) manually by clicking on “<< Go back to manage campaign” button at the top.

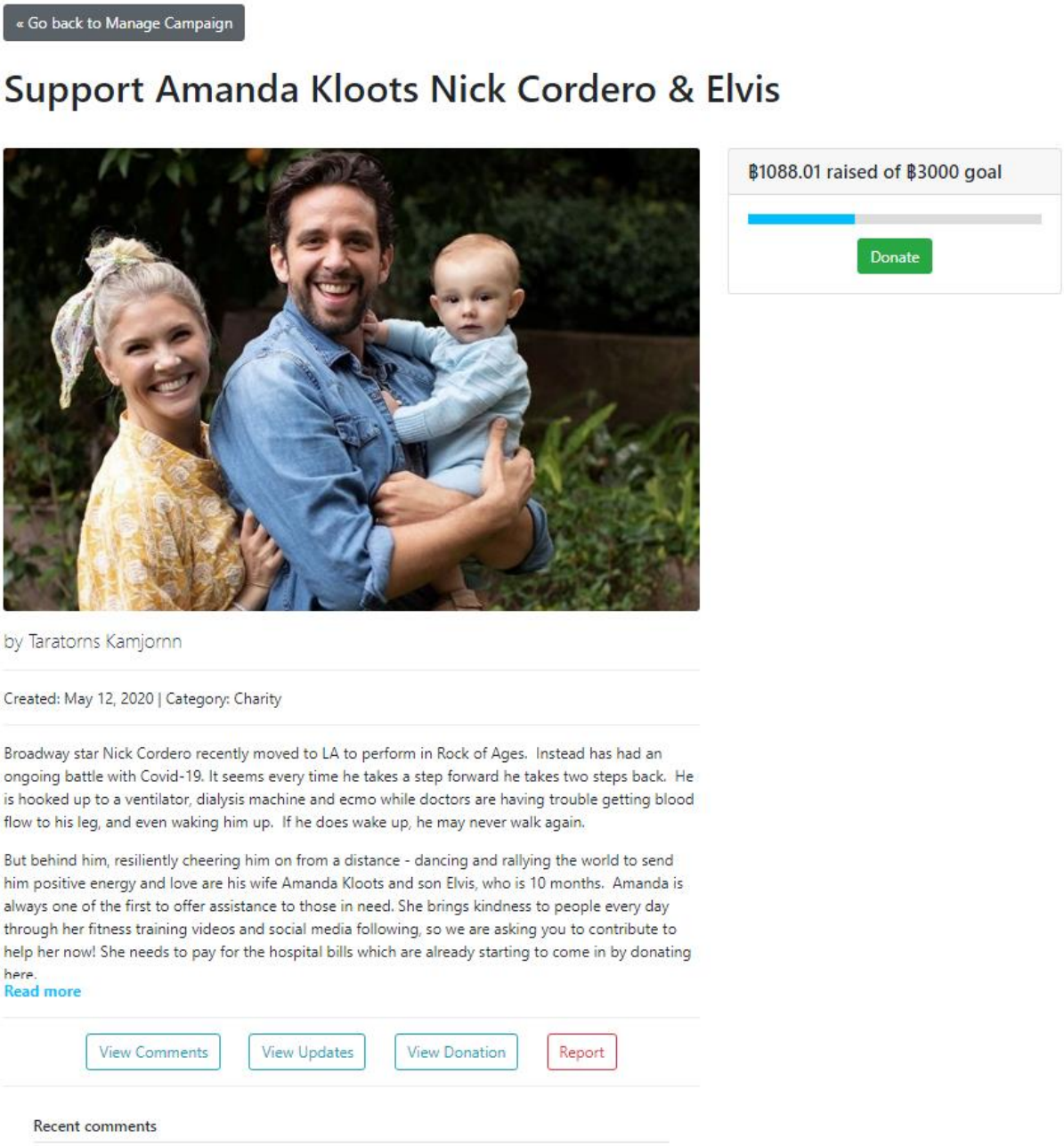


Figure 4.22: Update select campaign page

Figure 4.22 is the same as the Overall campaign page (Figure 4.6) plus the "<< Go back to manage campaign" button at the top for campaign's owner to go back to the Manage Select Campaign page (Figure 4.19) manually.

4.1.10 Donation History page

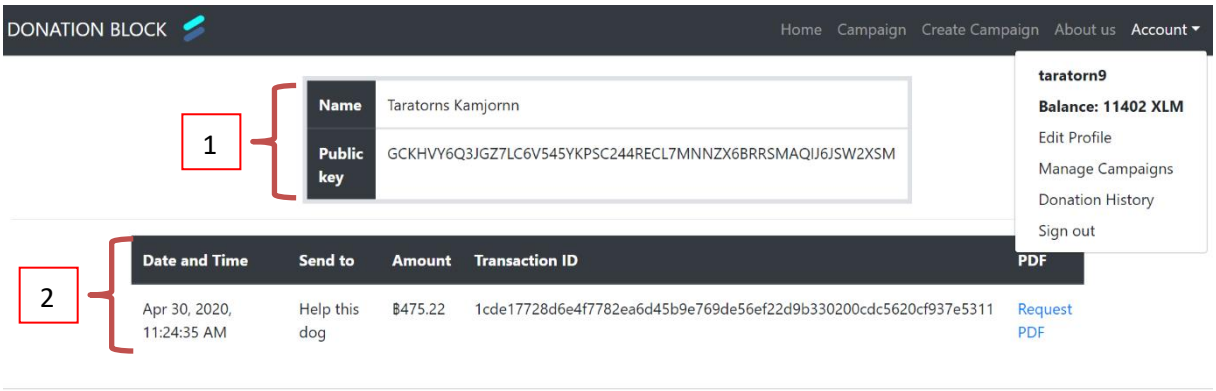


Figure 4.23: Donation History page

Figure 4.23 will display when the user selects “Donation History” in the “Account” dropdown menu on the top right corner. We can separate this page into 2 parts

1. User information - This section will show the user's full name and their public key.
2. Donation history - This section will show history of the user donation when they donate to a campaign. It includes Date and Time when they donated, name of the campaign, amount of the donation in Baht, and TransactionID of the donation. Users can also request the pdf file of the donation for tax reduction by simply clicking on “Request PDF” in the PDF column.

4.1.11 Admin page

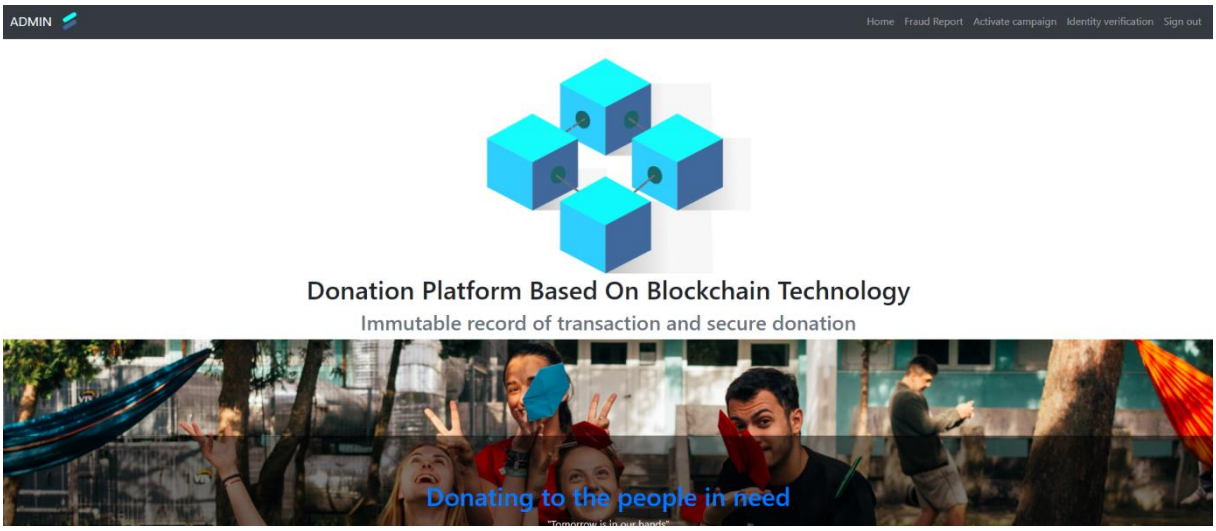


Figure 4.24: Admin Home page

From figure 4.24, it displays the overview of the Admin homepage. In order to access the admin home page user must have an account with admin privilege level.

Figure 4.25: Admin navigation bar

From figure 4.25, it shows all option that admin can operation with the admin account

1. Home - This option is used to navigate users to the Admin home page.
2. Fraud Report - This option will bring the user to the Fraud Report page which is used for displaying all campaigns that have been reported by users. Moreover, this page also provides an ability for admin to suspend a campaign that has a malicious activity.

3. Activate Campaign - This option will bring the user to the Activate Campaign page which is used to activate suspended campaigns.
4. Identity Verification - This option will bring the user to the Identity Verification page which is used to verify the user account.
5. Sign out – this option will sign out and clear current session. After that the system will navigate user to normal home page

4.1.12 Fraud Report page

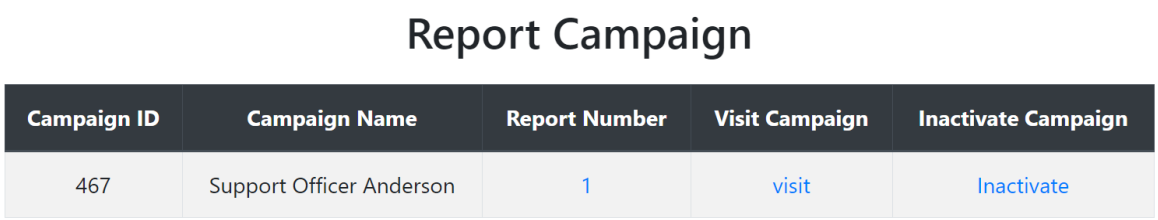


Figure 4.26: Fraud Report Page

From figure 4.26, it illustrates campaigns that have been reported by users. Each campaign that display in this page will has the following information

1. Campaign ID - ID of the campaign.
2. Campaign Name - Name of the campaign.
3. Report Number - The number of reports that have been submitted about the campaign. (For further report detail the user can click on the number that displays in the block). Moreover, this link will bring the user to the Report Detail page.
4. Visit Campaign - This block is used to bring the user to the original campaign for this campaign.
5. Inactivate Campaign - This block is used to inactivate the campaign.

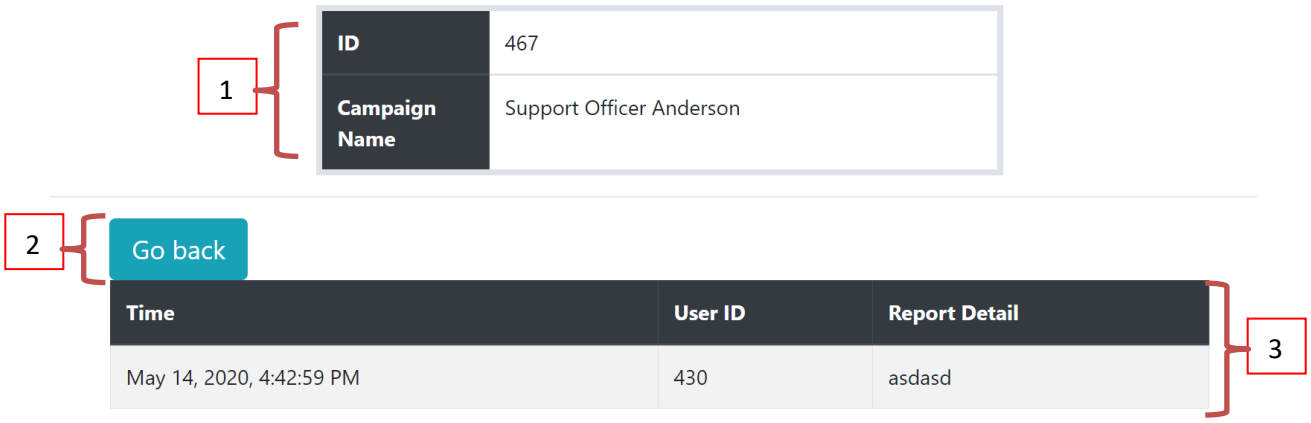


Figure 4.27: Fraud Report Detail Page

From figure 4.27, it displays report detail in each campaign. There are 3 main elements in this page

1. Campaign basic information – This part will display campaign basic information like Campaign ID and Campaign name
2. Go back – This button will navigate user back to display all reports page
3. Information table
    a. Time - Time that user send the report
    b. User ID - User ID of user that report the campaign
    c. Report Detail - Detail of this report

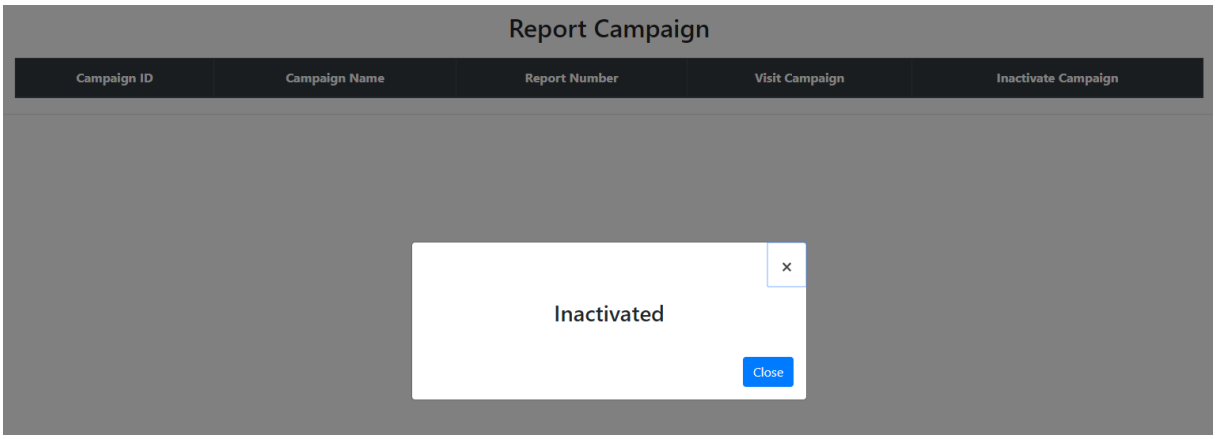


Figure 4.28: Inactivate Campaign

From figure 4.28, it illustrates the pop-up after admin click on “Inactivate” link.

4.1.12 Activate campaign page

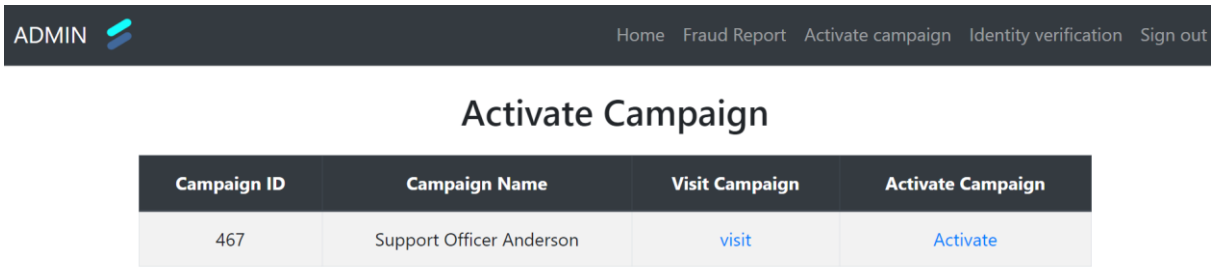


Figure 4.29: Activate campaign table

From figure 4.29, it shows us the Activate Campaign page. This page will be used to activate the suspended campaign. Each row of the table will consist of

1. Campaign ID - ID of the suspended campaign.
2. Campaign Name - Name of the suspended campaign.
3. Visit Campaign - Link to bring the user to the campaign page
4. Activate Campaign - Link to activate the suspended campaign

4.1.13 Identity Verification page

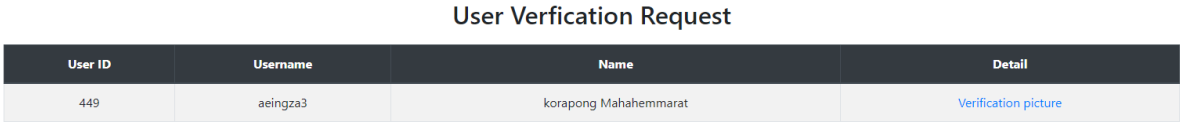


Figure 4.30: User request for identity verification

Figure 4.30 displays a table that contains users who have requested identity verification. Each row in this table will consist of

1. User ID - ID of the user that requested for identity verification
2. Username - Username of the user that requested for identity verification
3. Name - First name and Last name of the user that requested for identity verification
4. Detail - Link for bring the user to User Identity Verification Detail page

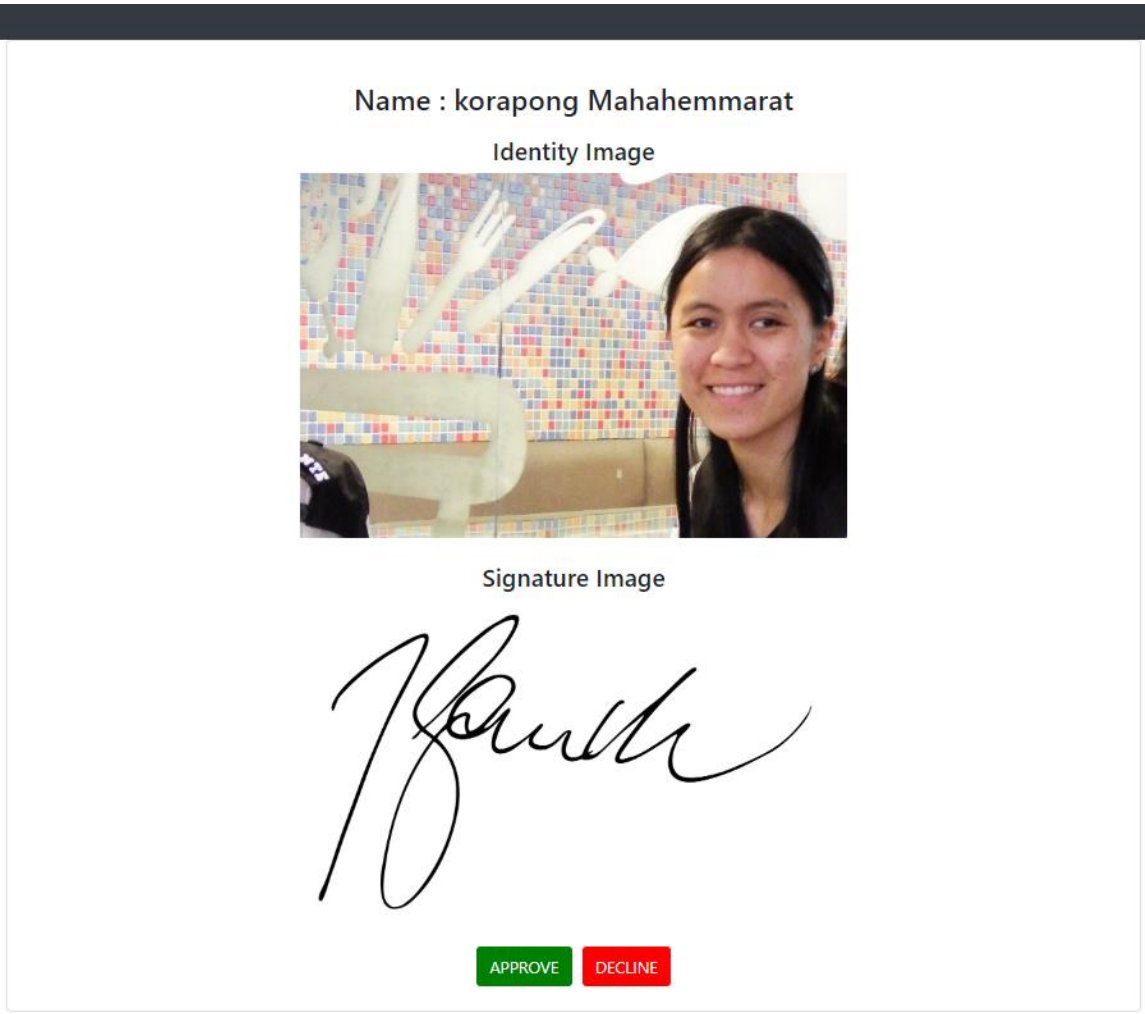


Figure 4.31: User identity in detail

From figure 4.31, it illustrates the requested user in detail. In this page, it will display the identity image and signature image. In this page will consists of two buttons

1. Approve - this button will use when admin want to approve the user request
2. Decline - this will decline the user request

4.1.14 About us

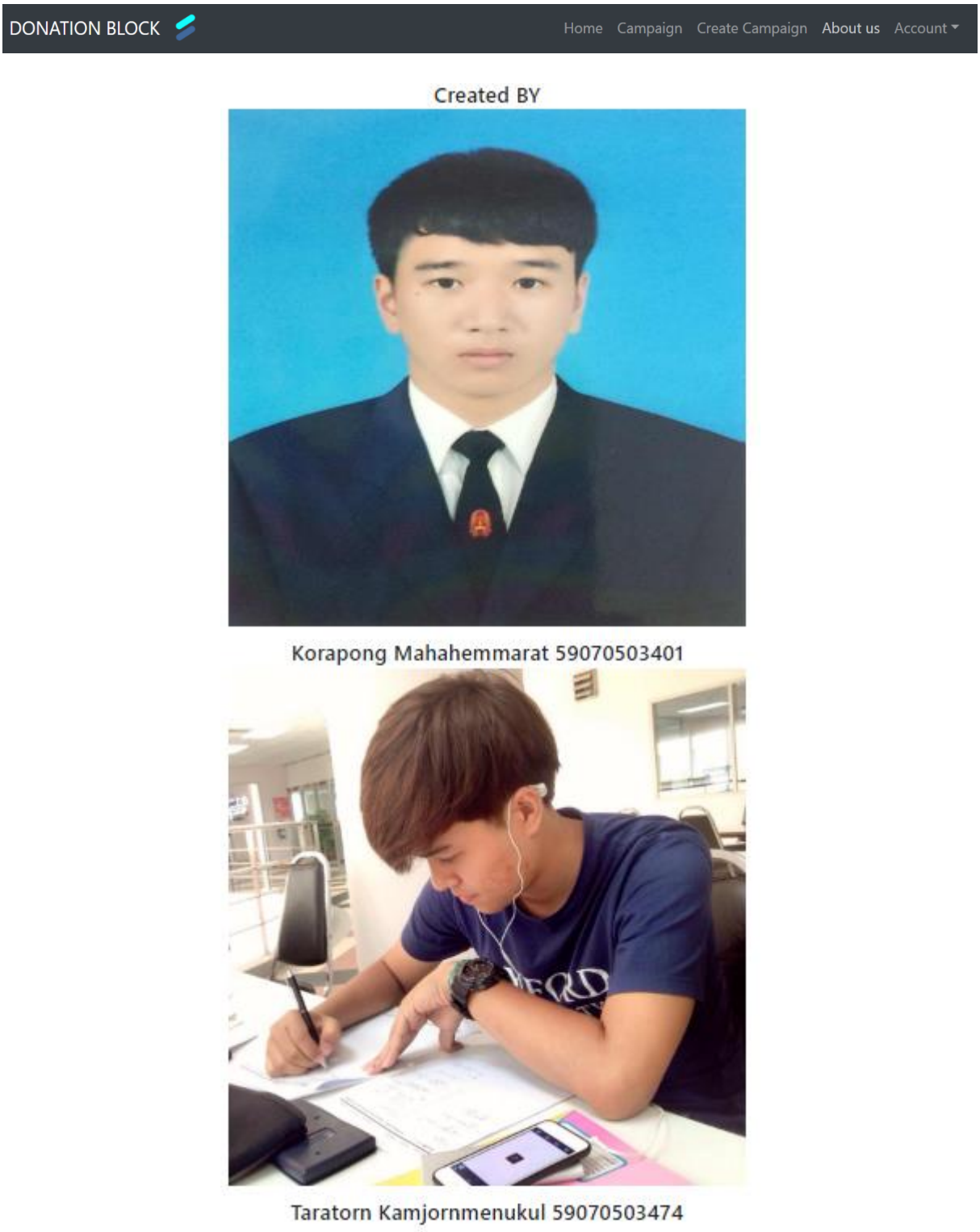


Figure 4.32: About us page

Figure 4.32 will display the website's creators after the user clicks on "About us" in the navigation bar at the top.

4.2 Database

At the current version of our database, our database is able to store users' information whenever they sign up with our system (as described in 4.2.1). Next, our database is able to store the campaign and display it to users correctly. Moreover, we changed the approach on how we should store the campaign details (as described in 4.2.2). Moreover, we removed some of the unnecessary tables such as “Receipt Table”, “Guest Donation” (as described in 4.2.3). However, there is a problem about how the Stellar private key should be stored. First, we planned to store the Stellar private key in the “User” table but we thought that would be a problem about our system security (as described in 4.2.4).

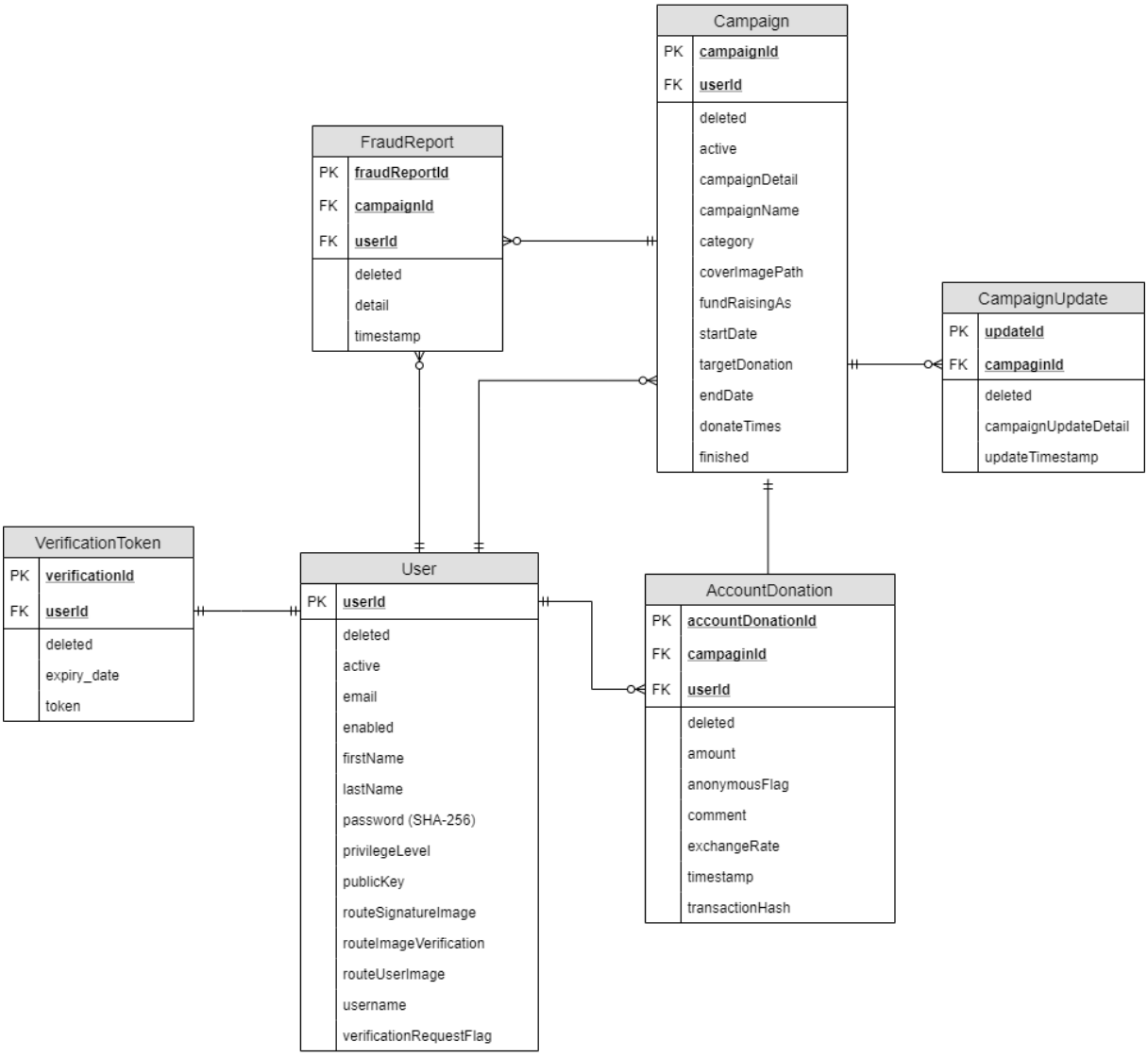


Figure 4.33: Updated database schema

Figure 4.33 is the schema of the system database. There are several changes compared to the old schema.

4.2.1 Verification Table

We add a new table called VerificationToken. This table will operate with the email system. This table will be used during operation that dealing with email system like “Reset user password” and “Email Verification”

Table 4.1 Verification Token

Entity Name	Attribute	Data Type	Constrain	Definition
Verification Token	VerificationId	BigInt(20)	NOT NULL	Identification number for each verification token
	UserId	BigInt(20)	NOT NULL	Identification number for each user
	deleted	Bit(1)	NOT NULL	A bit that use to tell this verification token is delete or not
	expiry_date	Date	NOT NULL	The expire date of this verification token
	token	Varchar(255)	NOT NULL	A string that included in the back verification link

4.2.2 Campaign detail table

We removed the “Campaign detail” table from our database and stored the campaign detail as HTML markup inside the campaignDetail column instead. Since all campaign pages are dynamic, we could handle it in two ways: generate the layout from components when the page is requested for display by using dynamic form to generate components, or generate the HTML and other page components immediately when the user creates or modifies the page content. The initial schema assumed the first approach, but we’ve changed to the second approach. We decided to use WYSIWYG quill editor (what you see is what you get). It uses HTML markup to create content of campaign

detail when the campaign is created or edited. Then we will store the markup in the database. When requested we will display said content as rendered HTML.

4.2.3 Removed Table

From the old database design, some of the tables are not actually used in our system.

1. Receipt Table - Right now we use a package in Java called “ITextpdf” for creating a pdf file when a user requests a receipt.
2. GuestDonation Table - For the reason that we remove this table is because the transaction that occurred outside the system is really hard to track

4.2.4 Private key problem

We are facing a problem about private keys due to the fact that private keys are very valuable. Private is a key that is used to create a signature for each transaction which means if the private got hacked by a hacker, the hacker will be able to do a transaction that is linked to that private key. First, we planned to store a private key in the database but it turns out that we need more security in order to protect a customer private key and we don't want to risk it. Therefore, here are some possible solutions for storing the private key: 1) let users keep it locally; 2) encrypt the private key with some symmetric cryptography.

For the solution, we let user keep the private by their own. When users sign up with our system, our system will send a verification email to each user.

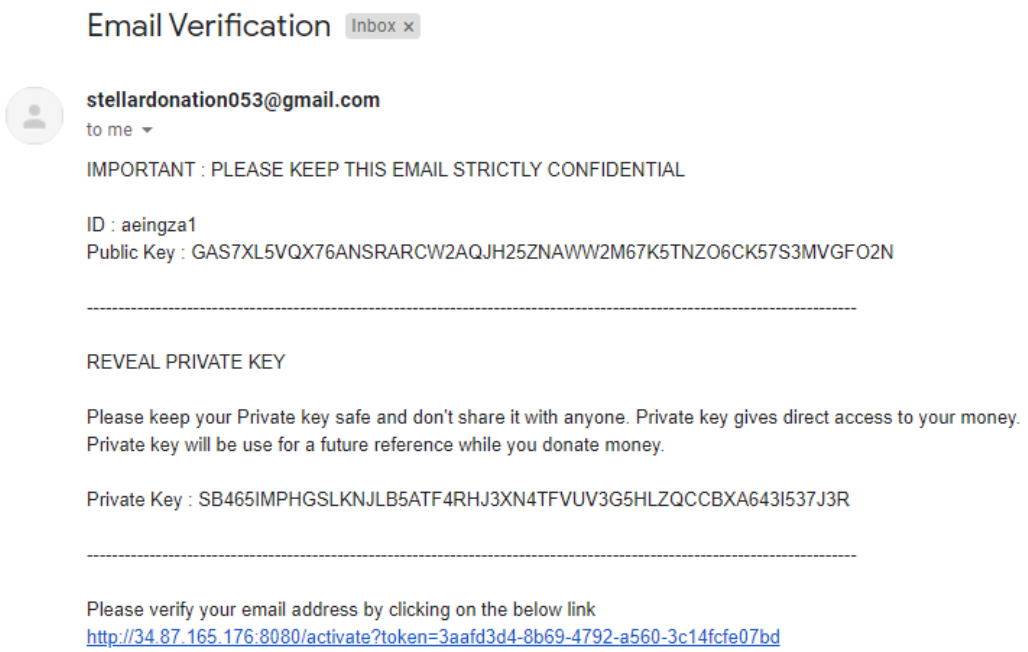


Figure 4.34: Verification email

From figure 4.34, it displays the information of verification email that has been sent to each user that signed up with our system. Inside the email will contain a message that the user should keep the email confidentially and should not expose this email with anyone in order for them to protect their private key.

4.3 Transactions with the blockchain

There are several operations that deal with Stellar Horizon which are Donate, Register New User, Review Own Donation history, and Review Campaign's History of receiving donations.

4.3.1 Stellar core and Horizon implementation

Stellar core and Horizon are the core of our project that use to communicate with the Stellar network. Both of them are in a Docker container that we put in a Google Cloud VM instance. Inside the container, there are two modes for user to choose which are Ephemeral mode and Persistent mode. The Ephemeral mode is a mode for that use

testing environments. Each time user starts a container, the database will start empty and the configuration will be set to default. Next, Persistent mode is much more powerful but it more complicated to operate than the Ephemeral mode. Moreover, the Persistent mode will use user directory from host machine to store all database data and the configuration files that used for running services. From the fact the Persistent mode use a host directory to store everything, this allow host to config or modify the container configuration file. In addition, both of these two modes allow users to choose whether they wanted to deploy on a Test network or a Public network. In this case we choose the Persistent mode and deploy it on the Test network. When we run a Docker container, after that the elements inside the Docker will be spawned and display a message log in the terminal.

Figure 4.35: VM instance that run Stellar core and Horizon

In figure 4.35, it displays the VM instance in Google Cloud that is used for running Stellar core and Horizon. Horizon can be accessed through the public IP address "34.84.1.213" (Allow only HTTP).

```
 stellar/quickstart --testnet
running `/start --testnet'
pids are [6]

Starting Stellar Quickstart

mode: persistent
network: testnet (Test SDF Network ; September 2015)
postgres: config directory exists, skipping copy
supervisor: config directory exists, skipping copy
stellar-core: config directory exists, skipping copy
horizon: config directory exists, skipping copy
postgres: already initialized
chown-core: ok
core: already initialized
horizon: already initialized
starting supervisor
2020-03-04 05:05:50,913 CRIT Supervisor running as root (no user in config file)
2020-03-04 05:05:50,963 INFO RPC interface 'supervisor' initialized
2020-03-04 05:05:50,964 CRIT Server 'unix_http_server' running without any HTTP authentication checking
2020-03-04 05:05:50,964 INFO supervisord started with pid 6
2020-03-04 05:05:51,969 INFO spawned: 'postgresql' with pid 20
2020-03-04 05:05:51,972 INFO spawned: 'stellar-core' with pid 21
2020-03-04 05:05:51,975 INFO spawned: 'horizon' with pid 22
2020-03-04 05:05:51,980 INFO reaped unknown pid 17
2020-03-04 05:05:52,982 INFO success: postgresql entered RUNNING state, process has stayed up for > than 1 seconds
(startsecs)
2020-03-04 05:05:52,987 INFO success: stellar-core entered RUNNING state, process has stayed up for > than 1 second
s (startsecs)
2020-03-04 05:05:52,987 INFO success: horizon entered RUNNING state, process has stayed up for > than 1 seconds (st
artsecs)
```

Figure 4.36: Initial Stellar docker container

Figure 4.36 illustrates the log of Stellar docker initialization. The log displays which component has been initialized. All of these logs happen inside Google Cloud VM instances.

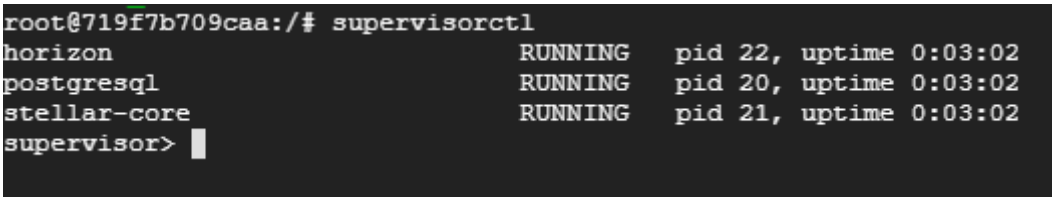


Figure 4.37: Inside Stellar docker container

Figures 4.35, 4.36 and 4.37 show us the component that has been initialized inside the docker. From “supervisorctl” command this allowed us to restart, shut down, or configure something inside the Stellar Docker.

4.3.2 Stellar transaction

In our project there are some modules that deal with the module that we need to create a Transaction to Stellar network in order to store the data in the ledger. Here are the modules that need to create a Stellar transaction 1. Sign up 2. Donate. Both of these two modules are communicating with our Stellar instance inside the Google VM instance

4.3.2.1 Sign up

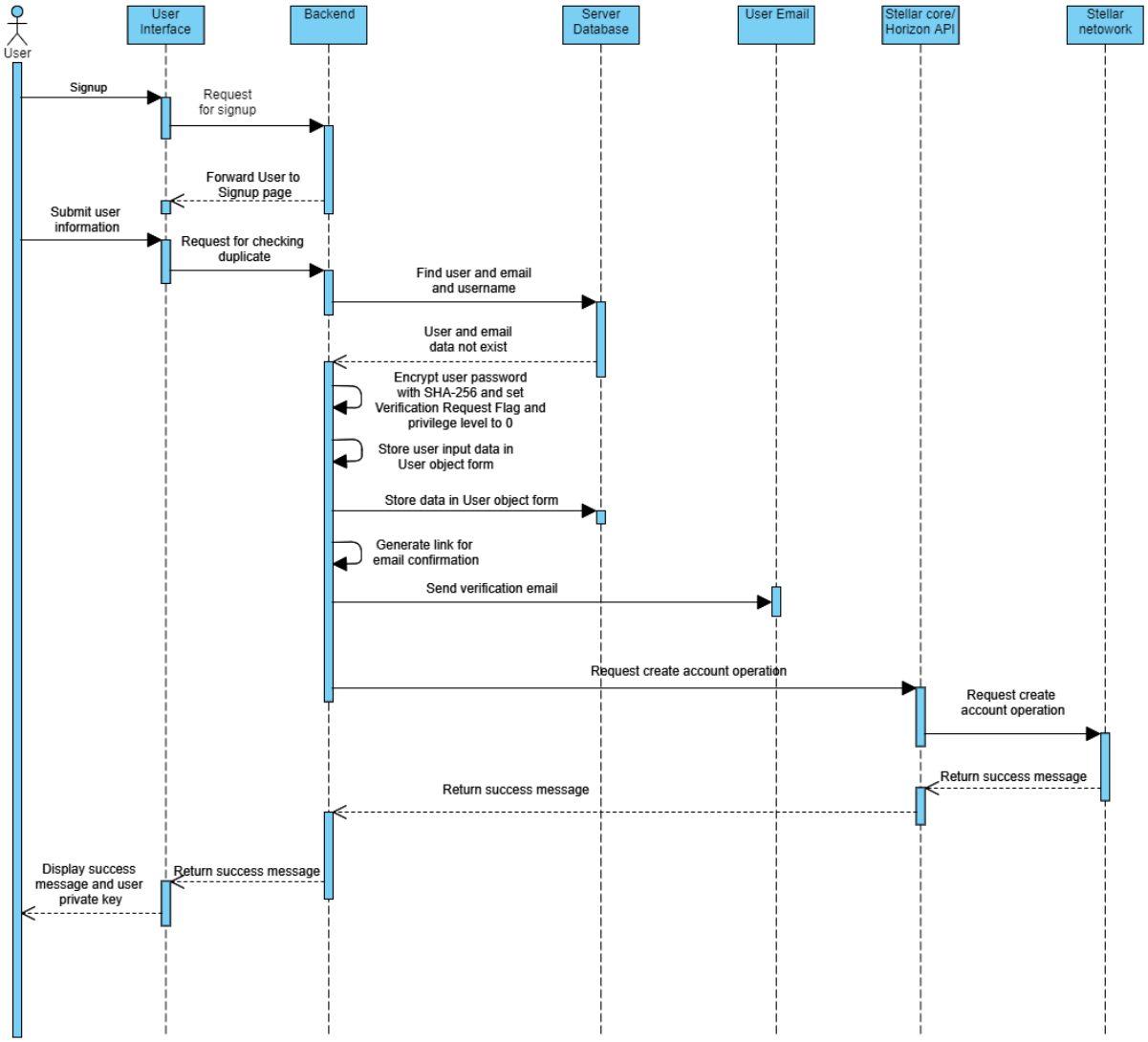


Figure 4.38: Signup Process

Figure 4.38 displays the signup process inside our system. In this process there are 4 components that are involved in this process: 1) System; 2) Stellar Core & Horizon; 3) Stellar network; 4) Database.

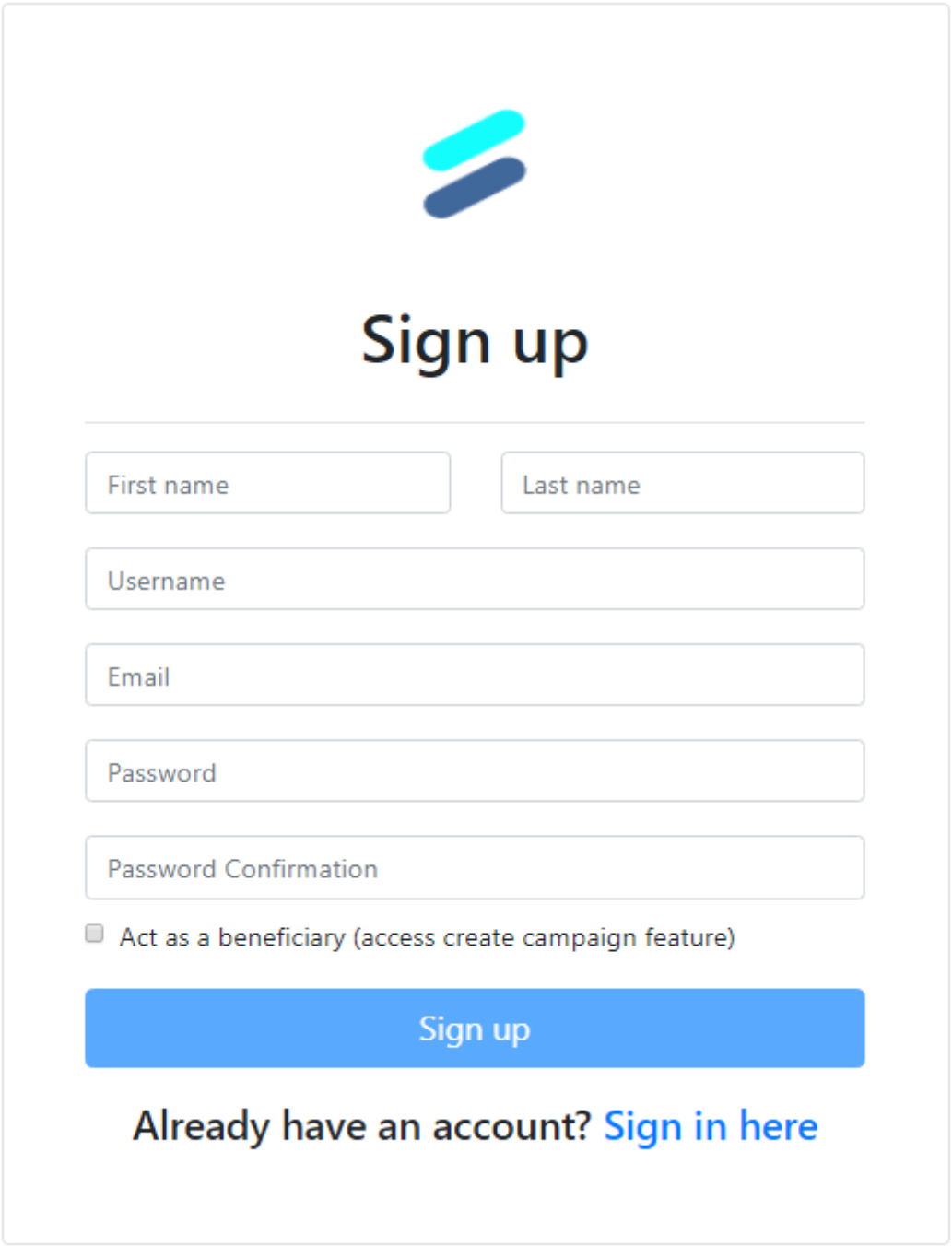


Figure 4.39: Signup page

Figure 4.39 displays a signup form for each user. User must specify all of this information in order to sign up into our system

Figure 4.40: Create account process

Figure 4.40 illustrates the backend (Spring Boot) side log after the user has clicked on the submit button.

1. Our backend will give a result of the transaction result of the account created by Stellar network.
2. Due to the fact that currently our system is working on a Test network so we are able to get a free Stellar asset from the network.
3. Next, we save our user into the database and send a verification email to the user. After the user clicks on the verification link our backend will set the enabled flag to true.

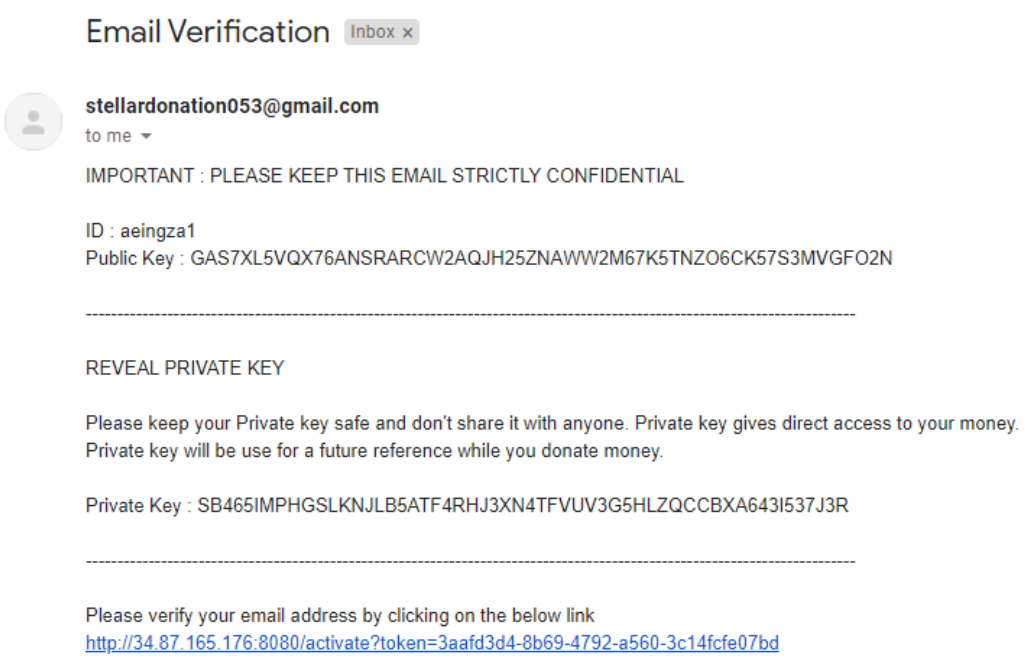


Figure 4.41: Activation link

Figure 4.41 shows the verification link that is sent to the given email. Inside this email contain Username, Public Key, Private Key, link for activate account, and text for user to understand about the important of Private Key

4.3.2.2 Donate

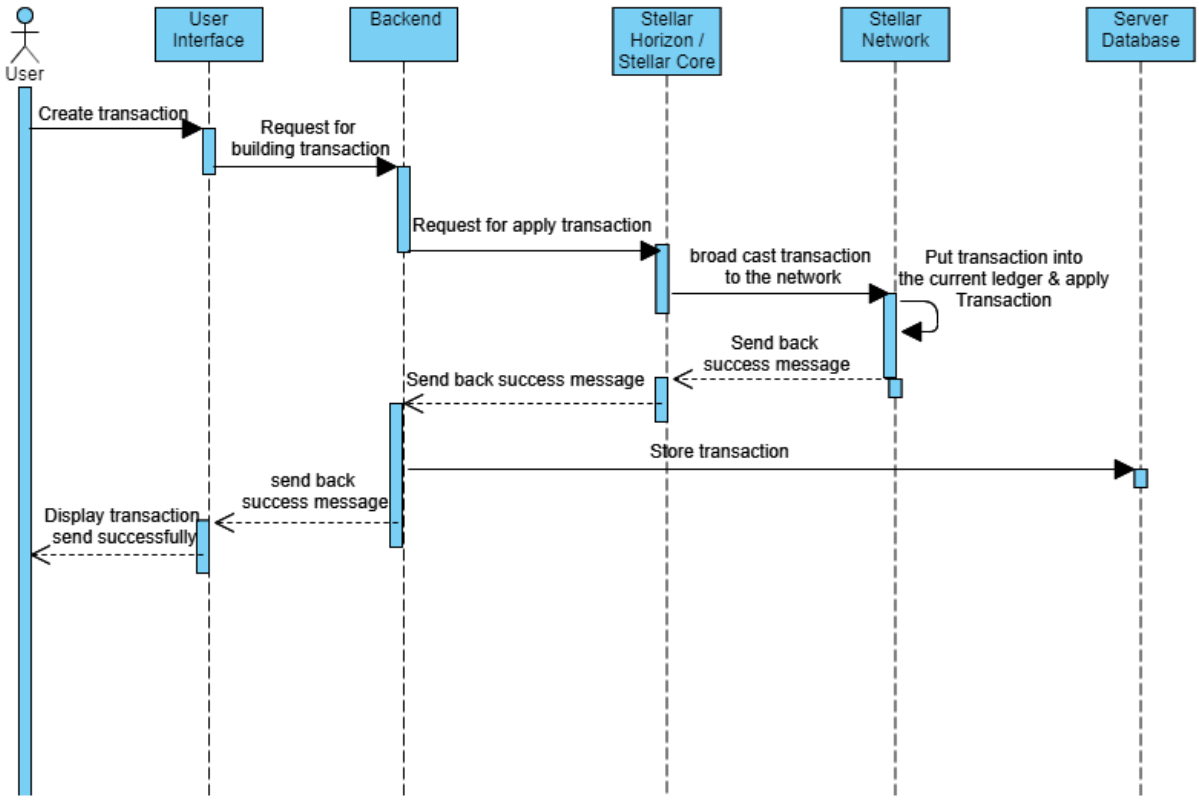


Figure 4.42: Donate process

Figure 4.42 illustrates how each transaction is processed through our system. In this image, there are 4 components that are involved in this process: 1) System; 2) Stellar Core & Horizon; 3) Stellar network; 4) Database.

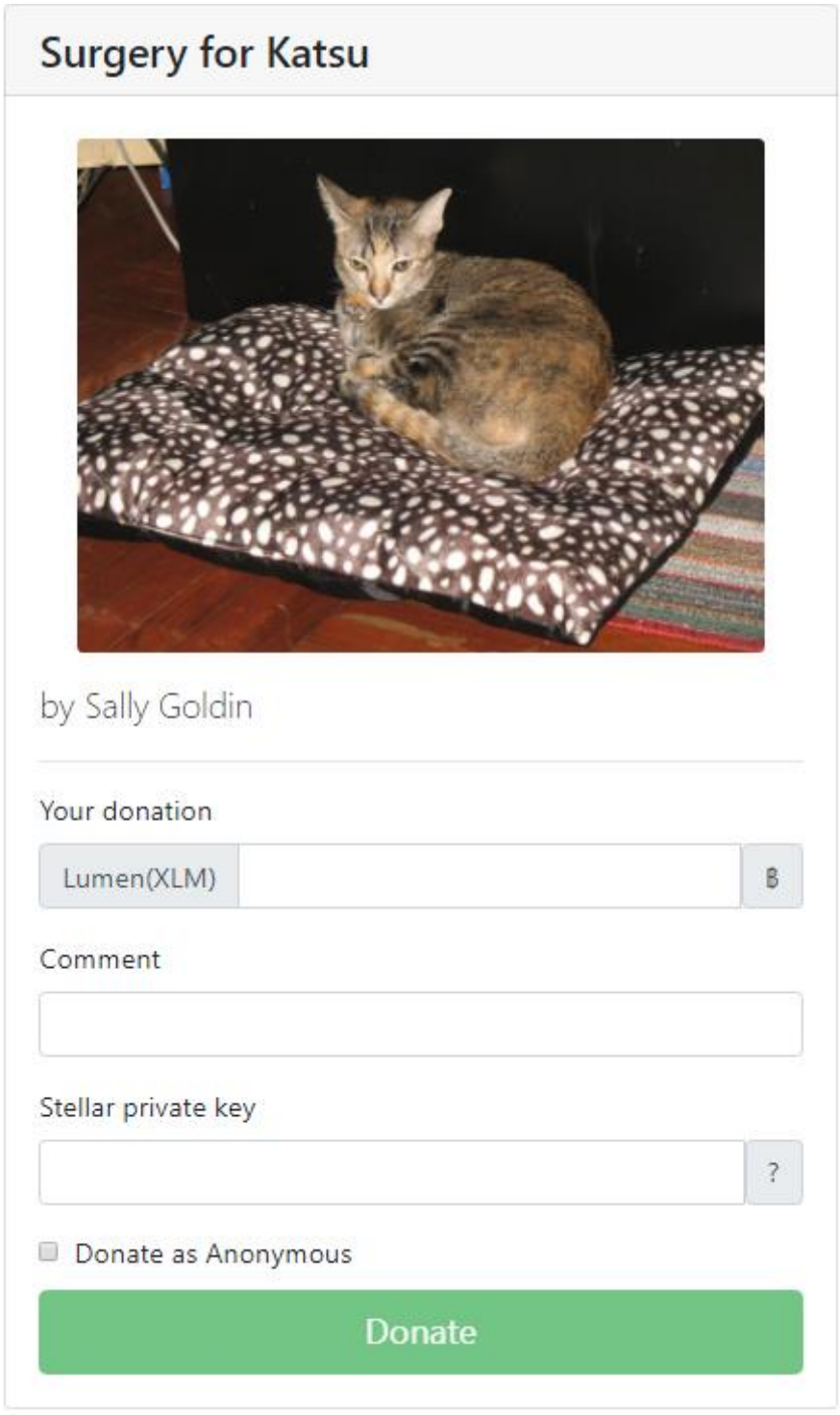


Figure 4.43: Donate page

Figure 4.43 displays the Donate page of our web application. All of these fields except the comment field must be specified in order to do the transaction.

```
Get Private key
Building Transaction
Signing
Get response from serve
Get Hash
Optional.of(org.stellar.sdk.xdr.TransactionResult@f67533c1)
Save database
```

Figure 4.44: Transaction log in Spring Boot

Figure 4.44 illustrates the information log that displays in the backend Spring Boot. As you can see, there are several processes until the transaction will be completed: 1) Building a Stellar transaction object; 2) Signing transaction with user private key; 3) Send transaction object to the Stellar network through our Stellar instance; 4) Get response from Stellar network; 5) Store transaction information in database. From this both of our database and Stellar ledger will store the same information.

id	active	deleted	amount	anonymous_flag	comment	timestamp	transaction_hash	campaign_id	user_id
71	0	0	1000	0	EZ CORONA	2020-03-12 10:30:14.502000	d22ab911d8d444ee61339f20ba1425aa2698c7...	66	69
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.45: Transaction records in the database

Figure 4.45 shows the information about each transaction that is sent by each user.

4.4 Validation plan

This topic will talk about our plan to make a fraud or hacking scenario to demonstrate the utility of the DLT component or Stellar to increase the integrity of the donation system. These are scenarios that we came up with

1. Hacker removes donation records to make it look like the campaign has less money than it does (allow the campaign owner to embezzle money)
2. Hackers change donation amounts (lower) to make it look like the campaign has less money (to allow embezzlement) or MORE money (to encourage donations – e.g. “we’re almost at the limit”.)

First scenario, this problem can be solved by the capability of Stellar Network due to the fact Stellar Network always stores all of transaction information in its ledger. Moreover, Stellar also provides an ability to store a memo(short) message for each transaction so we will set up a memo in order to determine which transaction came from our system. Therefore, even though hackers try to remove donation records we still got Stellar Ledger as a backup database.

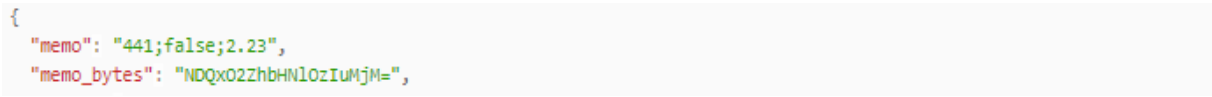


Figure 4.46: Transaction memo

From figure 4.46, it displays the memo that is attached to each transaction that we sent up to Stellar network. The string in memo text can be converted into transaction information. The reason that we attached a memo with each transaction is because each transaction in Stellar did not provide us enough information in order to determine which transaction in our system has been deleted. Therefore, in order to restore transactions, we need some kind of information that is attached with each transaction so that’s why a memo is necessary.

1. First field before semicolon gives us the CampaignID of this transaction.
2. Second field provides us with a user option for donating as anonymous or non-anonymous.

3. Third field provides us the exchange rate from Stellar to Baht at that time.

Therefore, from the memo and the information in each Stellar transaction we can easily construct a new transaction from this information.

Second scenario, the solution of this scenario is to query out the total transaction amount from Stellar network that has been sent to the campaign or public key and compare it with our database.

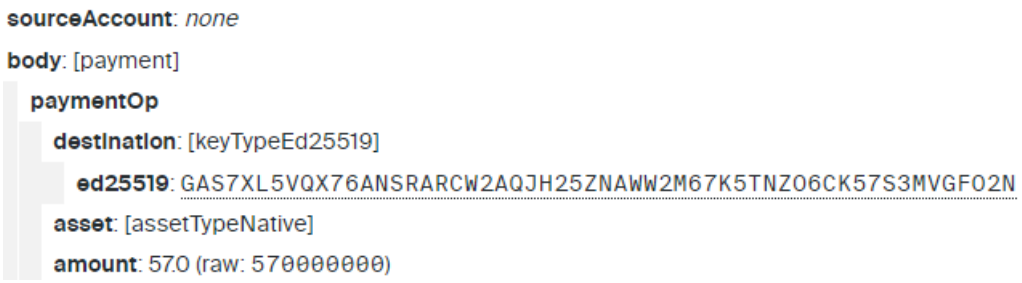


Figure 4.47: Amount of Lumen in the Stellar transaction

From figure 4.47, it illustrates the amount of money (Lumen) in each transaction. Therefore, from this information we can guarantee that if someone changes the amount of money in our system database, we still can convert it back to the original value by querying the Stellar transaction.

The process of comparing between our database and Stellar network can be done by using a sum up method. We will use only data that exists on Stellar Ledger and compare it to our data. If the result between these two are different, we can conclude that there are some changes occurred in our database system. Therefore, this process will happen once a day.

In conclusion, Stellar Ledger acts like a backup database for our system. The Stellar network can be used to query all transaction data that happened in our system. Even though our database got hacked, we still got Stellar Ledger to retrieve the transaction data.

Chapter 5

Conclusions

5.1 Implementation Status

Table 5.1: Task and status

Task	Status	Remaining Work
1.Find Topic		
1.1 List all problem	Complete	-
1.2 List technology	Complete	-
1.3 Select idea topic	Complete	-
2.Research		
2.1 Identify potential DLT Platforms	Complete	-
3.Project Design		
3.1 Use Case	Complete	-
3.2 User interface design	Complete	-
3.3 Architecture design	Complete	-
3.4 Database design	Complete	-
4.Experiment with tools		
4.1 Experiment on Stellar network	Complete	-
4.2 Experiment on Angular framework	Complete	-
4.3 Experiment on Spring Boot	Complete	-
4.4 Build simple prototype	Complete	-
5.Project Development		
5.1 Develop system modules		

5.1.1 Develop core modules		
5.1.1.1 Sign-in	Complete	-
5.1.1.2 Sign-up	Complete	-
5.1.1.3 Create campaign	Complete	-
5.1.1.4 View all campaign	Complete	-
5.1.1.5 Donate using Stellar	Complete	-
5.1.1.6 Track financial activity of beneficiary and individual user	Complete	-
5.1.2 Develop other modules		
5.1.2.1 Search campaign function	Complete	-
5.1.2.2 Admin module	Complete	-
5.1.2.3 Fraud report for campaign	Complete	-
5.1.2.4 Receipt request for campaign	Complete	-
5.1.2.5 Comment on a campaign	Complete	-
5.2 Integrate system modules	Complete	-
5.3 Perform initial testing	Complete	-
6. Testing		
6.1 System testing	Complete	-
6.2 Identify problems	Complete	
6.3 Integrity Testing	Complete	-
6.4 Deployment testing	Complete	-
7. Deployment		

7.1 Deploy web application	Complete	-

5.2 Problems and Solutions

Most of our problems came from financial regulation and fees. At first in our scope, we included features that support both legacy and digital currency which are Thai Baht and Stellar Lumen (XLM) respectively.

For Thai Baht currency there are 2 approaches which are Bank Payment Gateway and Third-Party Payment Gateway. For Bank Payment Gateway like Kasikornbank, problems that we found are 1) A guarantee deposit of 200,000 baht; 2) Transaction fee is 3-5 % of the transaction amount. Because the initial cost of activation and the fee are high, we then decided to drop this feature. For Third Party Payment Gateway for example Omise, problems that we found are 1) Transaction fee is 3.65 % of the transaction amount; 2) VAT 7 % on top of the fee. Because of this we also decided to drop this feature and focus only in Stellar Lumen instead.

As for Stellar currency, we found an approach by using Stellar Wallet like Lobstr, but in order to use the wallet we also need to find an anchor that holds Thai Baht assets and the minimum amount to buy any asset is $30 which is quite a lot. After that, we look into Stellar Test Network which is a private network that is free for developers or anyone to use and experiment with the Stellar system. In the Stellar Test Network, after you register, they will automatically generate you with free 10,000 Lumen that we can test and experiment around when we test our system. After we studied all the options, we came up with the solution that we will reduce our scope and use only the Stellar Test Network.

5.3 What we have learn from this project

We just want to mention that this is the first time for us to have an opportunity to do something this big. We have learned so much from Ajarn Sally on how to manage a project like this.

First, the report, it's true that we all at some point in our University life have learned how to write a report but to write and make readers understand our point of view is quite challenging. By having Ajarn Sally guide us on how to do a proper report, I think we can say that we have progressed quite far from when we started.

Second, technology, blockchain is one of the most talked about tech trends and by having a chance to talk with Dr. Piacentini, a blockchain expert, on this topic and getting the latest technology update from him is quite an opportunity.

Third, tools, we are still new and quite inexperienced in website developing. Each day that we spend time working on the website makes us more familiar with the tools and approach that we should use but there is a lot we have to learn to be a good developer.

Finally, as we've already said that this is our first time with a big project like this. It is really important to understand the scope of what we are trying to do. If you just jump right into a big project and didn't really have a well thought out plan or didn't care for scope or schedule, you will never get it done. That sort of planning when you first started working on a project will help prevent you from getting overwhelmed and giving up. Focus on one problem at a time until you reach your goal.

5.4 Plan for the future

First, if we have a lot more budget and more time, we would like to add features like Bank Payment Gateway or Third-Party Payment Gateway and Stellar Public Network in our website to provide more options for users.

Second, we want to continue to improve our UX/UI to enhance usability, improve user experience and create our own identity to make our website standout more.

Third, we would like to add an option for campaign owners to share their campaign through their social media platforms. This is also one of many factors for a campaign to be successful, by using social media they can attract more donors to their cause.

References

[1] What is charitable? [Online], available: https://www.canada.ca/en/revenue-agency/services/charities-giving/charities/applying-registration/charitable-purposes-activities/what-charitable.html [2019, October 4]

[2] CAF WORLD GIVING INDEX 2017(35) [Online], available: https://www.cafonline.org/docs/default-source/about-us-publications/cafworldgivingindex2017_2167a_web_210917.pdf [2019, October 4]

[3] CAF WORLD GIVING INDEX 2017(33) [Online], available: https://www.cafonline.org/docs/default-source/about-us-publications/caf_wgi2018_report_webnopw_2379a_261018.pdf [2019, October 4]

[4] What is Public Key Cryptography? [Online], available: https://www.twilio.com/blog/what-is-public-key-cryptography [2019, October 4]

[5] asymmetric cryptography (public key cryptography) [Online], available: https://searchsecurity.techtarget.com/definition/asymmetric-cryptography [2019, October 4]

[6] Asymmetric Cryptography In Blockchains [Online], available: https://hackernoon.com/asymmetric-cryptography-in-blockchains-d1a4c1654a71 [2019, October 4]

[7] distributed ledger technology (DLT) [Online], available: https://searchcio.techtarget.com/definition/distributed-ledger [2019, October 4]

[8] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang (2017) An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends (557) , available:

https://www.academia.edu/36208204/An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends [2019, October 3]

[9] BOB WIGLEY NICOLAS CARY Founder Commissioner, Blockchain Commission Co-Founder, Blockchain (23 December 2018) The future is Decentralised Block chains, Distributed Ledgers, & The future of sustainable development (6) available: https://www.undp.org/content/dam/undp/library/innovation/The-Future-is-Decentralised.pdf [2019, October 3]

[10] What Is A Full Node? [Online],available : https://bitcoin.org/en/full-node#what-is-a-full-node [2019, October 3]

[11] Bisade Asolo (November 1, 2018) Full Node and Lightweight Node, available : https://www.mycryptopedia.com/full-node-lightweight-node/ [2019, October 3]

[12] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang (2017) An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends (558), available : https://www.academia.edu/36208204/An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends [2019, October 3]

[13] Du Mingxiao, Ma Xiaofeng, Zhang Zhe, Wang Xiangwei, Chen Qijun [Online] A Review on Consensus Algorithm of Blockchain, available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8123011&tag=1 [2019, October 4]

[14] Blockgenic (November 22, 2018) Asymmetric Cryptography In Blockchains, available : https://hackernoon.com/asymmetric-cryptography-in-blockchains-d1a4c1654a71 [2019, October 4]

[15] PHILIP HAYES & GLORIA ZHAO [Online] Blockchain Fundamentals Lecture 3(43),available : https://blockchain.berkeley.edu/courses/spring-2018-fundamentals-decal/ [2019, October 4]

[16] BLOCKCHAIN ≠ BITCOIN [Online], available: https://kaihan.net/blockchain-does-not-equal-bitcoin/ [2019, October 4]

[17] Everything You Need to Know About Public, Private, and Consortium Blockchain [Online], available: https://medium.com/swlh/everything-you-need-to-know-about-public-private-and-consortium-blockchain-54821c159c7a [2019, October 4]

[18] Public, Private and Consortium blockchains: What's the best flavour? [Online], available:https://medium.com/blockchain-strategy-and-use-cases/public-private-and-consortium-blockchains-whats-the-best-flavour-7728834a4b1c [2019, October 4]

[19] Introduction to Consensus [Online], available: https://xrpl.org/intro-to-consensus.html [2019, October 4]

[20] Antonio Madeira [Online] What is a Block Header in Bitcoin?, available : https://www.cryptocompare.com/coins/guides/what-is-a-block-header-in-bitcoin/ [2019, October 4]

[21] Administration Stellar Core [Online], available: https://www.Stellar.org/developers/Stellar-core/software/admin.html [2019, October 4]

[22] Why run a node? [Online], available: https://www.Stellar.org/developers/Stellar-core/software/admin.html#why-run-a-node [2019, October 4]

[23] Architecture anchor [Online], available: https://www.Stellar.org/developers/guides/anchor/ [2019, October 4]

[24] A Simple Guide to Understanding the Stellar Blockchain Network[Online], available: https://hackernoon.com/a-simple-guide-to-understanding-the-Stellar-blockchain-network-3609d728e9a7 [2019, October 4]

[25] Simply Explained: Stellar Consensus Protocol[Online], available: https://medium.com/london-blockchain-labs/blockchain-definition-of-the-week-Stellar-consensus-protocol-5006f42203b0 [2019, October 4]

[26] What is HTTP?[Online], available: https://www.w3schools.com/tags/ref_httpmethods.asp [2019, October 3]

[27] HTTP - HyperText Transfer Protocol[Online], available: https://www.webopedia.com/TERM/H/HTTP.html [2019, October 3]

[28] Difference between Stateless and Stateful Protocol[Online], available: https://www.geeksforgeeks.org/difference-between-stateless-and-stateful-protocol/ [2019, October 3]

[29] What are HTTP and HTML?[Online], available: https://whatismyipaddress.com/http-html [2019, October 3]

[30] REST Architectural Constraints[Online], available: https://restfulapi.net/rest-architectural-constraints/ [2019, October 3]

[31] Web applications (Web app)[Online], available: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app [2019, October 4]

[32] Software Architecture: One-Tier, Two-Tier, Three Tier, N Tier [Online], available: https://www.softwaretestingmaterial.com/software-architecture/ [2019, October 4]

[33] What is JavaScript? [Online], available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript [2019, October 4]

[34] What is Python? [Online], available: https://www.pythonforbeginners.com/learn-python/what-is-python/ [2019, October 4]

[35] Angular 8 Introduction[Online], available: https://www.javatpoint.com/angular-8-introduction [2019, October 4]

[36] What is AngularJS Good For? [Online], available: https://dev.to/teclogiq/what-is-angularjs-good-for-52im [2019, November 21]

[37] The Good and the Bad of Angular Development[Online], available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/ [2019, October 4]

[38] Explain what Flask is and its benefits? [Online], available: https://www.i2tutorials.com/technology/explain-what-flask-is-and-its-benefits/ [2019, October 4]

[39] The Major Benefits of Using Typescript [Online], available: https://medium.com/swlh/the-major-benefits-of-using-typescript-aa8553f5e2ed [2019, November 21]

[40] Inflation [Online], available: https://www.Stellar.org/developers/guides/concepts/inflation.html [2019, December 4]

[41] Minimum Account Balance [Online], available: https://www.Stellar.org/developers/guides/concepts/fees.html#minimum-account-balance [2019, December 4]

[42] Ledger [Online], available: https://www.Stellar.org/developers/guides/concepts/ledger.html [2019, December 4]

[43] Stellar-core [Online], available: https://github.com/Stellar/Stellar-core/tree/master/docs [2019, December 4]

[44] Homepage of Taejai.com [Online], available: https://taejai.com/th/ [2019, December 4]

[45] Taejai donation page from "พี่สุขสันต์ ปันน้องอิ่มสมอง ปี 2" campaign [Online], available: https://taejai.com/th/d/punbook2/#donate [2019, December 4]

[46] Process of how Asymmetric key work [Online], available: https://hackernoon.com/asymmetric-cryptography-in-blockchains-d1a4c1654a71 [2019, December 4]

[47] The difference between Centralized and Distributed ledger [Online], available: https://tradeix.com/distributed-ledger-technology/ [2019, December 4]

[48] Consensus Algorithm comparison [Online], available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8123011&tag=1 [2019, December 4]

[49] Signing and Verification phase [Online], available: https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq [2019, December 4]

[50] Blockchain classified into 2 dimensions type of network [Online], available: https://kaihan.net/blockchain-does-not-equal-bitcoin/ [2019, December 4]

[51] Block structure connected by hash function [Online], available: https://www.academia.edu/36208204/An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends [2019, December 4]

[52] Block header with more detail [Online], available: https://www.academia.edu/36208204/An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends [2019, December 4]

[53] Block creation process [Online], available: https://www.sciencedirect.com/science/article/pii/S240595951930164X [2019, December 4]

[54] Description of block header data [Online], available: https://www.cryptocompare.com/coins/guides/what-is-a-block-header-in-bitcoin/ [2019, December 4]

[55] Overall view of Stellar network [Online], available: https://hackernoon.com/building-financial-systems-with-Stellar-513a6a43dda2 [2019, December 4]

[56] Role of each level of node [Online], available: https://www.Stellar.org/developers/Stellar-core/software/admin.html#why-run-a-node [2019, December 4]

[57] Disjoint Quorums [Online], available: https://medium.com/london-blockchain-labs/blockchain-definition-of-the-week-Stellar-consensus-protocol-5006f42203b0 [2019, December 4]

[58] Intersect Quorum [Online], available: https://medium.com/london-blockchain-labs/blockchain-definition-of-the-week-Stellar-consensus-protocol-5006f42203b0 [2019, December 4]

[59] Node behavior types [Online], available: https://medium.com/london-blockchain-labs/blockchain-definition-of-the-week-Stellar-consensus-protocol-5006f42203b0 [2019, December 4]

[60] The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus [Online], available: https://assets.website-files.com/5deac75ecad2173c2ccccbc7/5df2560fba2fb0526f0ed55f_Stellar-consensus-protocol.pdf [2020, May 17]

[61] Safety, liveness and fault tolerance—the consensus choices [Online], available: https://www.Stellar.org/blog/safety-liveness-and-fault-tolerance-consensus-choice [2020, May 17]

[62] What is Docker and why is it so darn popular? [Online], available: https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/ [2020, May 17]

[63] What is a Container [Online], available: https://www.docker.com/resources/what-container [2020, March 15]

Appendix A

Use Case Narrative Detail

Use case narratives

Use Case: Sign up

Actor: Campaign creator, User

Goal: To apply and become a member of the system

Preconditions: -

Main success scenario:

1. User choose the sign-up operation
2. System forward user to sign up page
3. User then enter username, password, password, first name, last name
4. User click on submit button
5. System check on duplicate email and username
6. System can't detect duplicate email and username
7. System display email confirmation message
8. System encode user password with SHA-256 hash function and store in User object form
9. System store data in User object form into database with privilege level of 0 (Account is not activated) and Verification Request Flag to 0
10. System generates a link for email confirmation
11. System sends a link to user given email
12. System received confirmation from email sent by system

13. System generate public key and private key for user
14. System sends public key and private key to Stellar network in order to create a Stellar Account through “Create Account” operation by Horizon API
15. System received a success message from Stellar network
16. System store public key into database and set user privilege level up to 1 (Account is activated)
17. System displays success and user’s private key to the user in order for the user to keep it locally

Alternative scenario 1 - Stellar provides failure message:

1. User choose the sign-up operation
2. System forward user to sign up page
3. User then enter username, password, password, first name, last name
4. User click on submit button
5. System check on duplicate email and username
6. System can’t detect duplicate email and username
7. System display email confirmation message
8. System encode user password with SHA-256 hash function and store in User object form
9. System store data in User object form into database with privilege level of 0 (Account is not activated) and Verification Request Flag to 0
10. System generates a link for email confirmation
11. System sends a link to user given email

12. System received confirmation from email sent by system
13. System generate public key and private key for user
14. System sends public key and private key to Stellar network in order to create a Stellar Account through “Create Account” operation by Horizon API
15. System received a failure message from Stellar network
16. System read error code from failure message
17. System resend “Create Account” operation by Horizon API
18. If operation failed more than 5 times, System display error message to user

Alternative scenario 2 - Duplicated email or username:

1. User choose the sign-up operation
2. System forward user to sign up page
3. User then enter username, password, password, first name, last name
4. User click on submit button
5. System check on duplicate email and username
6. System detect duplicate email or password
7. System displays an error message to user

Alternative scenario 3 - User choose to sign up with campaign creator privilege level:

1. User choose the sign-up operation
2. System forward user to sign up page
3. User then enter username, password, password, first name, last name

4. User click on “Act as a beneficiary (access create campaign feature)” choice
5. System displays identity verification that user need to specify
6. User provides an identity image and signature
7. User click on submit button
8. System display email confirmation message
9. System encode user password with SHA-256 hash function and store in User object form
10. System store data in User object form into database with privilege level of 0 (Account is not activated) and Verification Request Flag to 1
11. System generates a link for email confirmation
12. System sends a link to user given email
13. System received confirmation from email sent by system
14. System generate public key and private key for user
15. System sends public key and private key to Stellar network in order to create a Stellar Account through “Create Account” operation by Horizon API
16. System received a success message from Stellar network
17. System store public key into database and set user privilege level up to 1 (Account is activated)
18. System displays success and user’s private key to the user in order for the user to keep it locally
19. System displays user identity verification request to admin
20. Admin approved user identity verification

21. System set user privilege level up to 2 and set Verification Request Flag back to 0
22. System sends identity verification success to user's email

Alternative Scenario 4 - Admin deny identity verification request:

1. User choose the sign-up operation
2. System forward user to sign up page
3. User then enter username, password, password, first name, last name
4. User click on submit button
5. System check on duplicate email and username
6. System can't detect duplicate email and username
7. System display whether the person plans to act in the Beneficiary role
8. User click on "Yes" button
9. User provides identity verification information
10. User click on submit button
11. System display email confirmation message
12. System encode user password with SHA-256 hash function and store in User object form
13. System store data in User object form into database with privilege level of 0 (Account is not activated) and Verification Request Flag to 1
14. System generates a link for email confirmation
15. System sends a link to user given email
16. System received confirmation from email sent by system

17. System generate public key and private key for user
18. System sends public key and private key to Stellar network in order to create a Stellar Account through “Create Account” operation by Horizon API
19. System received a success message from Stellar network
20. System store public key into database and set user privilege level up to 1 (Account is activated)
21. System displays success and user’s private key to the user in order for the user to keep it locally
22. System displays user identity verification request to admin
23. Admin denied user identity verification
24. System set Verification Request Flag back to 0
25. System sends identity verification failed to user’s email

Use case: Login

Actor: Campaign creator, User

Goal: To be able to get a user feature of the platform (Depends on user privilege level)

Preconditions: User must already sign up with the system

Main success scenario:

1. Click “Sign in” icon on the top right
2. System forward user to Sign in page
3. User then enter their “Email address” and “Password”
4. Then click “Sign in” button
5. System hash user password with SHA-256
6. System search user from given Email address
7. System found user Email address
8. System compare hashed password and stored password to see if they match
9. System query out user’s information
10. System check any session open on this user account or not
11. System cannot detect any session on user account
12. System open session for user
13. System forwarding user to the home page

Alternative scenario 1 - Incorrect username

1. Click “Sign in” icon on the top right
2. System forward user to Sign in page
3. User then enter their “Email address” and “Password”
4. Then click “Sign in” button

5. System hash user password with SHA-256
6. System search user from given Email address
7. System can’t find user
8. System displays error message

Alternative scenario 2 - Incorrect password

1. Click “Sign in” icon on the top right
2. System forward user to Sign in page
3. User then enter their “Email address” and “Password”
4. Then click “Sign in” button
5. System hash user password with SHA-256
6. System search user from given Email address
7. System found user Email address
8. System compare password and password doesn’t match
9. System displays error message

Alternative scenario 3 - User is already logged in on another device

1. Click “Sign in” icon on the top right
2. System forward user to Sign in page
3. User then enter their “Email address” and “Password”
4. Then click “Sign in” button
5. System hash user password with SHA-256
6. System search user from given Email address
7. System found user Email address
8. System compare password and password matched
9. System query out user’s information
10. System check any session open on this user account or not

11. System detected existing session on user account

12. System displays error message

Use case: Logout

Actor: Campaign creator, User

Goal: Let user sign out of the system successfully

Preconditions: User must already sign in to the system

Main success scenario:

1. Click user icon on the top right
2. Choose “Sign out”
3. System clear current session of user
4. System displays sign out success message to user

Use Case: Create a campaign

Actor: Campaign creator

Goal: Create a new campaign for donation

Precondition: User must already sign in to the system with privilege level more than 1 or identity verified by admin

Main success scenario:

1. Click “Create Campaign” on the top
2. System forward user to Create campaign page
3. Then user must fill in “Enter your goal (Money)”, “Campaign title”, and “Fundraising as whom?”
    a. Fundraising for some organizations
    b. Fundraising for individual

4. User select “Fundraising for individual” and click “Next” button
5. System checks the existence of input information, input information existed
6. System check on user privilege level and current user privilege level is allowed
7. System save information into Campaign object form and forward both Campaign object form and user to the Create campaign add cover page
8. User add a cover photo of their campaign by Choosing “Upload photo” button
9. System display uploaded photo
10. User click on “Next” button
11. System check cover photo existence, cover photo exist
12. System store photo in Campaign object form that sent from previous page and pass Campaign object form to the Create campaign story page
13. System forward user to Create campaign story page
14. User write down a story of campaign by clicking “Text” button if user want to add text into the campaign story or “Upload Image” button if user want to add image into the campaign story
15. System get user option and detail and save into Campaign object form that sent from previous page
16. System display given text or image
17. Repeat step 14-16 until user click on the save button
18. System check existing campaign story, campaign story data exist

19. System store all photo data in directory and save path in Campaign object form
20. System store data from Campaign object form except photo into the database
21. System display success message

Alternative scenario 1 - Create campaign for some organizations

1. Click “Create Campaign” on the top
2. System forward user to Create campaign page
3. Then user must fill in “Enter your goal (Money)”, “Campaign title”, and “Fundraising as whom?”
    a. Fundraising for some organizations
    b. Fundraising for individual
4. System display organizations
5. User select organization from the list and click on “Next” button
6. System checks the existence of input information, input information existed
7. System check on user privilege level and current user privilege level is allowed
8. System save information into Campaign object form and forward both Campaign object form and user to the Create campaign add cover page
9. User add a cover photo of their campaign by Choosing “Upload photo” button
10. System display uploaded photo
11. User click on “Next” button

12. System check existence of cover photo data, cover photo exist
13. System store photo in Campaign object form that sent from previous page and pass Campaign object form to the Create campaign story page
14. System forward user to Create campaign story page
15. User write down a story of campaign by clicking “Text” button if user want to add text into the campaign story or “Upload Image” button if user want to add image into the campaign story
16. System get user option and detail and save into Campaign object form that sent from previous page
17. System display given text or image
18. Repeat step 16-18 until user click on the save button
19. System check existing campaign story, campaign story data exist
20. System store all photo data in directory and save path in Campaign object form
21. System store data from Campaign object form except photo into the database
22. System display success message

Alternative scenario 2 - User privilege is too low

1. Click “Create Campaign” on the top
2. System forward user to Create campaign page
3. Then user must fill in “Enter your goal (Money)”, “Campaign title”, and “Fundraising as who?”
    a. Fundraising for some organizations
    b. Fundraising for individual

4. User select “Fundraising for individual” and click “Next” button
5. System check on user privilege level and current user privilege level is too low
6. System display error message

Alternative scenario 3 - User didn’t specify campaign basic information

1. Click “Create Campaign” on the top
2. System forward user to Create campaign page
3. Then user must fill in “Enter your goal (Money)”, “Campaign title”, and “Fundraising as whom?”
    a. Fundraising for some organizations
    b. Fundraising for individual
4. User select “Fundraising for individual” and click “Next” button
5. System checks the existence of information, information does not exist
6. System display error message

Alternative scenario 4 - User didn’t insert campaign cover photo

1. Click “Create Campaign” on the top
2. System forward user to Create campaign page
3. Then user must fill in “Enter your goal (Money)”, “Campaign title”, and “Fundraising as who?”
    a. Fundraising for some organizations
    b. Fundraising for individual
4. User select “Fundraising for individual” and click “Next” button
5. System checks the existence of information, information existed

6. System check on user privilege level and current user privilege level is allowed
7. System save information into Campaign object form and forward both Campaign object form and user to the Create campaign add cover page
8. User click on “Next” button
9. System check existence of cover photo data, cover photo does not exist
10. System display error message

Alternative scenario 5 - User didn’t insert campaign story

1. Click “Create Campaign” on the top
2. System forward user to Create campaign page
3. Then user must fill in “Enter your goal (Money)”, “Campaign title”, and “Fundraising as whom?”
    a. Fundraising for some organizations
    b. Fundraising for individual
4. User select “Fundraising for individual” and click “Next” button
5. System checks the existence of information, information existed
6. System check on user privilege level and current user privilege level is allowed
7. System save information into Campaign object form and forward both Campaign object form and user to the Create campaign add cover page
8. User add a cover photo of their campaign by Choosing “Upload photo” button

9. System display added photo
10. User click on “Next” button
11. System check existence of cover photo data, cover photo exists
12. System store photo in Campaign object form that sent from previous page and pass Campaign object form to the Create campaign story page
13. System forward user to Create campaign story page
14. User click on the save button
15. System check existing campaign story, campaign story data does not exist
16. System display error message

Use Case: View owned campaign

Actor: Campaign creator

Goal: Let user view campaign that user owned

Precondition: User must login with campaign creator or higher privilege level

Main success scenario:

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it

Alternative scenario 1 - No campaign

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user but no campaign found

4. System display no campaign

Alternative scenario 2 - User updates campaign

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it
4. User select campaign
5. System received campaign id and query campaign depends on received campaign id and store it Campaign object form.
6. System forward user to view campaign page and display campaign
7. User click “Update campaign” button to update the campaign
8. System forward user to Update page and display campaign detail from Campaign object form
9. User write down an update of campaign by clicking “Text” button if user want to add text into the campaign story or “Upload Image” button if user want to add image into the campaign story
10. System get user option and detail and save into Campaign object form that sent from previous page
11. System display given text or image
12. Repeat step 9-11 until user click on the save button
13. System check existence of campaign update data, campaign update data exist
14. System store all photo data in directory and save path in Campaign object form
15. System store data from Campaign object form except photo into the database

16. System display success message

Alternative scenario 3 - User didn't write down an update

1. Click user icon on the top right and choose "Manage campaigns"
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it
4. User select campaign
5. System received campaign id and query campaign depends on received campaign id and store it Campaign object form.
6. System forward user to view campaign page and display campaign
7. User click "Update campaign" button to update the campaign
8. System forward user to Update page and display campaign detail from Campaign object form
9. User click on "Save" button
10. System check existing campaign update, campaign update data does not exist
11. System display error message

Alternative scenario 4 - Edit campaign

1. Click user icon on the top right and choose "Manage campaigns"
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it
4. User select campaign
5. System received campaign id and query campaign depends on received campaign id and store it Campaign object form.
6. System forward user to view campaign page and display campaign
7. User click "Edit campaign" button to edit the campaign

8. System forward user to Edit page and display campaign detail from Campaign object form
9. User click on “Overview” on navigation tab
10. System get campaign name and goal from Campaign object form and display
11. User edit Campaign title and Goal and clicking on “Save” button
12. System check input Campaign title and Goal exist or not, Campaign title and Goal exist
13. System change Campaign title and Goal in Campaign object form
14. System change user information on database depends on Campaign object form
15. System displays success message

Alternative scenario 5 - User didn’t insert any information in overview edit

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it
4. User select campaign
5. System received campaign id and query campaign depends on received campaign id and store it Campaign object form.
6. System forward user to view campaign page and display campaign
7. User click “Edit campaign” button to edit the campaign
8. System forward user to Edit page and display campaign detail from Campaign object form
9. User click on “Overview” on navigation tab

10. System get campaign name and goal from Campaign object form and display
11. User edit Campaign title and Goal and clicking on “Save” button
12. System check input Campaign title and Goal exist or not, Campaign title and Goal does not exist
13. System displays error message

Alternative scenario 6 - User edit campaign cover photo

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it
4. User select campaign
5. System received campaign id and query campaign depends on received campaign id and store it Campaign object form.
6. System forward user to view campaign page and display campaign
7. User click “Edit campaign” button to edit the campaign
8. System forward user to Edit page and display campaign detail from Campaign object form
9. User click on “Cover photo” on navigation tab
10. System display photo from path inside Campaign object form
11. User click “Change” button
12. System get picture path and save new photo instead of old photo
13. System displays success message

Alternative scenario 7 - User delete campaign

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page

3. System query campaign(s) that owned by this user and display it
4. User click “Delete” button on select campaign
5. System set Deleted Flag for this campaign to 1

Alternative scenario 8 - User edit campaign story

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it
4. User select campaign
5. System received campaign id and query campaign depends on received campaign id and store it Campaign object form.
6. System forward user to view campaign page and display campaign
7. User click “Edit campaign” button to edit the campaign
8. System forward user to Edit page and display campaign detail from Campaign object form
9. User click on “Story” tab on the navigation tab
10. System query information from Campaign object form and display campaign story
11. User add new story append new story to the story detail or remove some parts of the story detail
12. System display edited campaign story
13. User click “Save” button
14. System check input campaign data exist or not, input campaign data exist
15. System change Campaign story on Campaign object form

16. System change Campaign story information on database depends on Campaign object form
17. System display success message

Alternative scenario 9 - User remove all Campaign story details

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query campaign(s) that owned by this user and display it
4. User select campaign
5. System received campaign id and query campaign depends on received campaign id and store it Campaign object form.
6. System forward user to view campaign page and display campaign
7. User click “Edit campaign” button to edit the campaign
8. System forward user to Edit page and display campaign detail from Campaign object form
9. User click on “Story” tab on the navigation tab
10. System query information from Campaign object form and display campaign story
11. User remove all campaign story details
12. System display edited campaign story
13. User click “Save” button
14. System check input campaign data exist or not, input campaign data does not exist
15. System display error message

Use Case: Donate money

Actor: User

Goal: Let user make a donation and contribute to a campaign via Stellar

Precondition: -

Main success scenario:

1. User login to system
2. System displays all campaign
3. User select campaign
4. System query campaign by selected campaign and store in a Campaign object form
5. System forward user to the campaign page and display campaign data from Campaign object form
6. User click on “Donate” button
7. System forward user to donate page
8. User select on “Via our website” button and insert fill in financial data and click on “Donate” button
9. System check existence of input data, input data exist
10. System store input data into Transaction object form
11. System build up transaction and send to Stellar network through Horizon API
12. System receive success message from Stellar network and then store Transaction object form data in database
13. System display success message

Alternative scenario 1 - User donate via existing wallet (Third party Stellar supported wallet)

1. System displays all campaign
2. User select campaign
3. System query campaign by selected campaign and store in a Campaign object form
4. System forward user to the campaign page and display campaign data from Campaign object form
5. User click on “Donate” button
6. System forward user to donate page
7. User select on “Existing wallet”
8. User insert transfer amount and click “Submit” button
9. System displays public key and QR code of the campaign owner
10. System store amount of money into Transaction object form and listening to Stellar network transaction that coming into the owner of this campaign
11. System found transaction that has the same amount of money as amount of money data in the Transaction object form
12. System store that transaction data into the database

Alternative scenario 2 - User choose to donate anonymously

1. User login to the system
2. Select campaign
3. System query campaign by selected campaign and store in a Campaign object form
4. System forward user to the campaign page and display campaign data from Campaign object form
5. User click on “Donate” button

6. System forward user to donate page
7. User select on “Via our website” button and insert fill in financial data and click on “Donate” button
8. User click on “Donate anonymously” choice and then click “Donate” button
9. System check existence of input data, input data exist
10. System store input data into Transaction object form and set anonymous flag to 1
11. System build up transaction and send to Stellar network through Horizon API
12. System receive success message from Stellar network and then store transaction data in database
13. System display success message

Alternative scenario 3 - System receive failure message from Stellar network

1. Select campaign
2. System query campaign by selected campaign and store in a Campaign object form
3. System forward user to the campaign page and display campaign data from Campaign object form
4. User click on “Donate” button
5. System forward user to donate page
6. User select on “Via our website” button and insert fill in financial data and click on “Donate” button

7. User click on “Donate anonymously” choice and then click “Donate” button
8. System check existence of input data, input data exist
9. System store input data into Transaction object form and set anonymous flag to 1
10. System build up transaction and send to Stellar network through Horizon API
11. System receive failure message
12. System resend transaction by Horizon API
13. If operation failed more than 5 times, System display error message to user

Alternative scenario 4 - User didn’t insert financial information

1. Select campaign
2. System query campaign by selected campaign and store in a Campaign object form
3. System forward user to the campaign page and display campaign data from Campaign object form
4. User click on “Donate” button
5. System forward user to donate page
6. User select on “Via our website” button and insert fill in financial data and click on “Donate” button
7. User click on “Donate anonymously” choice and then click “Donate” button
8. System check existence of input data, input data does exist
9. System displays error message

Use Case: Track financial activity of beneficiary

Actor: User

Goal: Let user see the history transaction that send to this campaign

Precondition: -

Main success scenario:

1. Select campaign
2. System query campaign by selected campaign and store in a Campaign object form
3. System forward user to the campaign page and display campaign data from Campaign object form
4. User click on “Donors” tab on the navigation tab
5. System query transaction that owned by this campaign and store in Transaction object form
6. System request for details of transaction by transaction hash inside each Transaction object form through Horizon API
7. System received details of transaction and display

Use Case: View donation campaign

Actor: User

Goal: View campaign information

Precondition: -

Main success scenario:

1. System displays all campaign
2. User select campaign

3. System query campaign by selected campaign and store in a Campaign object form
4. System forward user to the campaign page and display campaign data from Campaign object form

Alternative scenario 1 - User didn't send report without any data

1. System displays all campaign
2. User select campaign
3. System query campaign by selected campaign and store in a Campaign object form
4. System forward user to the campaign page and display campaign data from Campaign object form
5. User click on "Report" button and insert reason and click on "Submit" button
6. System check existence of report data, report data does not exist
7. System display error message

Use Case: Report campaign

Actor: User

Goal: Report campaign to admin

Precondition: User must sign in to the system

Main scenario:

1. System displays campaign
2. User select campaign
3. System query campaign by selected campaign and store in a Campaign object form

4. System forward user to the campaign page and display campaign data from Campaign object form
5. User click on “Report” button and insert reason and click on “Submit” button
6. System check existence of report data, report data exist
7. Store report data in Report object form and into database

Use Case: Search campaign

Actor: User

Goal: Search for campaign

Precondition: -

Main success scenario:

1. Type campaign name on search bar
2. Click on Search button
3. System search input by name, category, and organization in database
4. System query campaign that matched
5. System display campaign(s)

Use Case: View donation history

Actor: User

Goal: To view donation that user donated

Main success scenario:

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query transaction(s) that owned by this user and display it

Alternative scenario 1 - View receipt

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query transaction(s) that owned by this user and display it
4. User click on “View a receipt” button
5. System query PDF file of this transaction and display

Alternative scenario 2 - Request a receipt

1. Click user icon on the top right and choose “Manage campaigns”
2. System forward user to Manage campaigns page
3. System query transaction(s) that owned by this user and display it
4. User click on “Request a receipt” button
5. System query PDF file of this transaction and sent to user’s email

Use Case: Account setting

Actor: User

Goal: To edit user account

Main success scenario:

1. Click user icon on the top right and choose “Account setting”
2. System forward user to Account setting page and query user information by user id and save in User object form
3. System displays user information from User object form
4. User edit account information and then click “Save” button
5. System check existence of input data, input data exist
6. System save new input data into User object form
7. System save new data in User object form into the database

Alternative scenario 1 - Identity verification

1. Click user icon on the top right and choose “Account setting”

2. System forward user to Account setting page and query user information by user id and save in User object form
3. User click on “Activate” button
4. System displays information for identity verification
5. User specify identity verification information and click “Submit” button
6. System set Verification Request Flag inside User object form to 1
7. System get User object form information and save into the database
8. System displays user identity verification request to admin
9. Admin approved user identity verification
10. System set user privilege level up to 2 and set Verification Request Flag back to 0
11. System sends identity verification success to user’s email

Use Case: Approve user identity verification

Actor: Admin

Goal: Verify user identity

Precondition: User must login as admin account

Main success scenario:

1. Login to admin account
2. Click on verify user identity button
3. Select user identity verification that user want to approve
4. Approve user identity
5. System set user privilege level up to 2 and set Verification Request Flag back to 0
6. System sends identity verification success to user's email

Use Case: View fraud report

Actor: Admin

Goal: View all fraud report from users

Precondition: User must login as admin account

Main success scenario:

1. Login to admin account
2. Click on view fraud report button
3. System query fraud report(s) from database and display

Use Case: Inactivate campaign

Actor: Admin

Goal: To inactive some campaign and hide from other user

Precondition: User must login as admin account

Main success scenario:

1. Login to admin account
2. Click on View campaign button
3. System query fraud campaign(s) from database and display
4. User click on “Inactivate” button
5. System set campaign Active Flag to 0

Appendix B

User Interface Design Detail

User interface design

This section will show mockups of our user interface design. Some detail in the mockups might change in the final product.

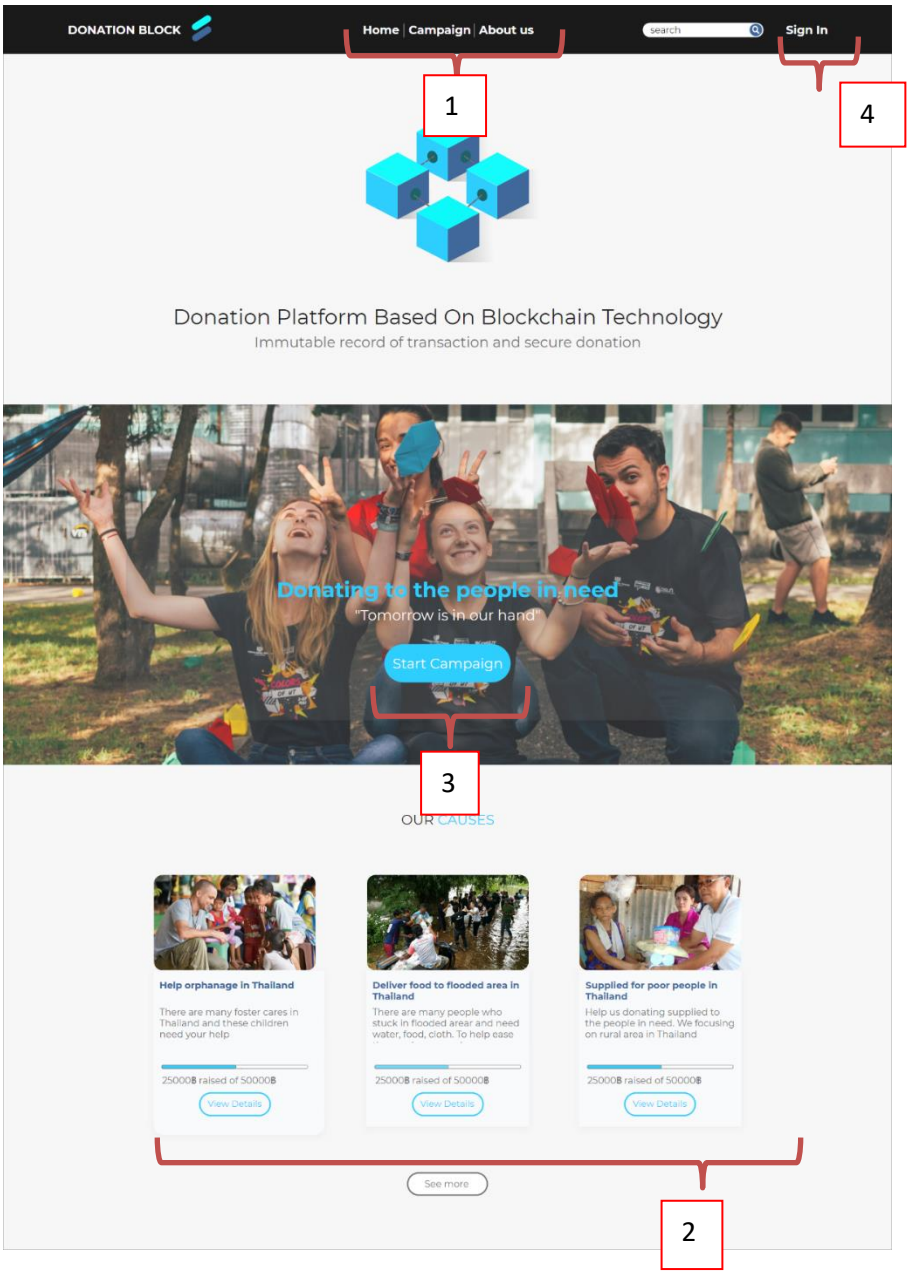


Figure 1: Home page

Figure 1 shows the home page of our website. It consists of navigation bar on the top, followed by the bottom part which is donation campaigns.

1. The navigation menu consists of “Home” button, “Campaign” button, “Create Campaign” button, and “About us” button. The buttons will bring user to Home page, Campaign page, and About us page respectively.
2. The bottom part which is donation campaigns will show the three most recently added campaigns.
3. The “Start Campaign” will also routing user to the Create Campaign page and if user haven’t sign in to our system, instead it will redirect user to the sign-up page first.
4. It is a “Sign in” button that will redirect user to the Sign in page.

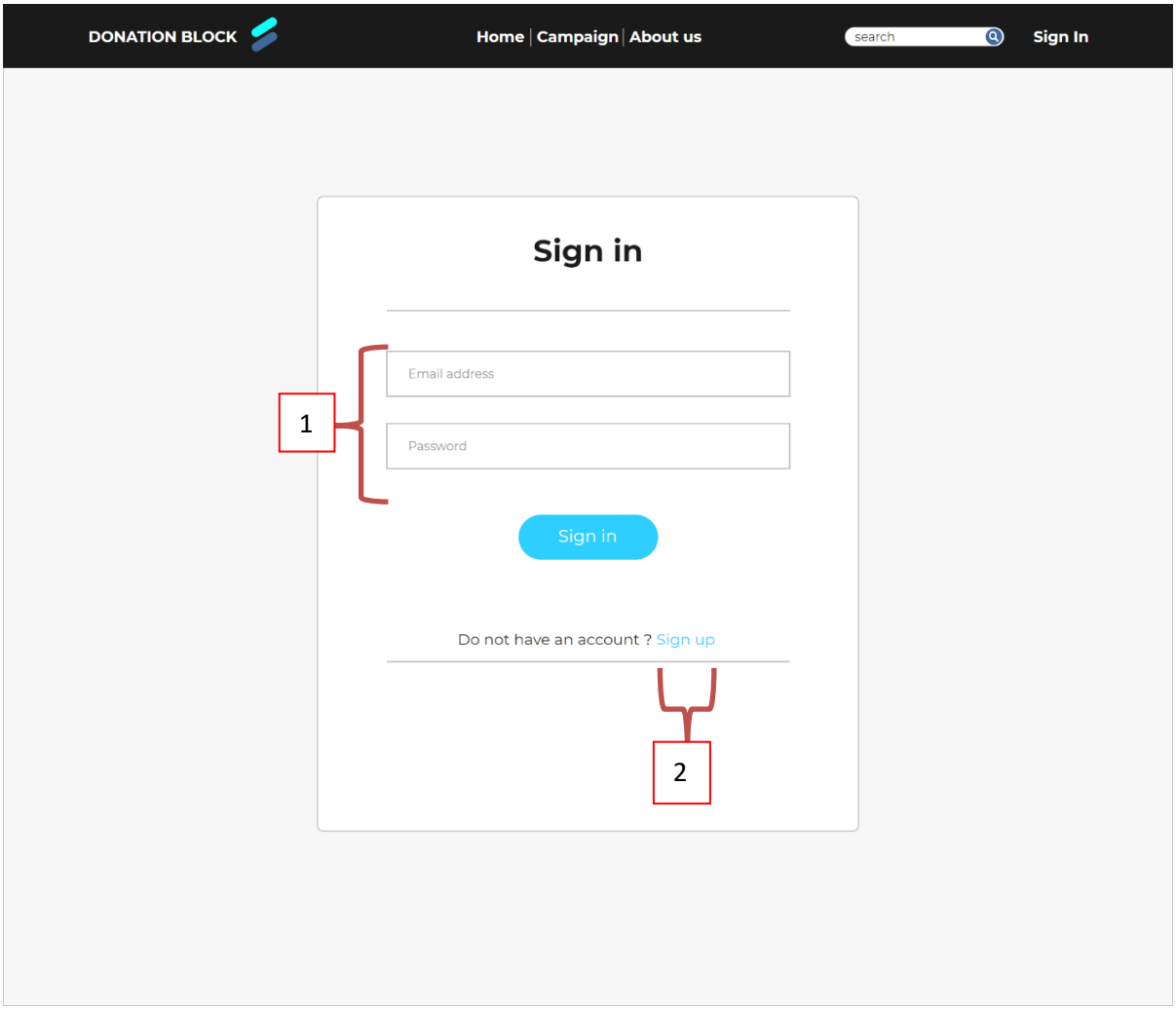


Figure 2: Sign in page

Figure 2 shows the sign in process to our system. In order to access this page user must click sign in button from Home page first. To sign in to our website user must already sign up to our website.

1. In this page, user must fill in Email address and Password in order to sign in to the system.
2. If user haven't sign up to our website yet they can click "Sign up" button to sign up first before sign in.

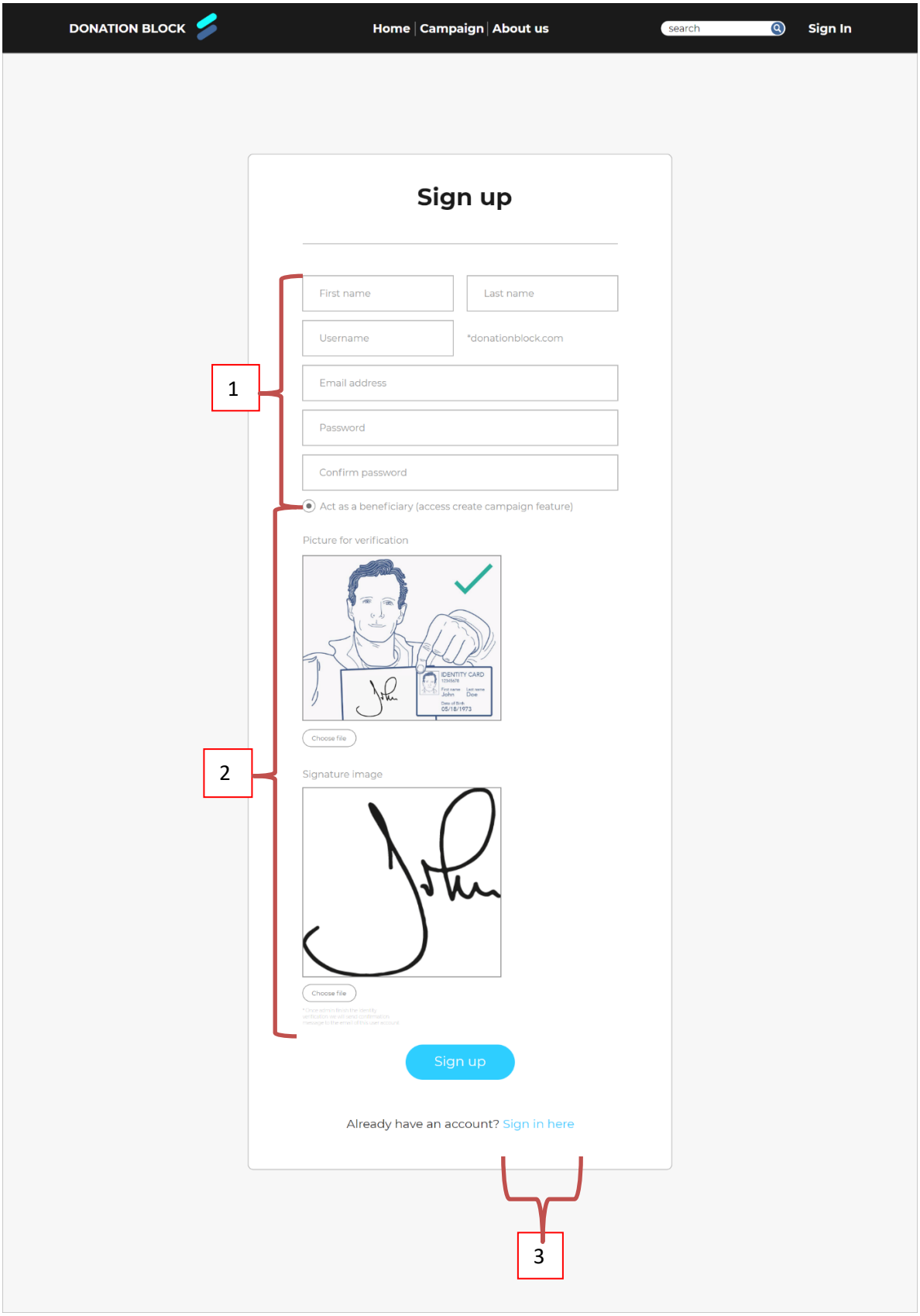


Figure 3: Sign up page

Figure 3 shows the sign-up process to our system. In order to access this page, users must click the sign-up button from Sign in page first. To sign up to our website users must already have an email address.

1. In this page, the user must fill in the First name, Last name, Username, Email address, Password, and confirm password. Then the user must click the sign-up button to sign up to the system.
2. In this section, if a user wants to access beneficiary features like Create Campaign and Manage campaigns, the user must select the “Act as a beneficiary” radio button. There will be a form for users to upload pictures for verification and their signature and wait for an admin to verify.
3. If user already sign up to our website, they can click “Sign in” button to go back to Sign in page.

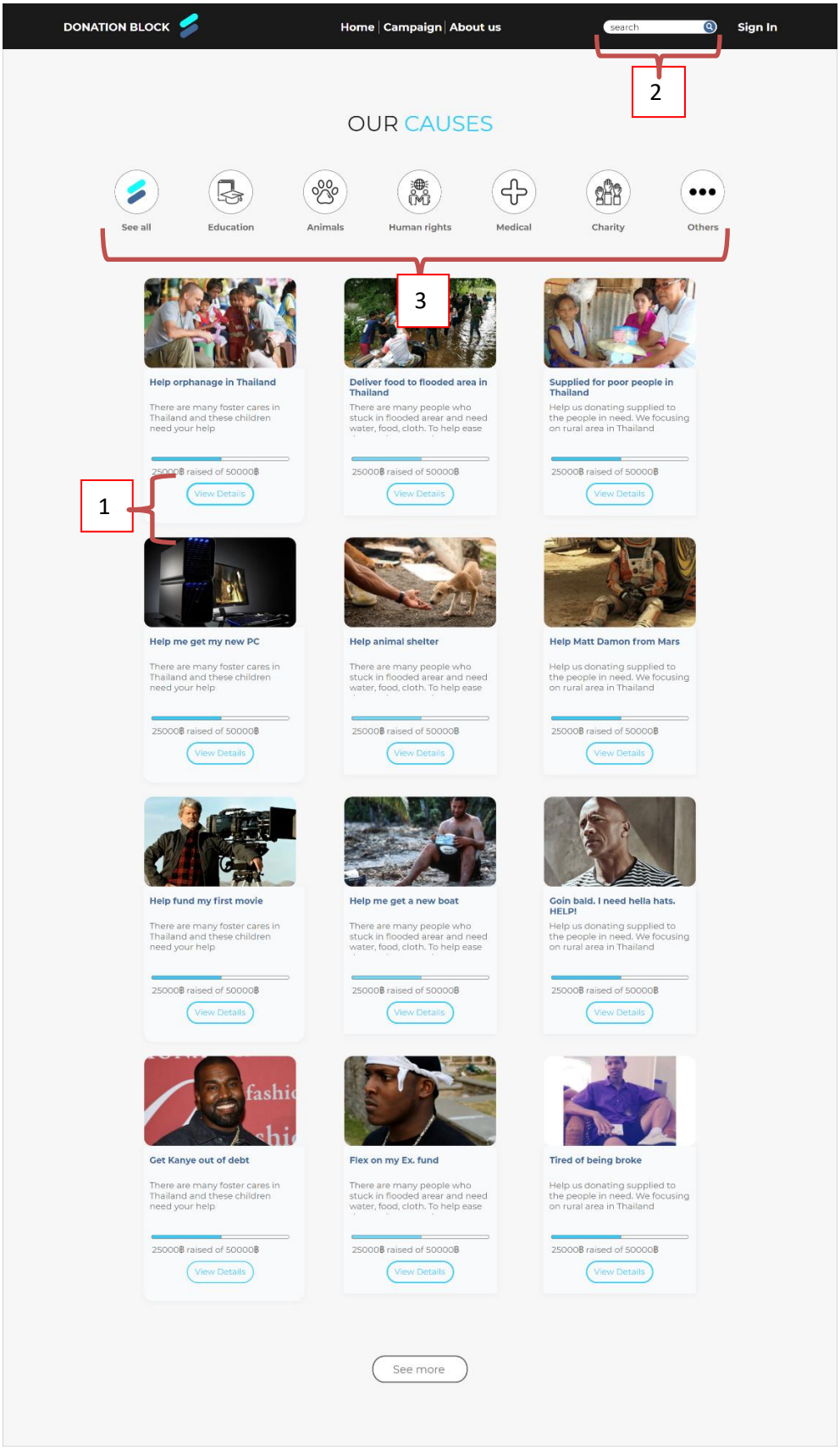


Figure 4: View Campaigns page

Figure 4 shows the campaigns page of our website. It consists of campaigns that register to our website.

1. When user click “View Details” button, it will redirect user to that campaign information and donation page
2. User can search campaign by category of campaign, specific organization, and specific beneficiary
3. User can filter from category like See all, Education, Animals, Human rights, Medical, Charity, and Others

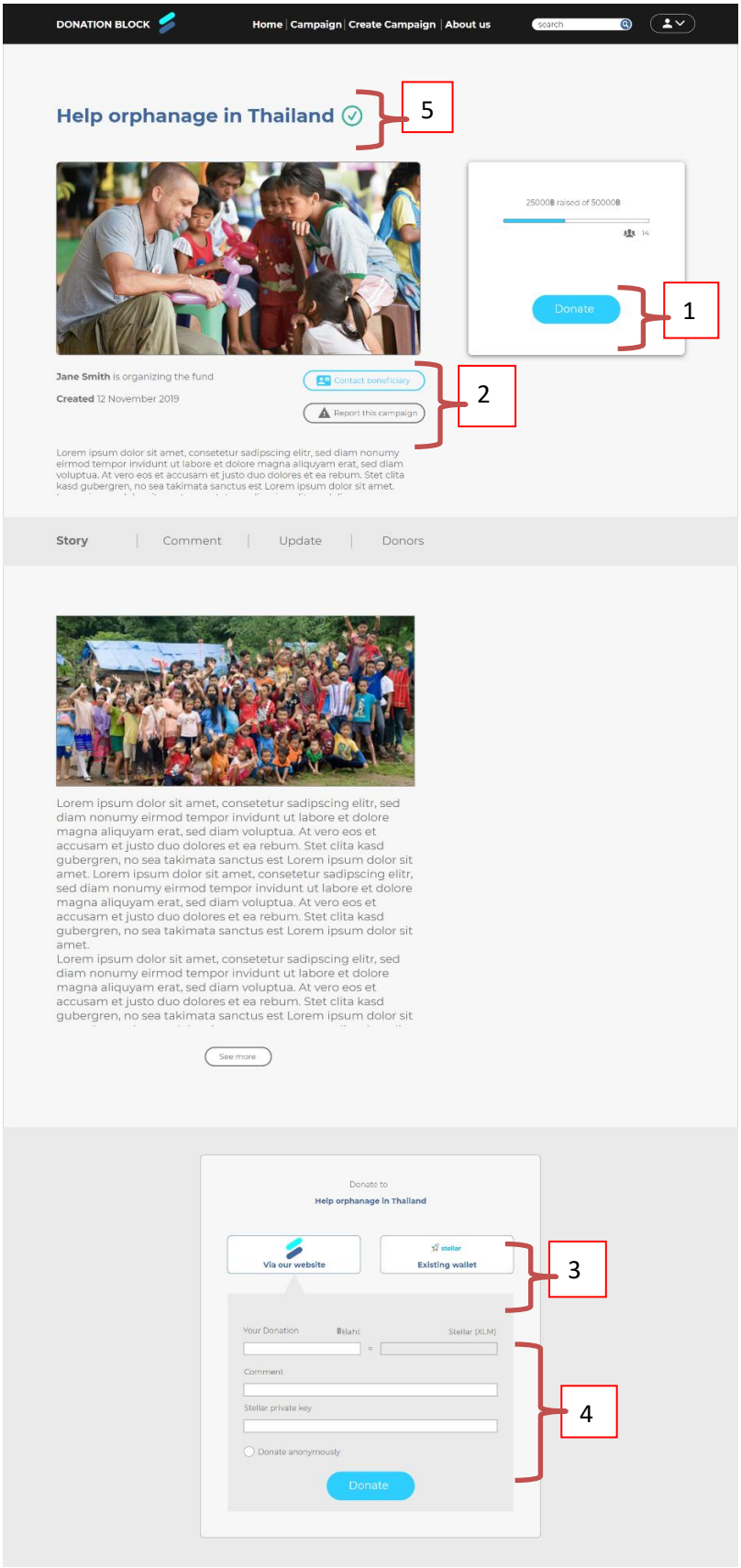


Figure 5: Campaign Information and donation via our website page

Figure 5 shows the Campaign Information of a campaign that donors will see. This page can be accessed when the user clicks one of the campaigns. This page consists of title, cover picture, campaign story, comment section, their update, and donors list. At the bottom of the page is the donation section. The donation section, which is located at the bottom part

1. User can click “Donate” button and it will bring the user to the donating section at the lower part of the page
2. User can click “Contact beneficiary” button or “Report the campaign” button to contact or report that campaign respectively
3. User can choose whether they want to donate using our website wallet (simulated) or choose to donate via other existing Stellar wallets
4. In case when user want to donate using our website wallet, they must fill in the amount of money in baht and there will be a box that converts the amount to lumen, comment for that campaign, their Stellar private key (which we will not store it in our database), and they can also choose to donate anonymously by clicking “Donate anonymously” radio button
5. The green check mark will show if this is an official campaign (created by an organization) or an unofficial campaign (created by an individual). If the campaign created by an individual, it will not show the green check mark

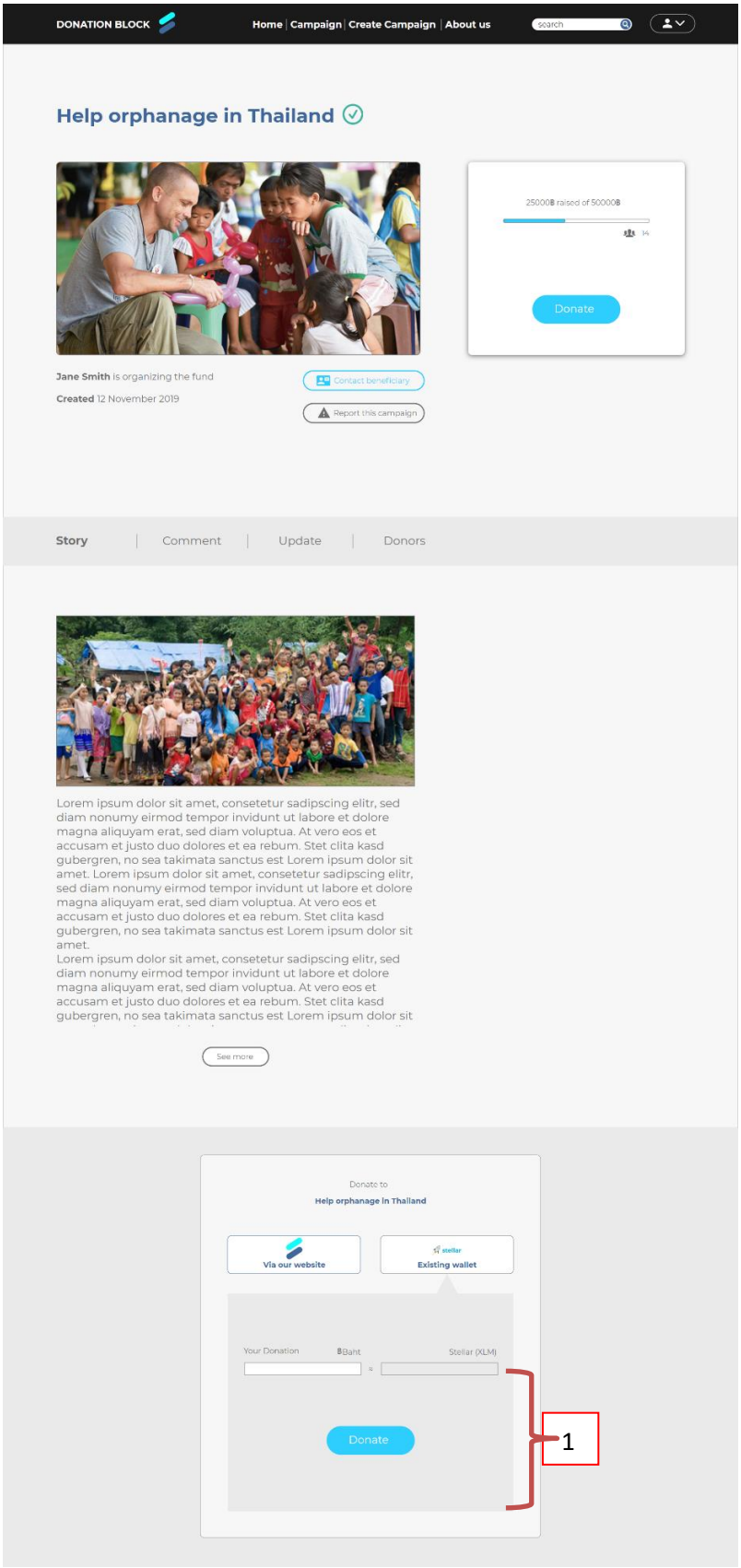


Figure 6: Campaign Information and donation via existing wallet step-1 page

Figure 6 is like “Campaign Information and donation via our website page” but different donation option

1. In case when users want to donate using our website wallet, they must fill in the amount of money in baht and there will be a box to convert the amount to lumen. After that user can click “Donate” button to go to Campaign Information and donation via existing wallet page – step2

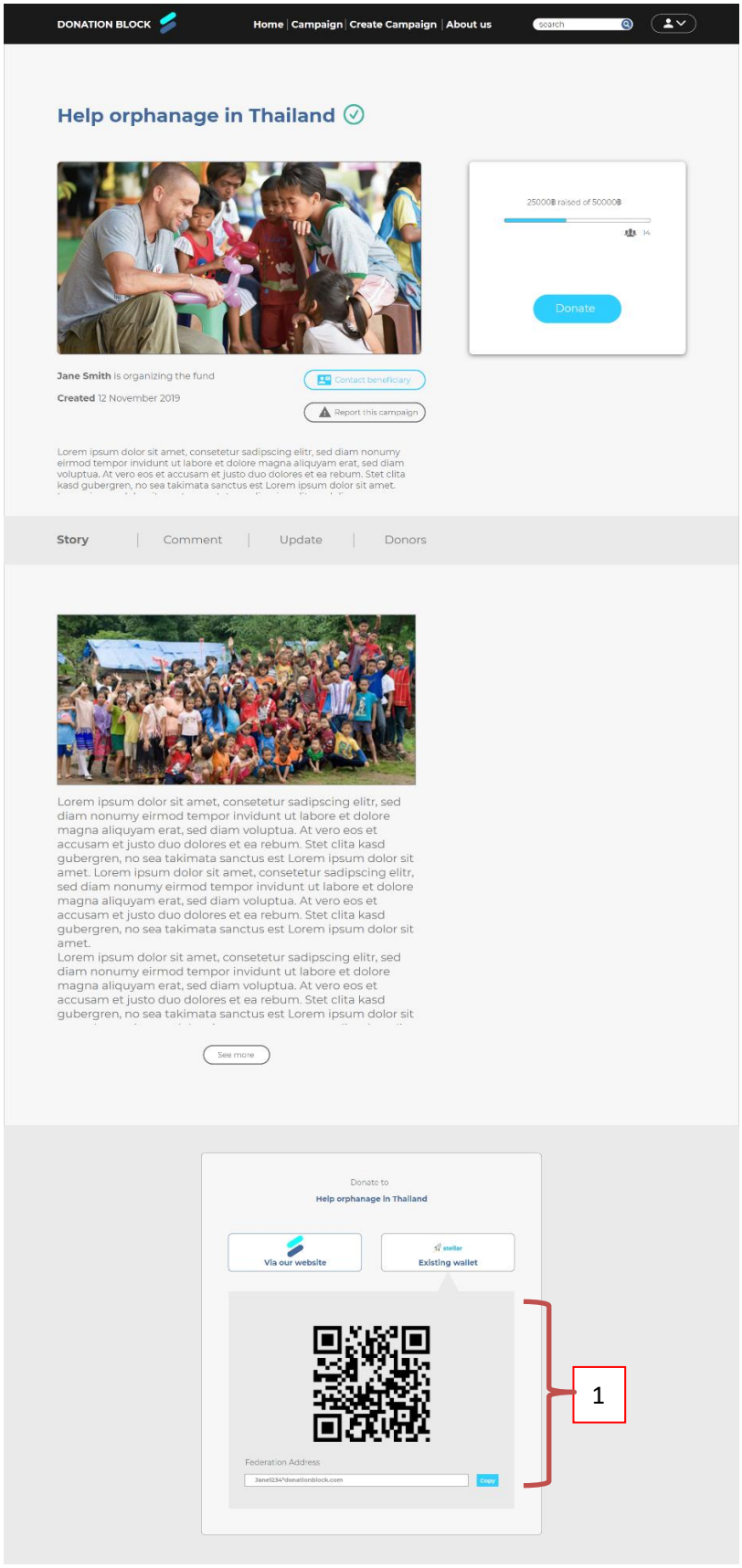


Figure 7: Campaign Information and donation via existing wallet step-2 page

Figure 7 is like “Campaign Information and donation via our website page” but different donation option

1. User can donate via their existing Stellar support wallet by using QR code or use beneficiary’s federation address

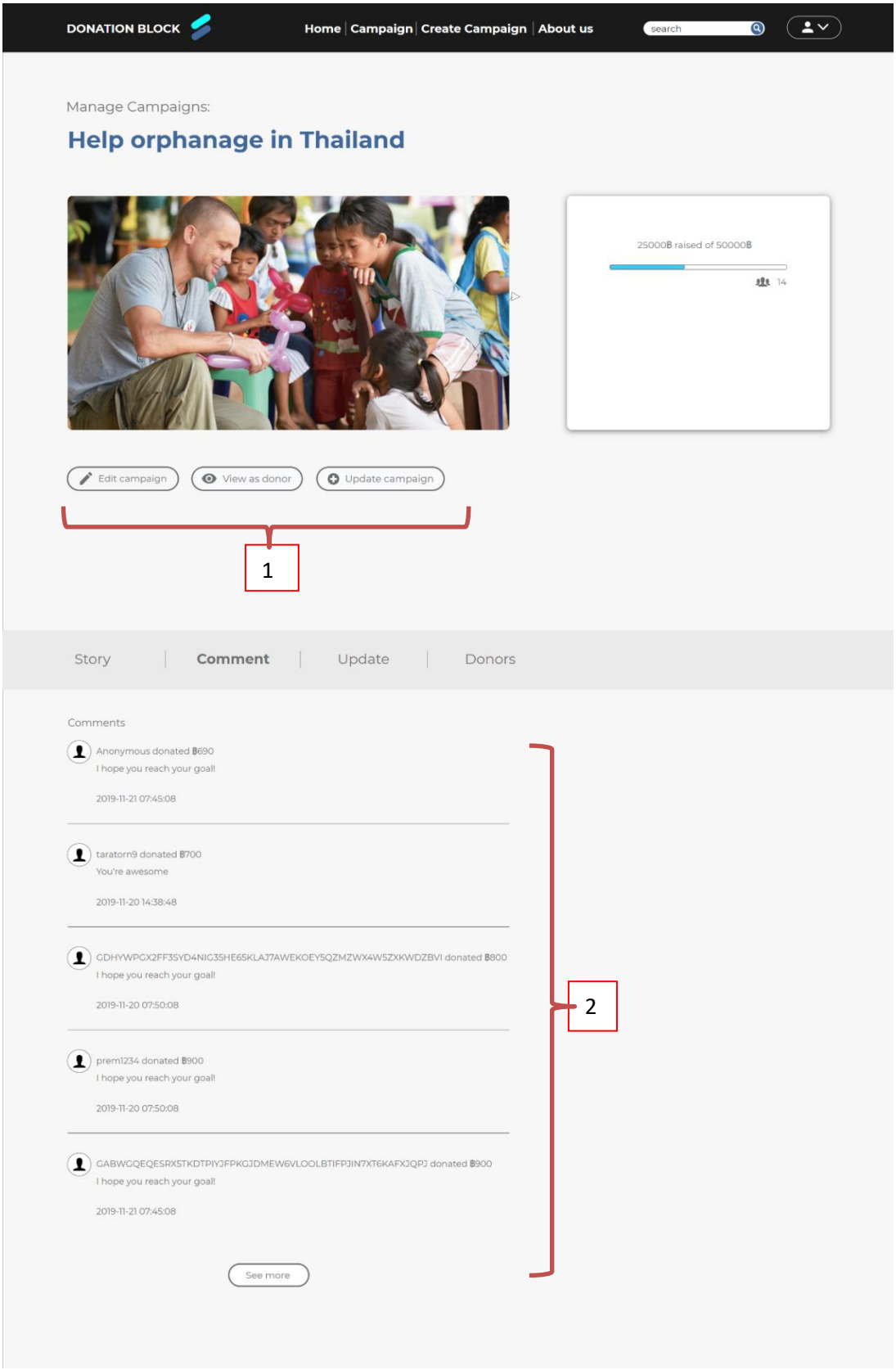


Figure 8: Edit campaign – Comment section page

Figure 8 shows the Campaign Edit of a campaign from beneficiary point of view. This page can be accessed when the user clicks one of the campaigns from the Manage campaign page.

1. Consists of “Edit campaign” button, “View as donor” button, “Update campaign” button. In the “Edit campaign” button, campaign creators can change their title, goal, cover photo, and their story. In “Update campaign” button, campaign creator can update their campaign by using the same form to fill the information like in “Create campaign page step-3” page
2. The comment section consists of the donor's public key and if they choose to donate anonymously it will show as “anonymous”, their comment, and timestamp when they donate to the campaign. For the transaction that came from outside of our system, a transaction object that provided by Stellar can display only the public key. But the transaction that came from our system, we can display their username or stay anonymous depending on the user choice.

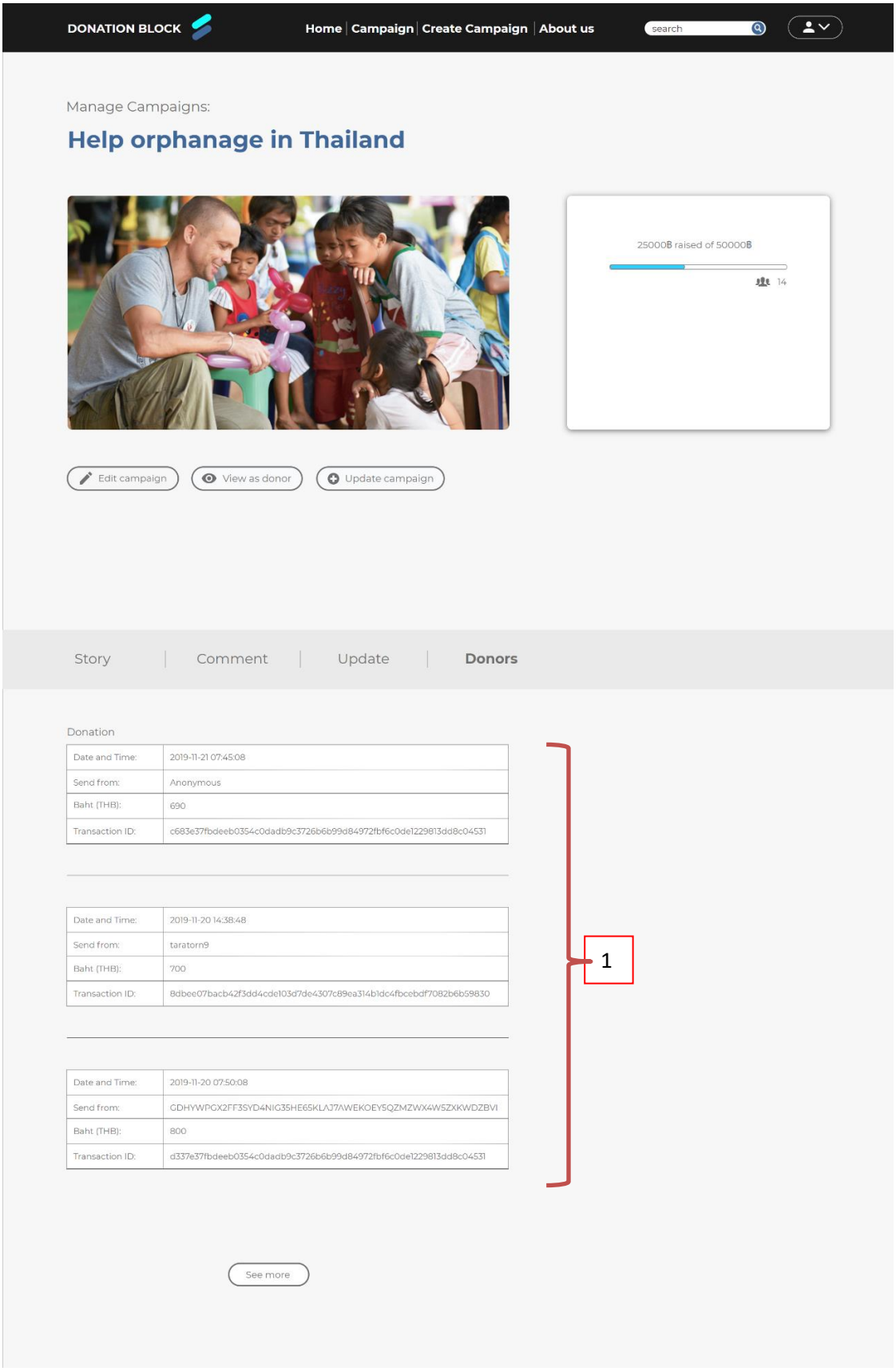


Figure 9: Edit campaign – Donors list section page

Figure 9 is like “Edit campaign – Comment section page” but this page will show the donors list section

1. At the donors list section consists of timestamp when they donate to the campaign, username of donors (public key in case a transaction came from outside of our system), the amount of money that they donate (Baht), and transactionID

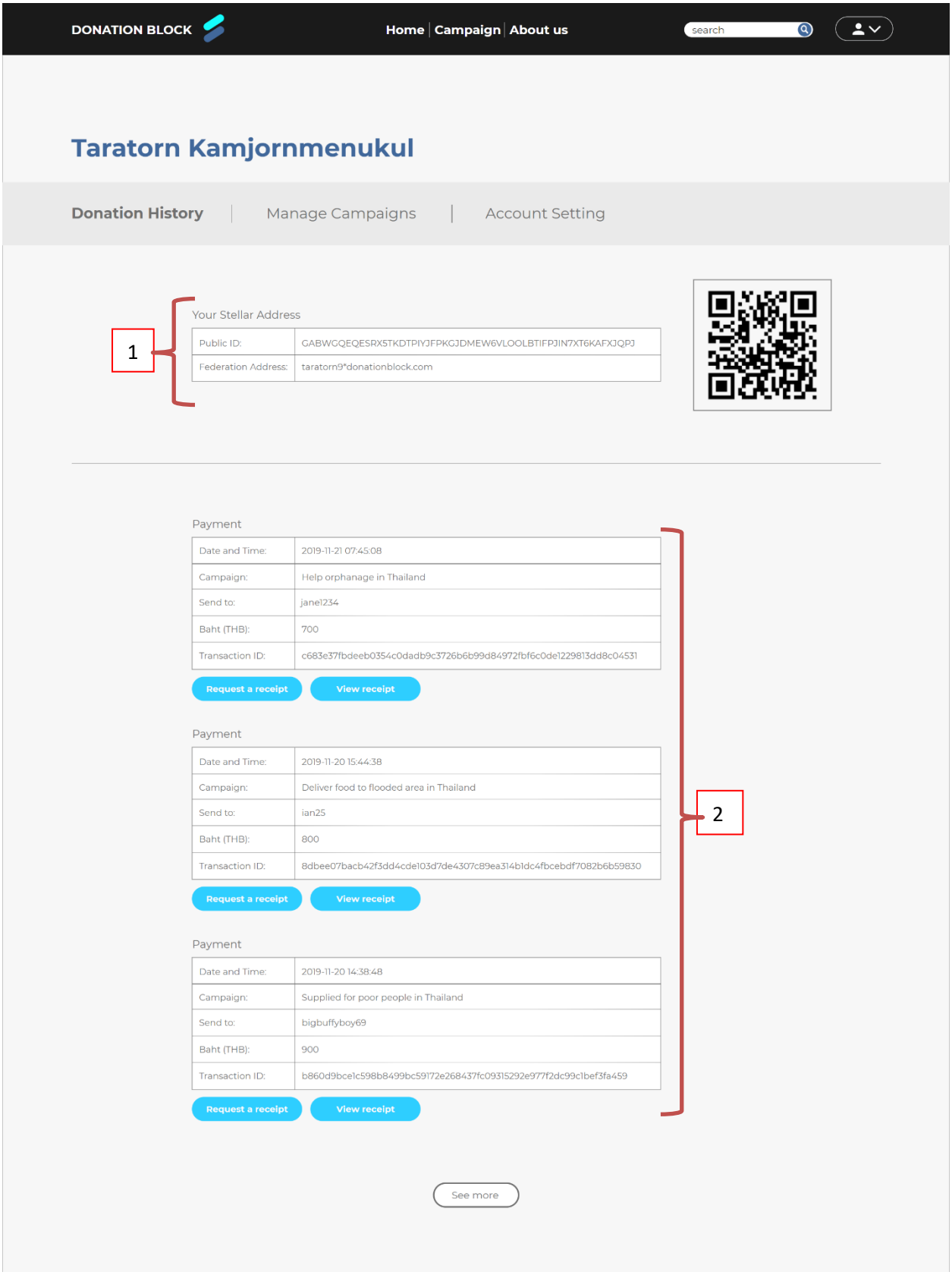


Figure 10: Donation history page

Figure 10, in each user account there will be 3 features which are “Donation History”, “Manage Campaigns”, and “Account Setting”. This figure shows the history donation page of a user.

1. It consists of publicID, federation address, and QR code of the publicID of that user.
2. It consists of each donation. For each donation, it consists of Date and Time of that donation, Campaign name, the username of the beneficiary, Amount of baht, transactionID, “Request a receipt” button, and “View receipt” button.

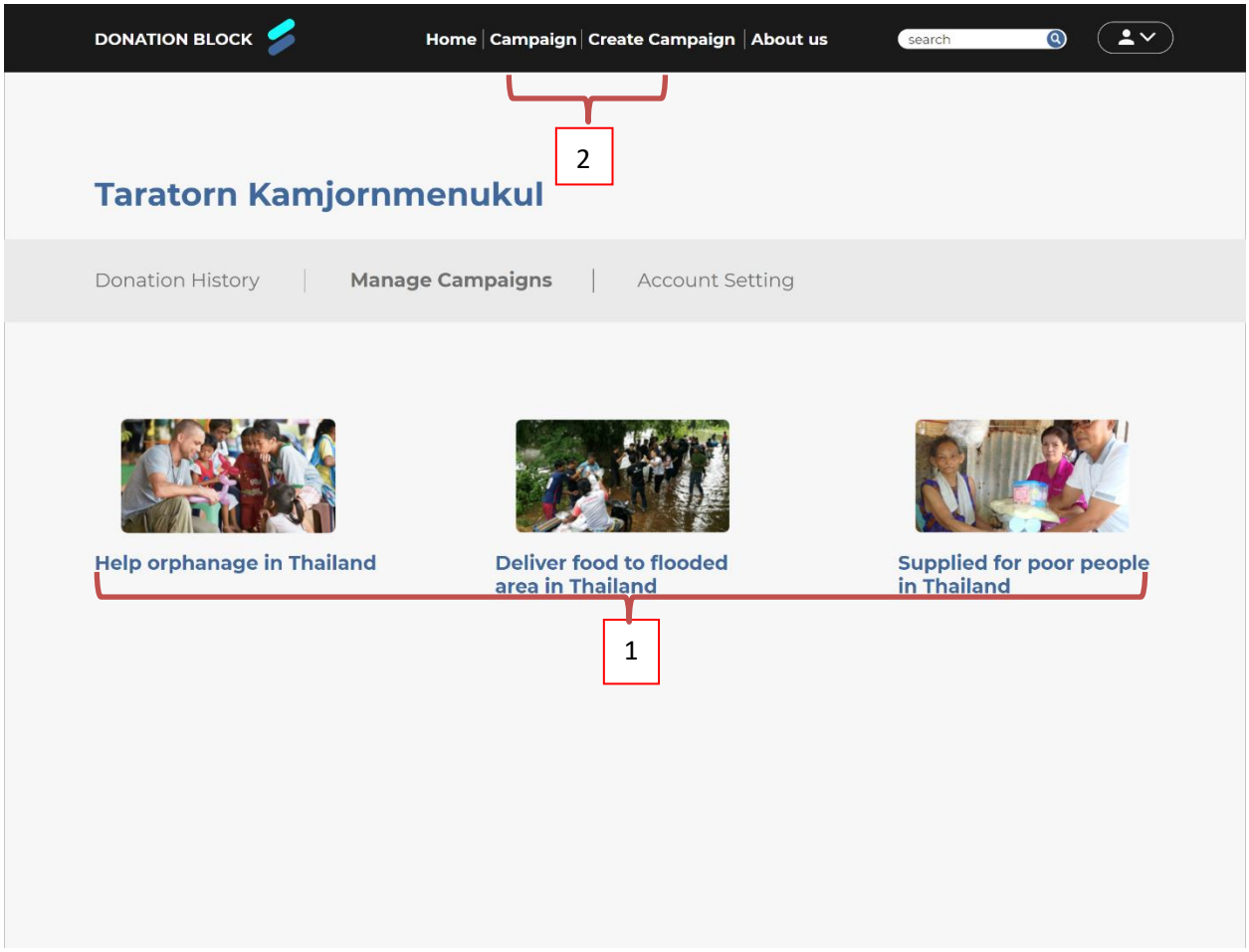


Figure 11: Manage campaign page

Figure 11 shows the managed campaign page of a user (User must get verification from admin first in order to access this feature).

1. It consists of campaigns that the user created. If click one of the campaigns, it will redirect user to the “Edit campaign” page of the campaign selected

2. “Create campaign” button and “Manage campaign” page can only access after user has been verified as beneficiary

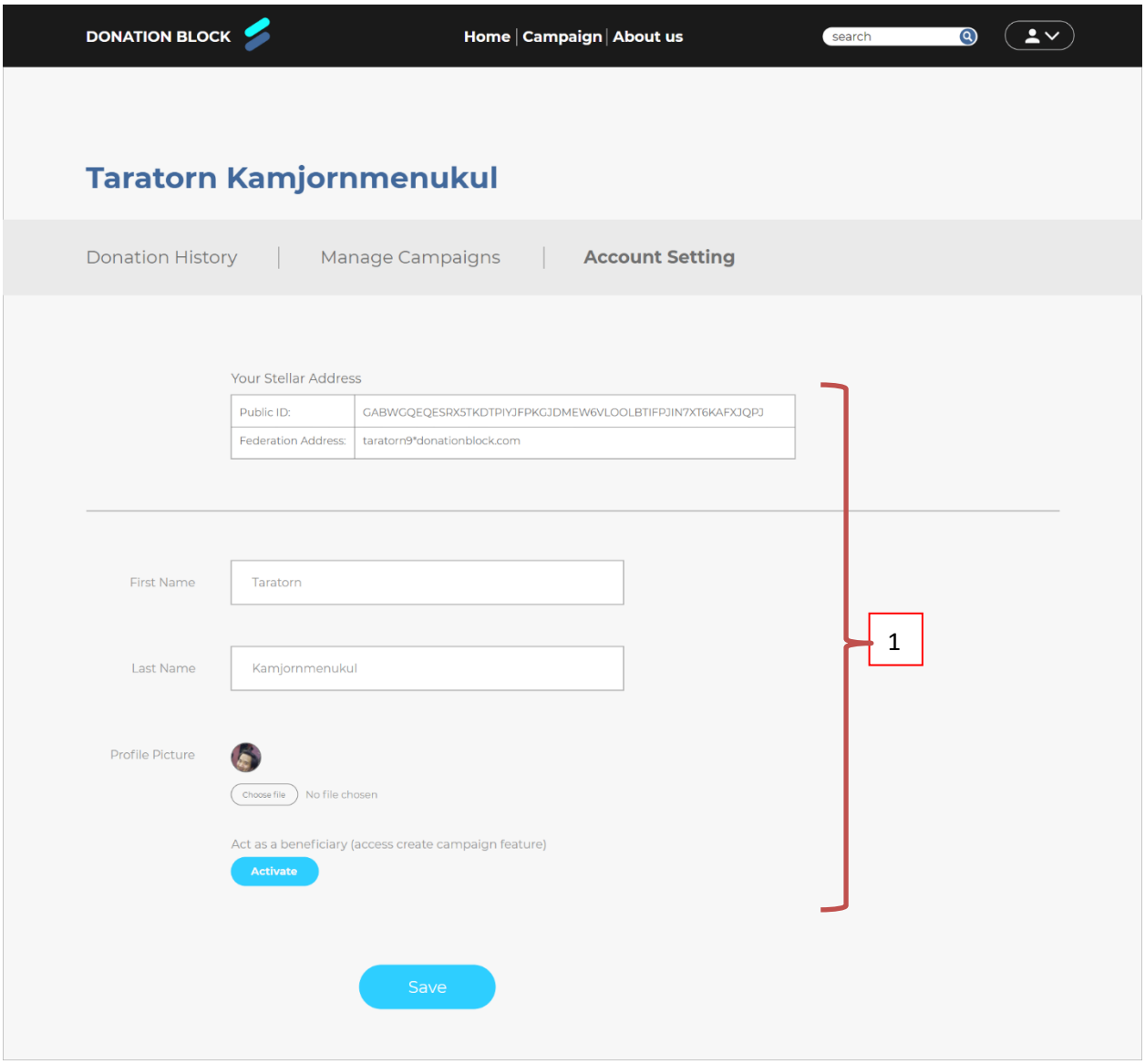


Figure 12: Account setting page

Figure 12 shows the account settings page of a user that the user can change.

1. It consists of FederationID, Public key, First name, Last name, and profile picture of the user. In case the user doesn't want to access the beneficiary feature when signing up, they can click the "Activate" button to fill up the information. There will be a form for users to upload picture for verification and their signature and wait for an admin to verify.

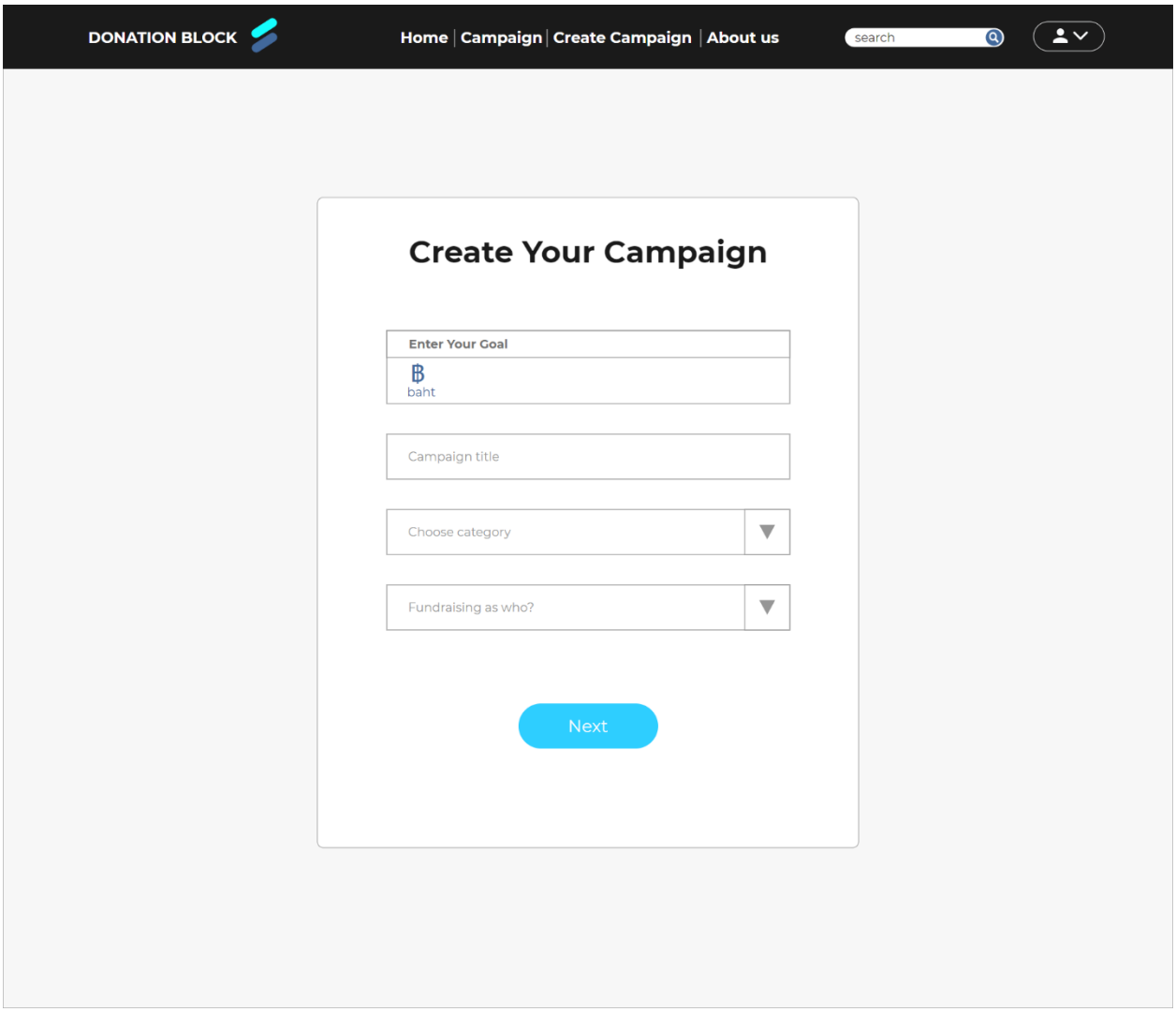


Figure 13: Create campaign page step-1 page

Figure 13 shows how to create a campaign. In step-1

1. It consists of fundraising goals in Baht, Campaign name, choosing category of the campaign, and fundraising as who? (Individual or organization)

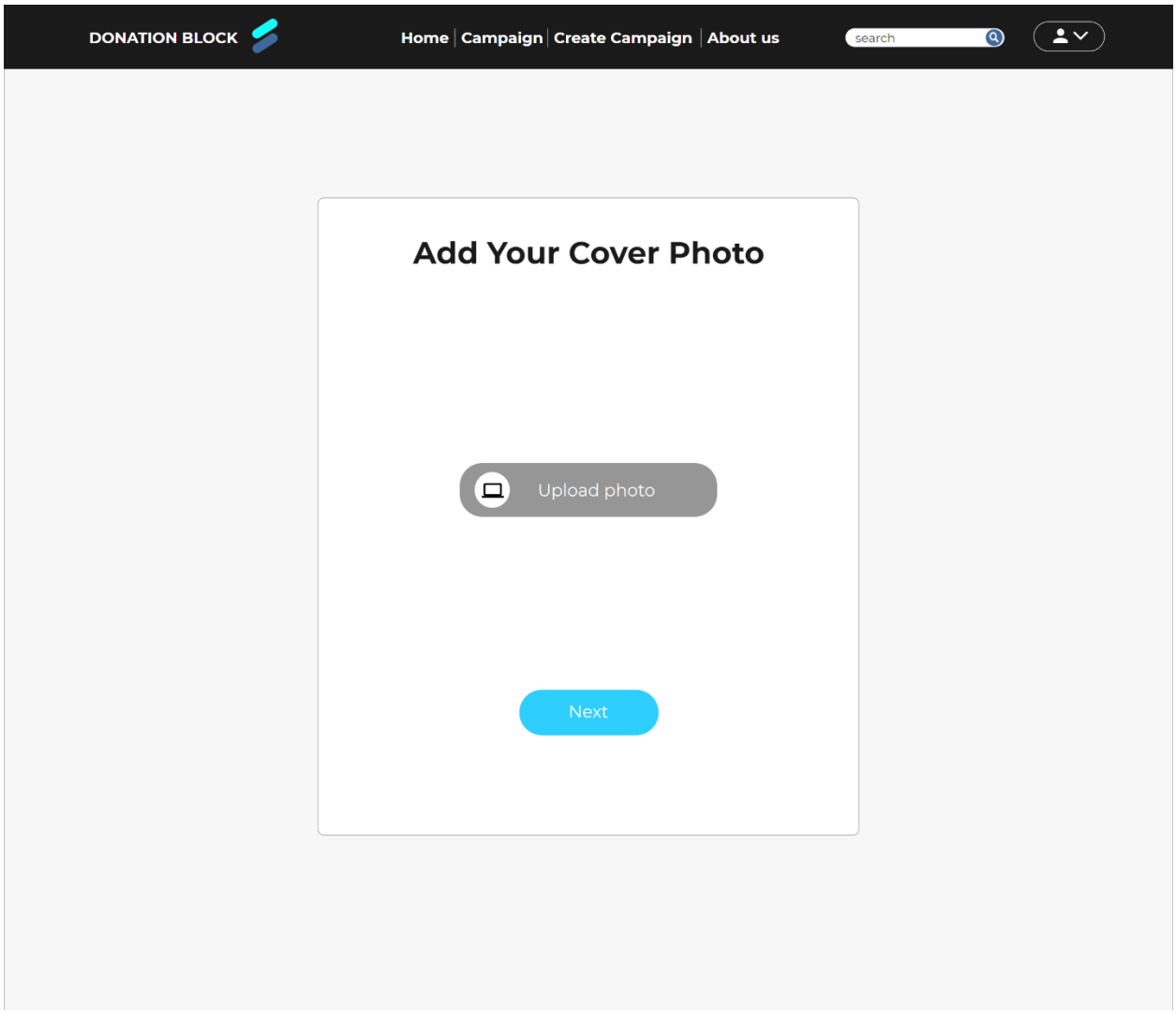


Figure 14: Create campaign page step-2 page

Figure 14 shows how to create a campaign. In step-2

1. It consists of Cover photo of user's campaign that user can upload from their computer

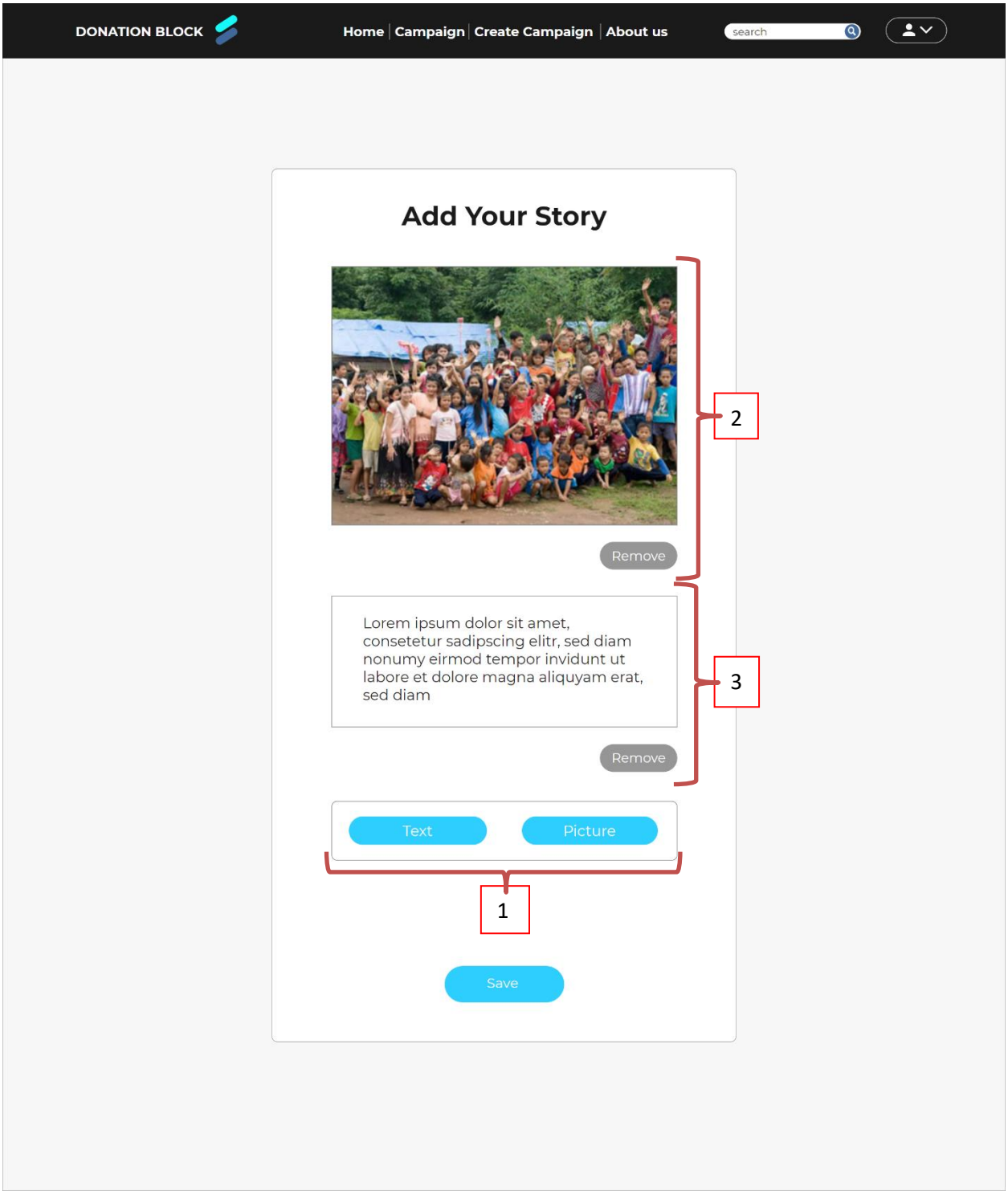


Figure 15: Create campaign page step-3 page

Figure 15 shows how to create campaign. In step-3, when click “Save” the campaign will be create

1. When user first reach this step, there will be only option box that user can choose to add text box or picture first
2. When a user chooses to add a picture first, they just click the “Picture” button and the user must choose a picture from their computer to upload. If they don’t like that picture, they can simply click “Remove” button to remove the picture
3. If a user chooses to keep the picture then want to add text next. They can simply click the “Text” button and there will be a textbox for users to fill in their story. And they can choose to add text or picture in any order