VALIDITY AND INTEGRITY CHECK IN SUPPLY CHAIN MANAGEMENT USING BLOCKCHAIN

<on the side rim printed in gold include <Validity and Integrity Check in Supply Chain Management using BlockChain> and A.Y. 2019 – 2020>

Validity and Integrity Check in Supply Chain Management using BlockChain

Submitted in partial fulfillment of the requirements
of the degree of

B. E. Computer Engineering

By

Pranav Gor	**60004160032**
Deep Gosalia	**60004160033**
Rahil Jhaveri	**60004160042**

Guide:

Prof. (Mrs.) Aruna Gawade
Associate Head & Assistant Professor

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

University of Mumbai
2019-2020

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

CERTIFICATE

This is to certify that the project entitled **Validity and Integrity Check in Supply Chain Management using BlockChain** is a bonafide work of **Pranav Gor (60004160032), Deep Gosalia (60004160033)** and **Rahil Jhaveri (60004160042)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering.

Prof. (Mrs.) Aruna U. Gawade
Project Guide

Dr. (Mrs.) Meera M Narvekar
Head of Department

Dr. Hari Vasudevan
Principal

Project Report Approval for B.E.

This project report entitled ***Validity and Integrity Check in Supply Chain Management using BlockChain*** by ***Pranav Gor, Deep Gosalia*** and ***Rahil Jhaveri*** is approved for the degree of ***B.E. in Computer Engineering.***

Examiners

1.----------

2.----------

Date:

Place:

Declaration

I/We declare that this written submission represents my/our ideas in my/our own words and where others' ideas or words have been included, I/We have adequately cited and referenced the original sources. I/We also declare that I/We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my/our submission. I/We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Pranav Gor - 60004160032

Deep Gosalia - 60004160033

Rahil Jhaveri - 60004160042

Date:

Abstract

Our project aims at improving the existing Blood supply chain management system. We propose an innovative blood collection chain system based on blockchain technology using a distributed ledger structure and transaction execution process of the situation where the collected blood reaches the final stage - hospitals. Existing centralized blood management systems have various drawbacks including lack of information on blood bags, inability to reflect real-time updates in details and the trust factor. In this respect, the blockchain technologies offer the possibility to maintain a transparent blood management system, especially since data cannot be counterfeited and tampered with. In addition, the system stores blood contracts between hospitals in the event of emergencies through the transaction and consensus flows. This approach is to allow hospitals that are far from the blood banks to accrue blood supply in emergency situations. Since blood has a certain storage period, it is expected that it will be possible to fulfill the demand of blood bags by supplying and using the bags that are nearing their expiration. Not only does it solve the primary purpose that is it helps to save lives, but it will also act as a waste management strategy for Blood by using near expiration bags first.

Keywords: Blockchain, Blood, Supply Chain Management, Distributed Ledger.

Table of Contents

List of Figures

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
1	SCM	Supply Chain Management
2	BC	BlockChain
3	ETH	Ethereum
4	CC	Crypto-Currency
5	UID	Unique Identifier
6	DP	Double Payment
7	SC	Smart Contracts
8	BCC	Blood Collection Center
9	T&P	Testing and Processing Center
10	SCT	Storage Center

CHAPTER 1

Introduction

1.1 Description

As a daily consumer ourselves, many a time we find ourselves in the wonderment, trying to figure out the entire lifecycle of a product when we purchase it. Right from its creation (manufacturing) process, until it is in our shopping cart. Every product that we purchase, is in turn connected to a chain of other products that are used as a basic set of ingredients in order to build the product we just purchased. Thus, emphasizing the fact that this intricate network of chain hierarchy is the basic building block for a product to be built and this is what we call Supply Chain. In order to ensure that the quality of the end product is up to the required mark, it depends on every single entity in the entire supply chain to be in place, in a timely manner whilst possessing good quality. It thus goes on to depict that proper management is required to handle the entire process, which is termed as Supply Chain Management (SCM).

In our project, using the Blockchain (BC) implementation, we aim at providing rather simplified yet quality-assured support for maintaining and monitoring the data from the collection of the Blood from various different sources and tracking the physical conditions of those blood bags as well.

1.2 Problem Formulation

While the current traditional SCM is hampered by some fundamental flaws like uncertainty and distrust among the network, there are some factors that are difficult to handle despite the fact of them being recognized. Firstly, for example, the wastage of blood occurs due to less demand for blood bags and acquiring a higher supply of blood. An organization cannot take risk of reducing the collection of blood, based on the assumption that since the current requirement is low, it will stay low in the future as well. An emergency can occur anytime, hence it calls for a decisive measure of proper management which can be done using a few variations as mentioned in our proposed BC-based SCM solution. Secondly, the factor of mistrust amongst the entities in the entire network can be a major discomfort for their constituents to function smoothly.

These posing challenges can be solved using our system. Our aim is to develop the BC implementation of the traditional SCM where traceability and efficient management is implied for the proper regulation of the entire distribution.

1.3 Motivation

Using traditional SCMs, when a product is launched into the market, the consumers are generally skeptical and may have some reservations with respect to the quality of the product. This factor of 'distrust' is a very common issue faced and is one of the major drawbacks of using traditional SCM. Moreover, this distrust factor is of primal importance when it is faced in the field of Blood donation and hence has to be rectified as early as possible. Thus there is a need of a system where a donor can satisfactorily get the whereabouts of the blood he donated and on the corollary, the recipient of the blood will also want to be assured that the blood that he is about to be injected with, is 100% free of any detrimental factors.

According to an article by the Times of India [1], there is a tremendous amount of blood being tagged as 'waste', approximating to nearly six lakh litres of blood over five years of span, in Maharashtra itself and is disposed of due to 'improper coordination between the blood banks and hospitals' with given justification that they expired prior to their use. This shows a lack of management skills against the collection and disposition of the blood bags and is in desperate need of proper maintenance formulation.

1.4 Proposed Solution

In order to maintain transparency in the entire system/network, our solution proposes the use of Blockchain. That is, integrating BC in the SCM network where each and every entity can be tracked as per their movement.

- The inclusion of Blockchain solves the disadvantage of ‘distrust’. By providing total transparency to its users, the entities involved in the network that are not be acquainted with one another and may become a potential cause for dissension during the phase of payment transactions. Our system resolves this problem, with the help of Smart Contracts, a feature of BC, that helps take control of the security aspect of these transactions. Only when a certain criterion is satisfied, the Smart Contract is fired, releasing the due payment.
- Secondly, all the data on a blockchain is stored securely through cryptographic hash function and the ledger is owned by each node in the network. This makes it very difficult for any intruder to manipulate the ledger data, ensuring protection against any possible mutations or fraudulent manipulation of the data stored insofar, thus establishing a common base of trust and immutability Also by exposing the supply chain ledger publicly, it helps gain the trust of blood donors.
- Our solution also provides a remedial strategy for improper waste management, which is one of the factors that deter donors away from this system. Using the data uploaded on the BC network, the internal entities can have access to the transparent data and consequently prioritize the process of dispatching the blood bags that are nearing its expiry. The bags which are nearing their expiry are flagged and are placed as ready to be dispatched in a case when an urgent call from a hospital is received.

1.5 Scope of the project

With respect to the donor, our system will take various parameters as input like the amount of blood donated, health conditions of the candidate, and blood group. This input obtained will then get hashed for the block creation process and get updated on the BC network. Simultaneously, the Certificate module will get initialized with the corresponding input values. However, the system does not limit itself to the block generation and updating it on the network. Upon further transportation of the blood bags, required entities will generate invoices and update the details on to the network as well.

- Once the bag has arrived at the T&P, after the testing and processing at this stage the system will output the new UID's of the split bags (if any), and update the expiry value for each and every bag received or generated, on to the Database as well as the BC network.
- After reaching the SCT, all the physical regulatory measures such as expiry check, temperature conditions, preservation and anticoagulation parameters, are updated on to the Database as well as the Network.
- Now as and when the requirement for the blood arises from the hospital, an appropriate urgency request is broadcasted to the nearby SCT and Hospitals, stating the precise values for the requirements. The system then calculates the cost structure and handles the payment of the transaction according to the various urgency-based scenarios. After the business is completed the details are updated on the Database and Network.

CHAPTER 2

Literature Review

We have divided this literature review into two sections wherein first we review the previous work accomplished on the use of BlockChain in general supply chain management and then move on to review the literature on the use of BlockChain in blood SCM.

2.1 Blockchain in SCM

BlockChain is an ideal platform that can solve a number of problems related to centralized SCM's such as provenance tracking, cost reduction and the most important of all, establishing trust. The paper proposed by el Maouchi et al. [2] justifies the application of BlockChain in SCM and its feasibility. Their work is generalized for a standard SCM where a product travels from the initial actor to the final actor through the entire cycle and at each exchange, a transaction is validated through consensus and authentication which is then added to the blockchain. The paper provides standard architecture and algorithms to be executed when two actors exchange the product.

Many other blockchain-based SCM papers include the one proposed by Patrick Sylim et al. [3]. This paper focuses on solving one of the major problems faced by the existing SCM in the medical field i.e. drug counterfeiting, using BlockChain to track the supply chain of medicines using Ethereum and proof-of-stake algorithm for consensus.

In the field of medicine, blockchain contributes to increased safety and reduced costs which is ensured by identifying changes in ownership of drugs between different participants of the chain such as manufacturers, distributors, packers, and end-users.

In another paper, Feng Tian proposed the use of RFID which allows the users to scan the product and validate its supply chain history, along with BlockChain to track the agri-food Supply Chain [4]. Tian supported the idea that a decentralized approach for tracking products could solve the issues in a centralized approach, such as trust, tampering, etc.

2.2 Blood SCM

Research on the blood cold chain system has been conducted by several approaches. Davis et al. proposed an RFID based system which dynamically manages the information related to blood [5]. Quite a few problems faced by the currently existing system for blood information management were pointed out in [5]. For example, the amount of information that can be contained in the bar code is limited, and the information is not reflected in the blood source in real-time, trusting the respective actors on updating legit information on the system, etc.

Jabbarzadeh et al. proposed a robust network design model for the blood supply chain in the event of a disaster [6]. They took into consideration a set of decision parameters such as the number of blood centers nearby etc in order to manage blood allocation, blood collection, and other such critical decisions during a disaster.

Kim et al. proposed a model that modifies the standard models proposed before that minimizes transportation time of blood during emergencies by not only allowing the blood storage centers to distribute blood but also authorizing specific hospitals as blood distribution institutions [7]. The existing blood information management system manages the information just focusing on the situations before it is supplied from blood banks to a number of medical institutions. Despite efforts to track real-time information on the blood boxes or transport vehicles, there is a limitation to how closely each medical institution conducts blood transfusions and disposal. Besides, in the event of an emergency, there is a possibility that placing temporary facilities or designating a new base for blood supply, does not take realistic complex factors into account. They proposed a system wherein an actor in need of blood can view the nearest hospitals or blood centers that have the available blood for distribution thus helping the end-users avoid delays during emergencies.

CHAPTER 3

System Analysis

3.1 Functional Requirements

The System will contain 5 users as follows.

The **donor** will be registered on to the network on donating for the first time and he/she can then login to the application and see/track his donations through a certificate on the web/app UI. The Donor will only be able to view the necessary details about his donated blood on the app.

Blood collection centers (**BCC**) receive donations and for every new blood bag donated it creates a new blood bag (with initial information) and updates it on the BC which will be the genesis block. BCC can make a transaction of blood bags to testing and processing center (T&P).

Testing and processing centers (**T&P**) accept transactions from BCC and adds necessary information about tests performed on the blood and then make a transaction to the Storage center (SCT).

Storage center (**SCT**) receives bags from T&P thus claiming the ownership of the bag for storing it under proper conditions until a request has been placed for the supply of blood by any hospital. On receiving a request to send blood, SCT then performs a transaction to hand over the ownership of the blood bag to the hospital and on a successful transaction, SCT gets paid for the same through Smart contracts (SCs).

Hospitals receive bags from SC and/or other hospitals that also act as suppliers on having excess blood at their disposal, on putting up a request for it. Hospitals can also get requests for blood from nearby hospitals. Finally, hospitals update the information of blood bags depending on whether they were used for any patients or are stored for future use, etc.

3.2 Non-Functional Requirements

Below are discussed some non-functional attributes of blockchain.

- Openness: The nodes in the blockchain show interoperability that means that the nodes in the Blockchain have the ability to exchange and use information during a transaction.
- Concurrency: Nodes process concurrently to enhance the performance of the blockchain.
- Scalability: Nodes can be added or removed to make the Blockchain flexible. Scalability considers three things:–Size: Mores nodes can be added easily in the Blockchain network.–Distribution/Transaction Processing rate: Geographical dispersion of the nodes does not degrade the performance of the Blockchain.–Manageability/latency: The Blockchain should be manageable as no. of nodes in the Blockchain increases and the different nodes are located in different parts of the world.
- Fault tolerance: The transactions in the Blockchain are immutable and any fault at any node will be transparent to all other nodes in the Blockchain.
- Transparency: Every transaction in the Blockchain is visible to each node in the blockchain network.
- Security: The Blockchain uses strong Cryptographic protocols such as SHA-256for securing the data in the Blockchain.
- Quality of Service: The quality of service (QoS) determines the reliability and response time and throughput that is the system's ability to deliver its services dependably and with response time and throughput that is satisfactory to the nodes in the Blockchain. Quality of Service is a critical parameter for critical data transactions among the nodes.
- Failure Management: The Blockchain shall be designed such that it is resilient to the failures. There shall be mechanisms that can discover the cause of the failure and suggest recovery schemes so that the Blockchain can automatically recover from the failures.

Non-functional requirements for the Users

- A web supporting device with an internet connection to use the application is preferred so that the devices can be used by the blockchain for consensus protocol and to ensure smooth working of the system.

3.3. Specific Requirements

1. Software Requirements -
    - An application that encapsulates the BC and the entire system for users to use for making transactions and tracking their blood bags.
    - Ethereum - Blockchain
    - Ganache - a personal blockchain for Ethereum development
    - Truffle - a development environment for Ethereum decentralized applications

2. Hardware Requirements -
    - Laptop / Computer
    - RAM: 8GB
    - GPU: 1080 Ti 8GB
    - Hard Disk: 256 GB

3.4 Use-Case Diagrams and Description

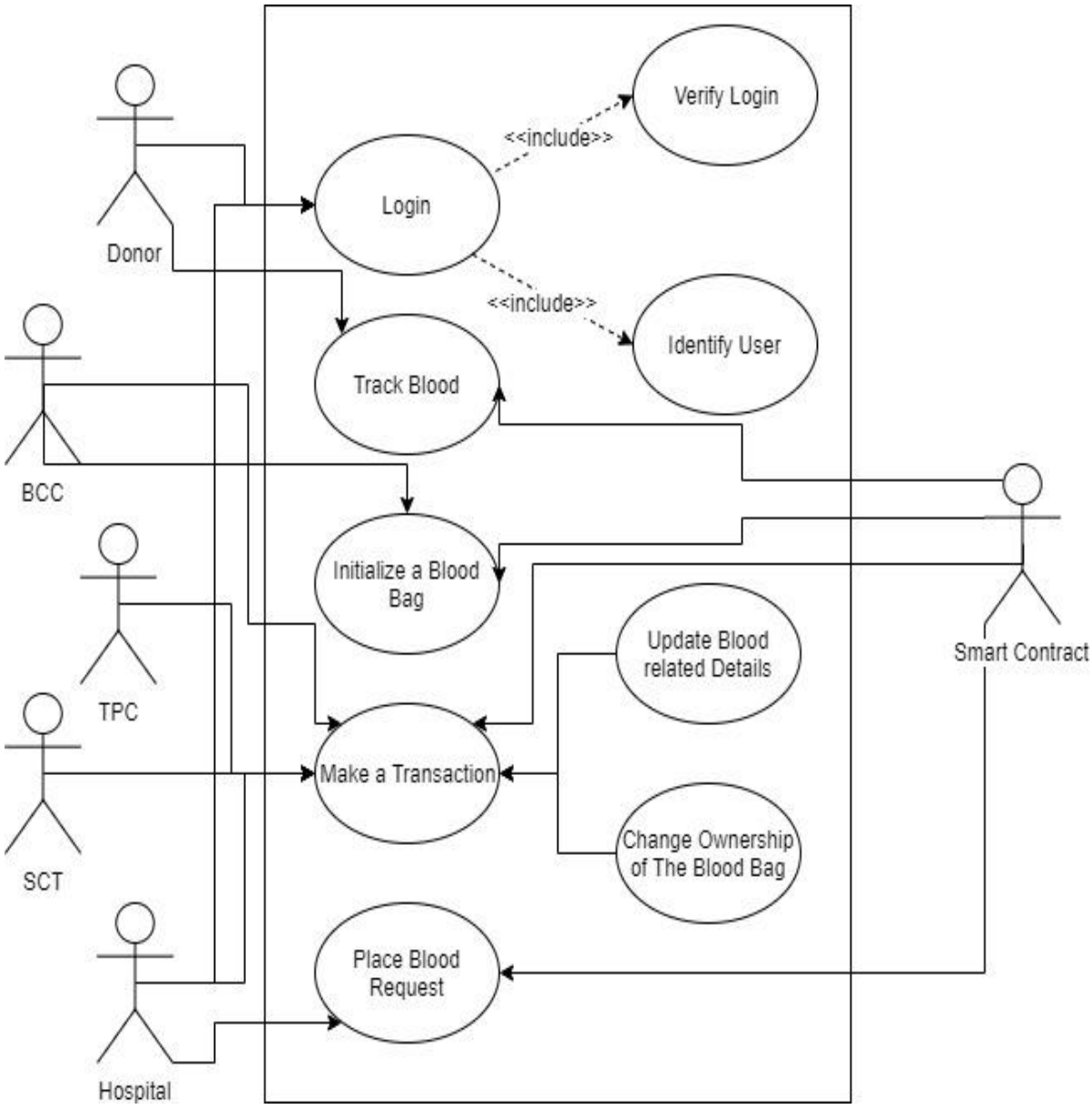

Fig 3.4.1 Use Case Diagram

CHAPTER 4

Analysis Modelling

4.1 Activity Diagrams

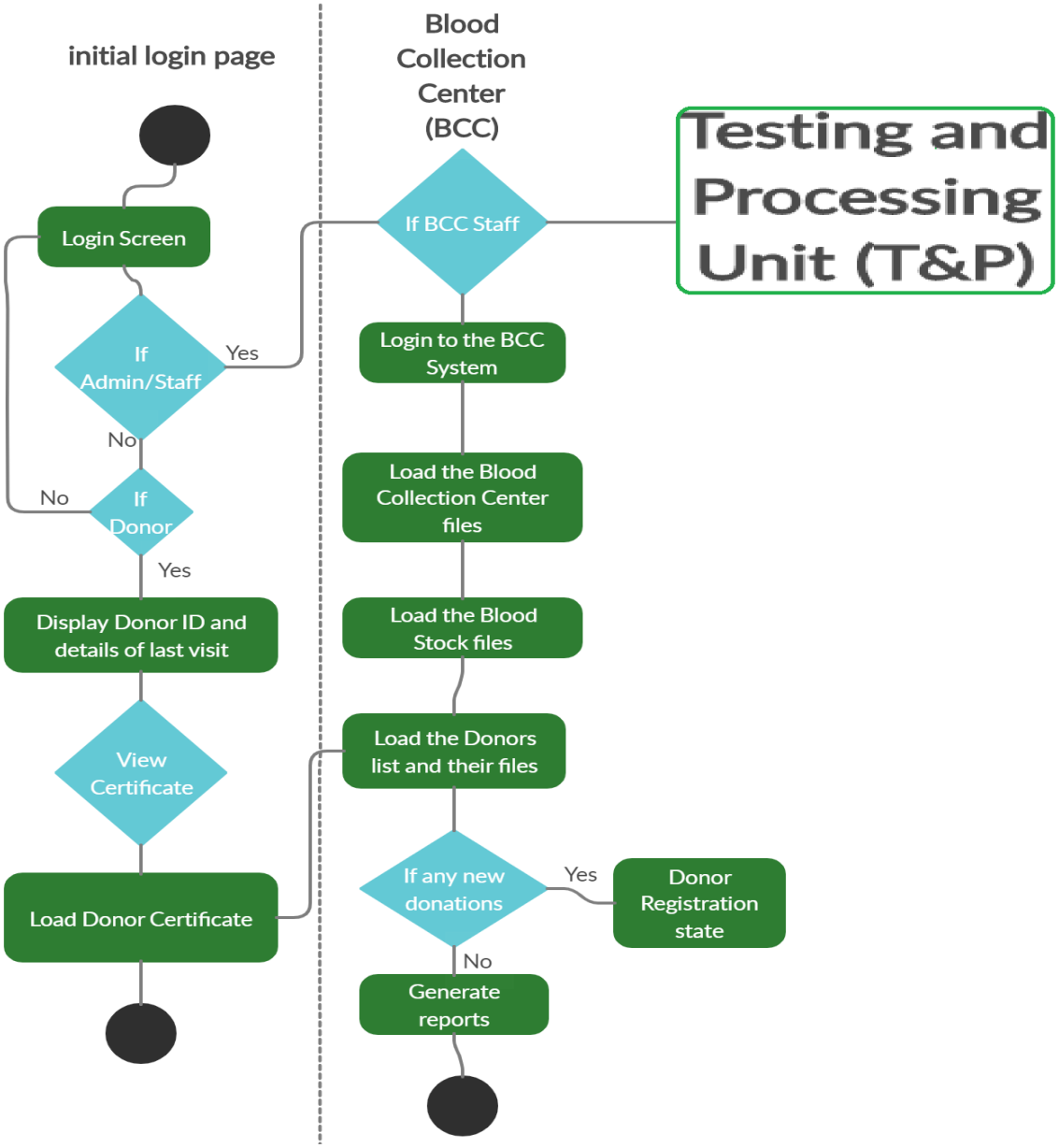

Fig 4.1.1 Activity Diagram for Donor and BCC

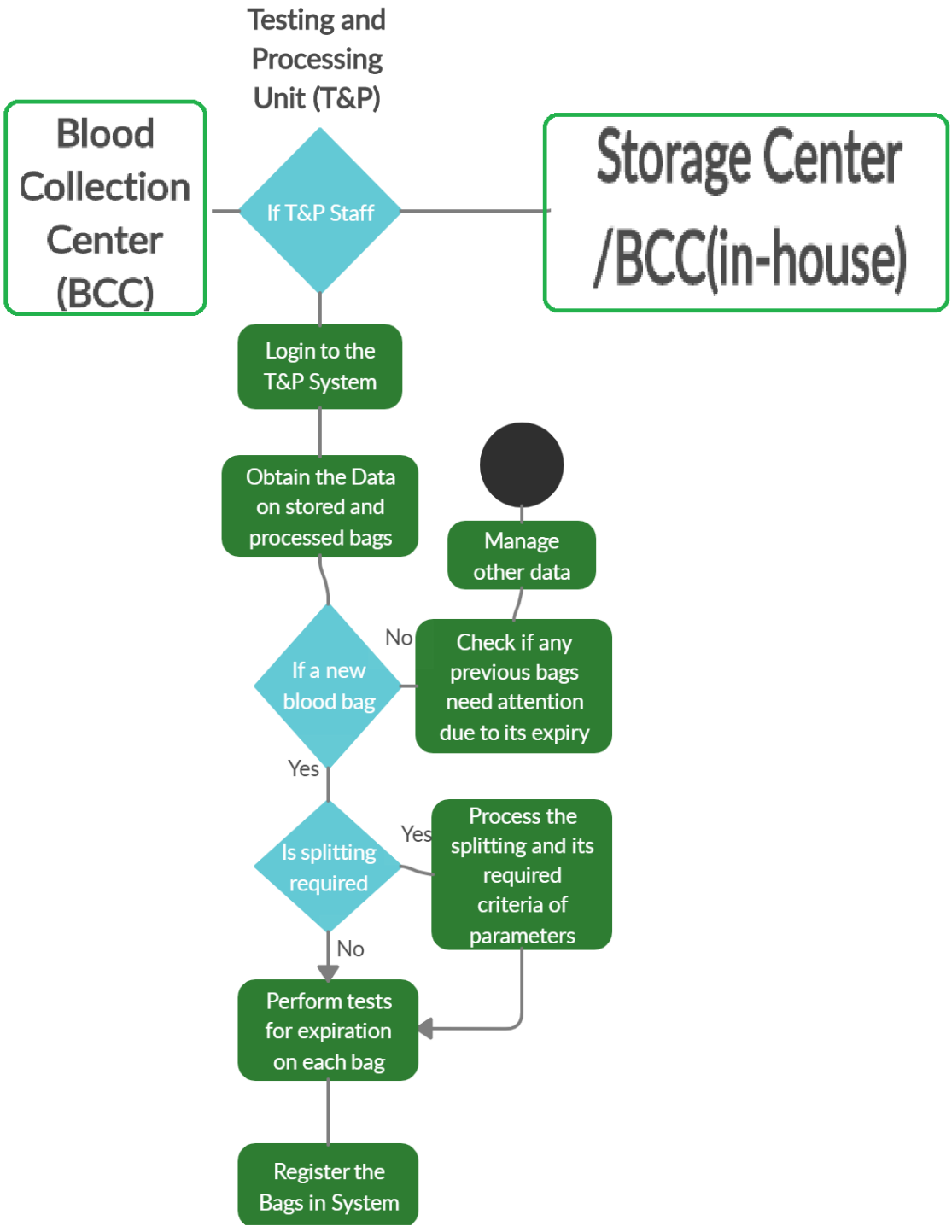

Fig 4.1.2 Activity Diagram for Testing and Processing Center (T&P)

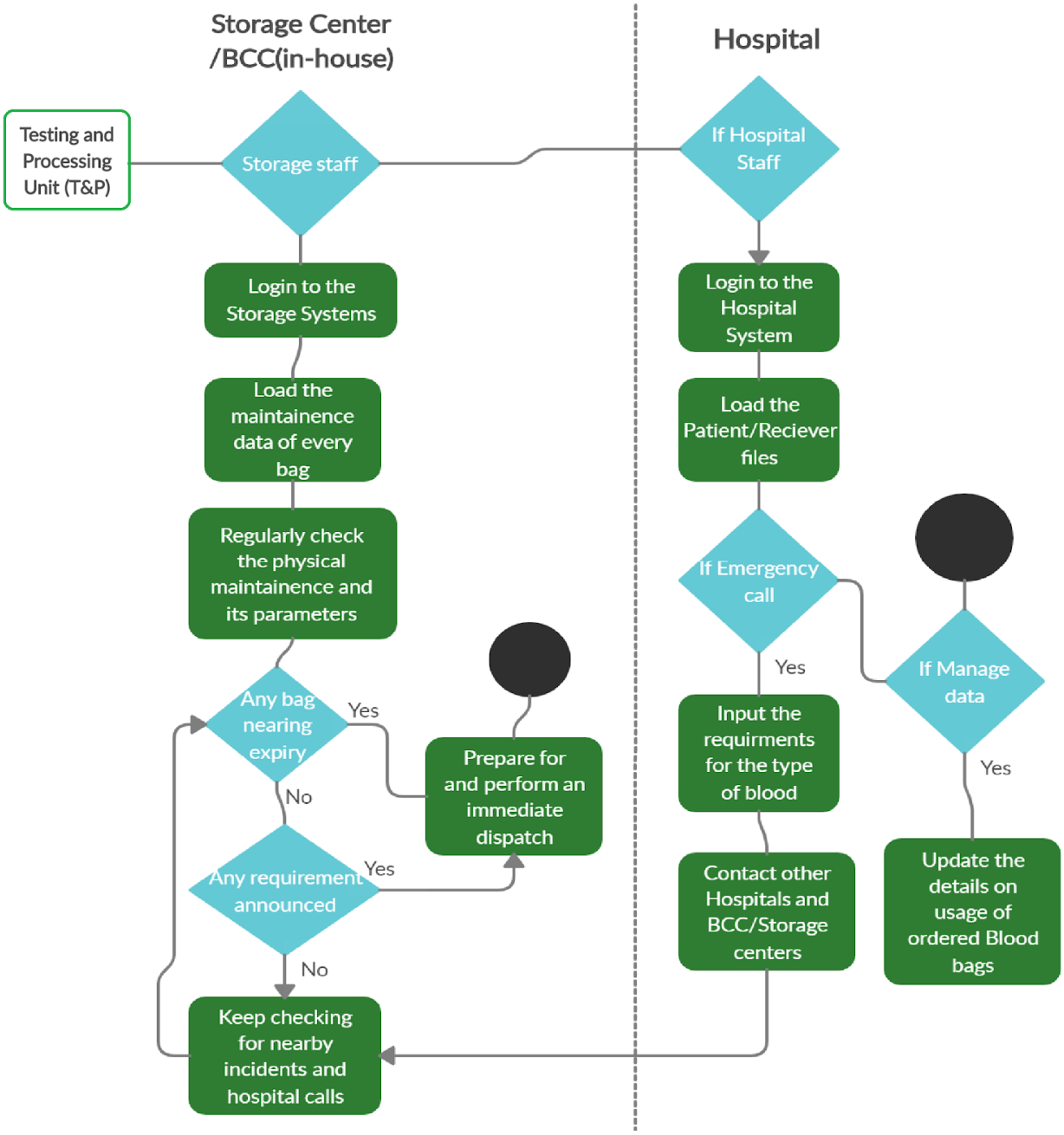

Fig 4.1.3 Activity Diagram for Storage Center and Hospital

4.2 Functional Modeling (DFD level 1)

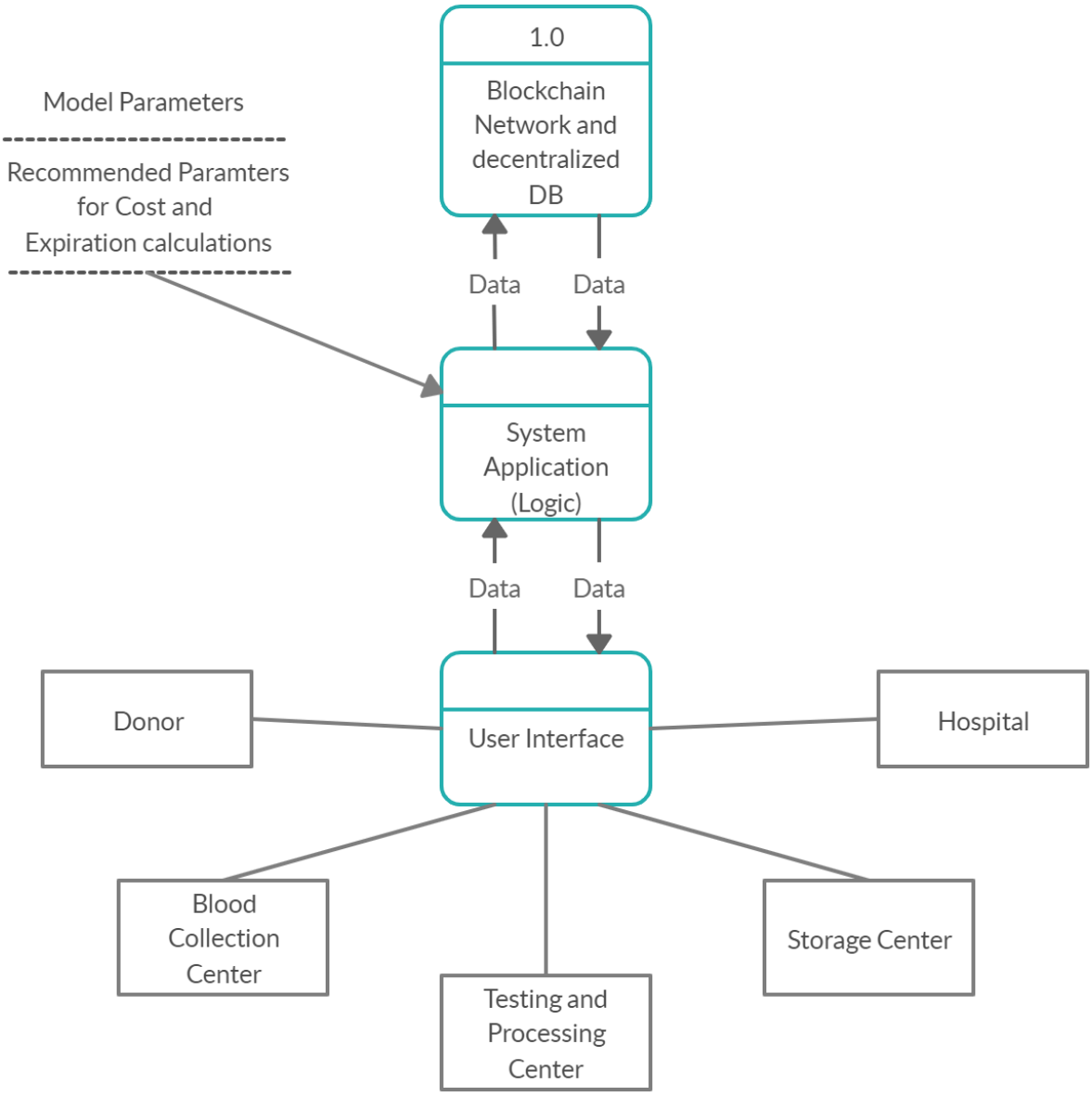

Fig 4.2.1 DFD level 1

4.3 TimeLine Chart

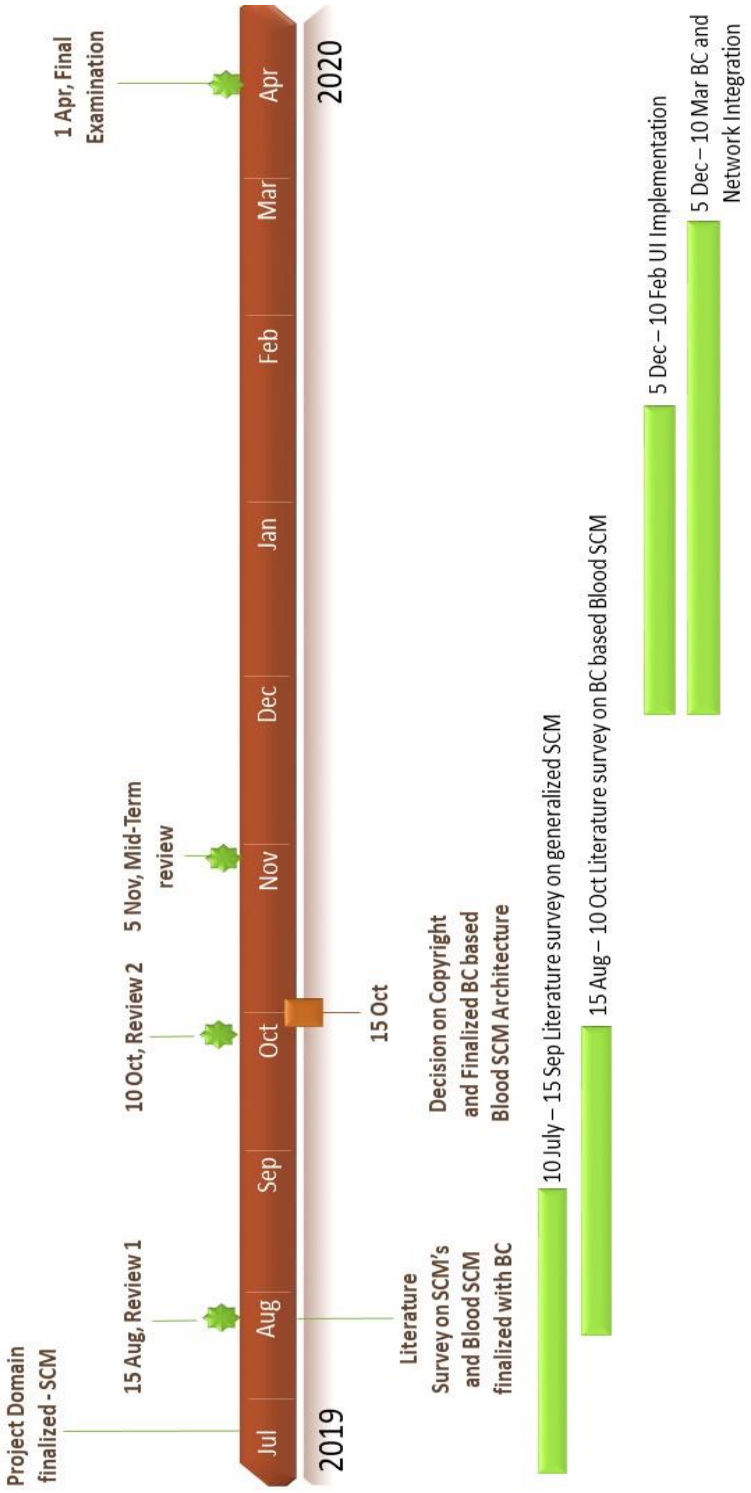

Fig 4.3.1 Timeline Chart

CHAPTER 5

Design

5.1 Architectural Design

Blood for transfusion needs to be accompanied by a quality assurance from the blood donors to the recipients. Therefore, end-to-end visibility of the blood cold chain is required. The figure below shows the system with the proposed shared ledger technologies.

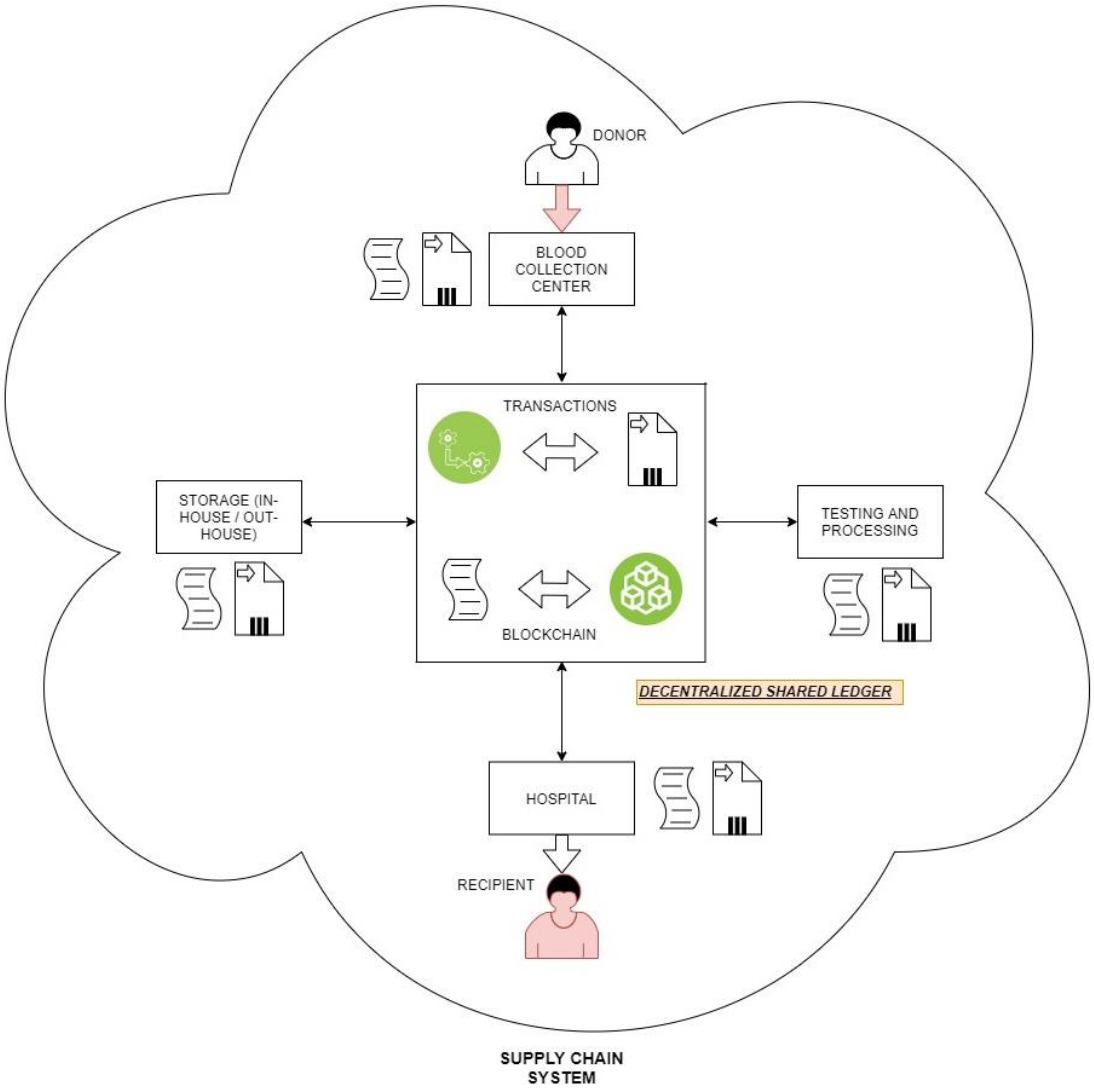

Fig 5.1.1 Proposed Architecture

5.2 Certificate

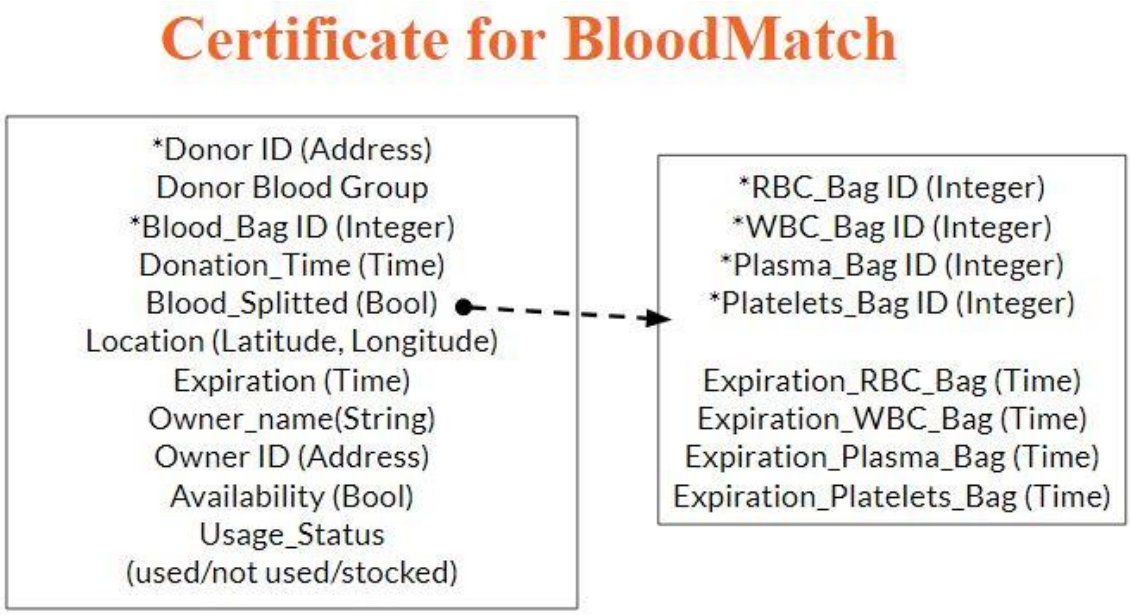

Fig 5.2.1 Certificate with required parameters

5.3 Block Transactions

In the distributed ledger structure, all transactions are recorded and shared within the blood cold chain network. The BCC block collects information about the blood and the transaction is recorded. In T&P Block, there is a transaction in which the blood that has been judged as conforming moves to the Storage unit. The Storage Block checks parameters for proper storage and indicates that transactions have occurred in which the blood is transferred to the hospital upon request of the hospital. In Hospital Block, the ownership status will be updated. The Boolean parameters viz. Usage Status and Availability will be updated after the requirement of Blood Bag has been satisfied. While processing the smart contract like this, the invoked transactions are recorded like logs.

The Location and Ownership Status will be updated as and when the Blood Bag reaches a particular user (module) of the system.

Since the quick supply of blood in golden time is an important issue, we propose a system to supply blood directly between hospitals. This is when blood is not supplied from the blood banks to the hospitals within the golden time, and the surplus blood is requested from the nearby hospitals.

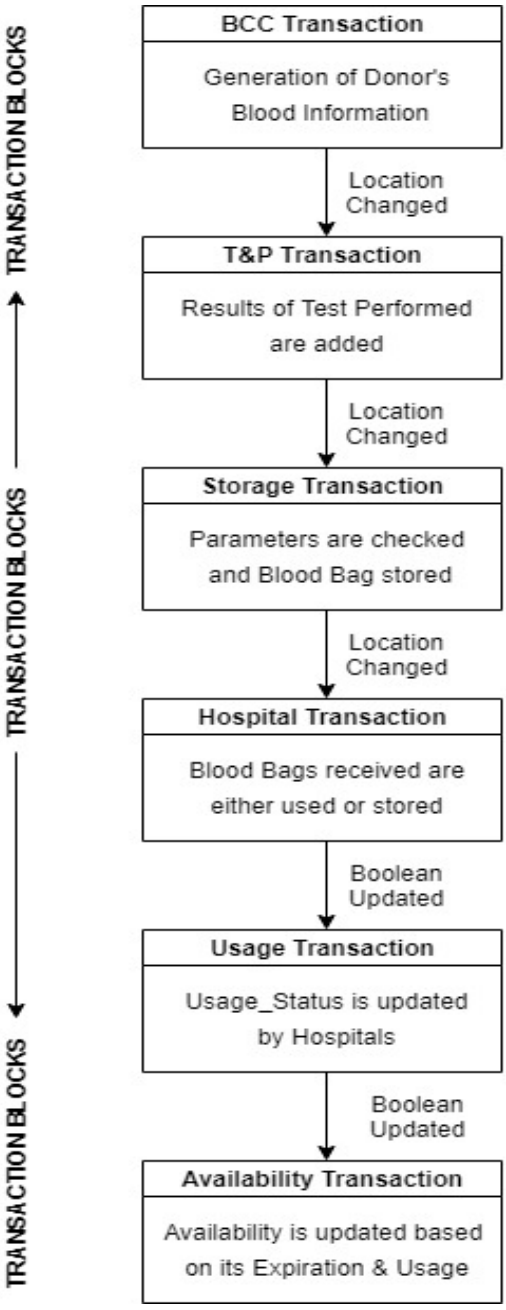

Fig 5.3.1 Block by Block Transactions

5.4 User – Interface (UI) Design

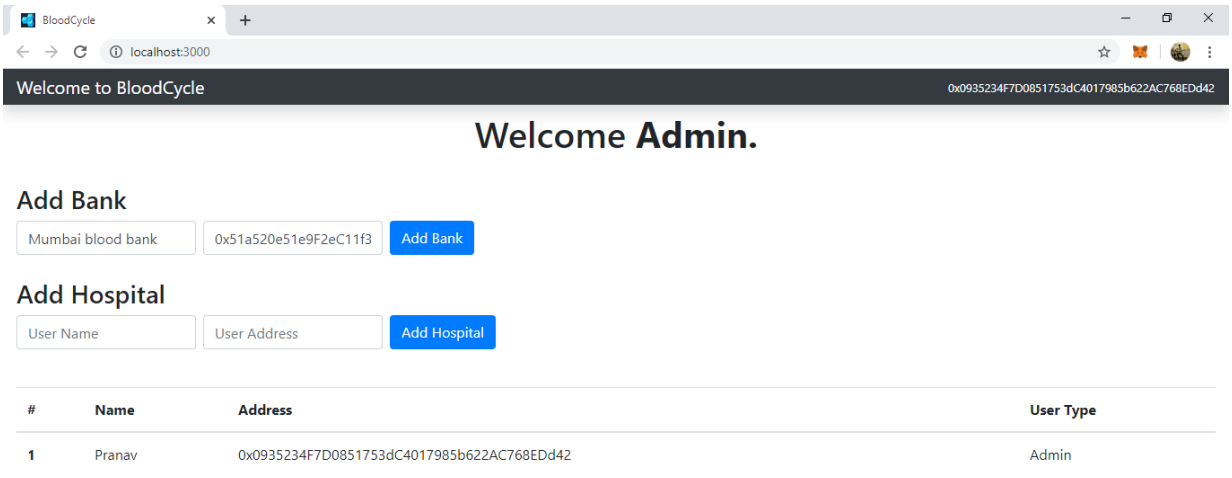

Fig 5.4.1 The Admin Interface

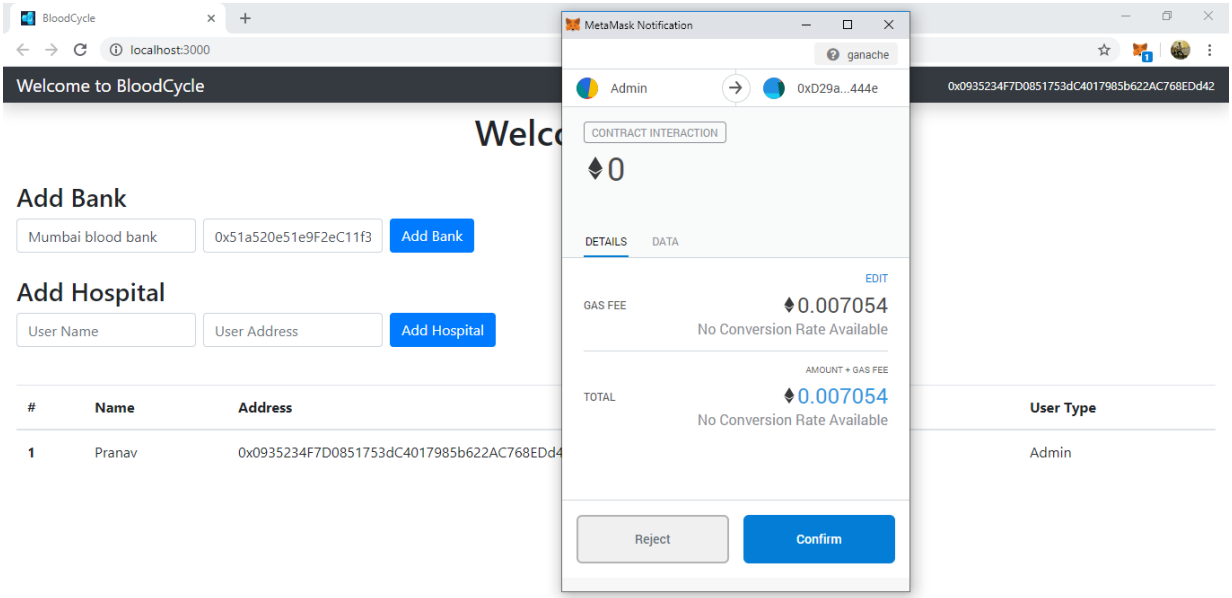

Fig 5.4.2 The Admin can add Bank or Hospital

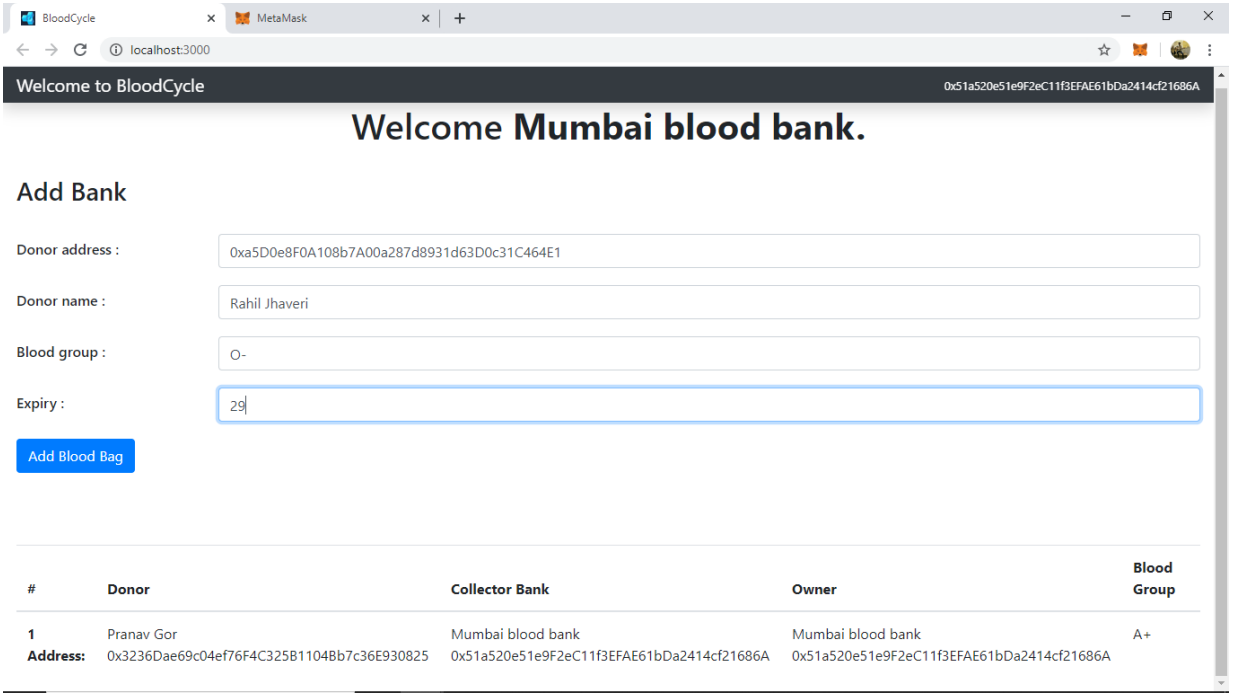

Fig 5.4.3 Bank can submit donor details

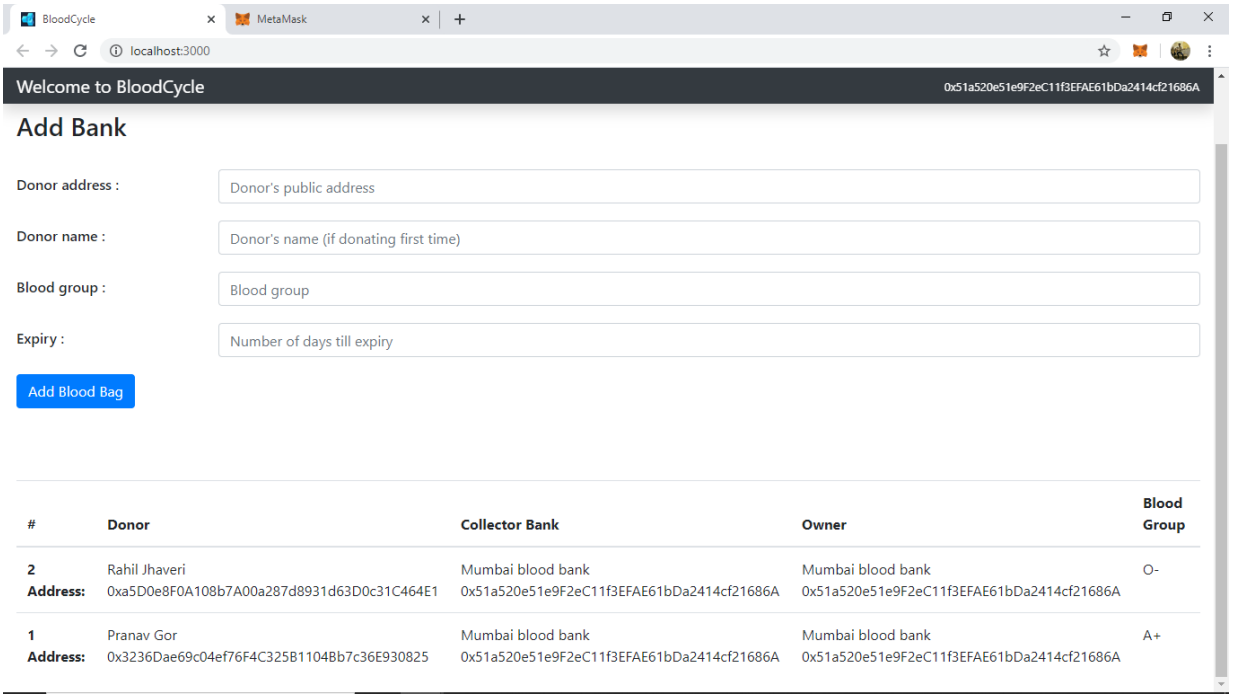

Fig 5.4.4 Bank will have access to blood bags collected by it

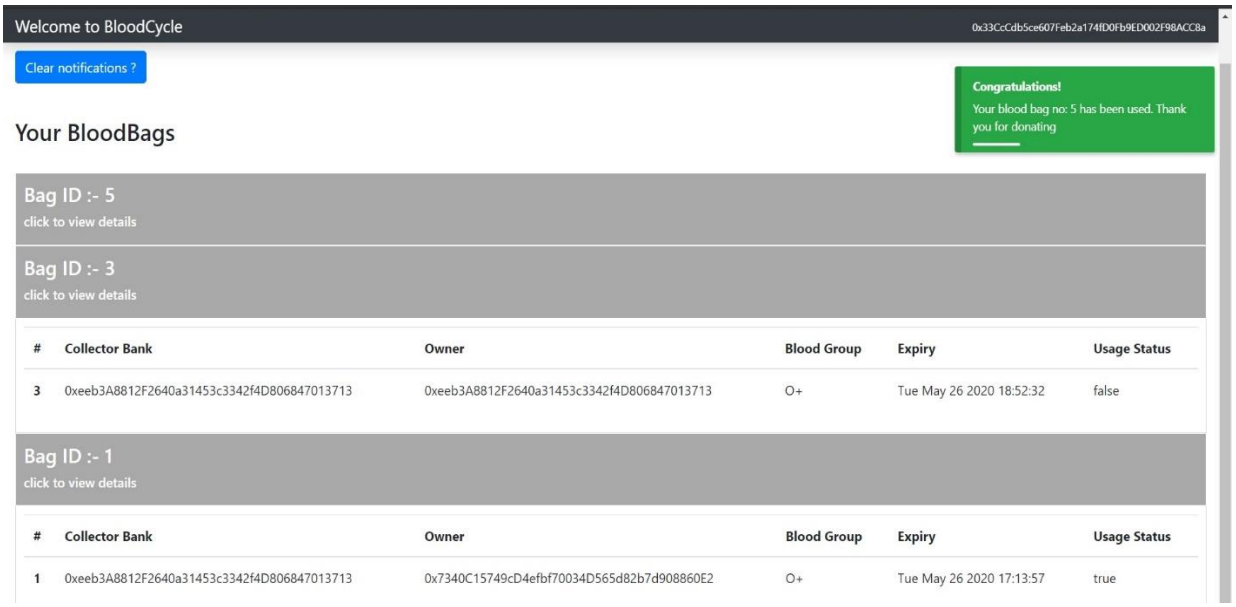

Fig 5.4.5 Donor will be able to track all his/her donated blood bags.

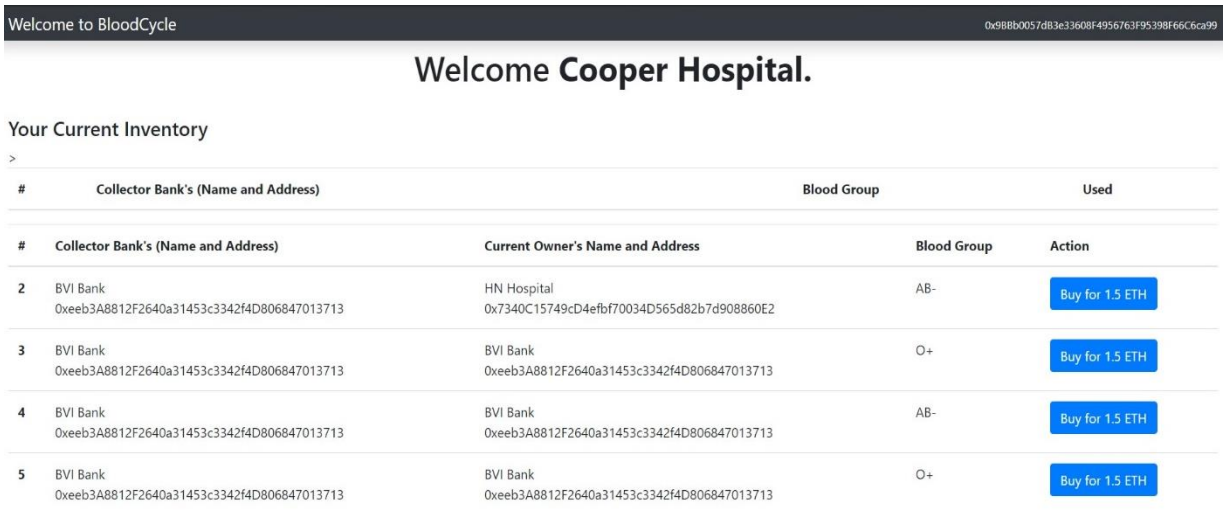

Fig 5.4.6 Hospitals can view available blood bags with its ownership details

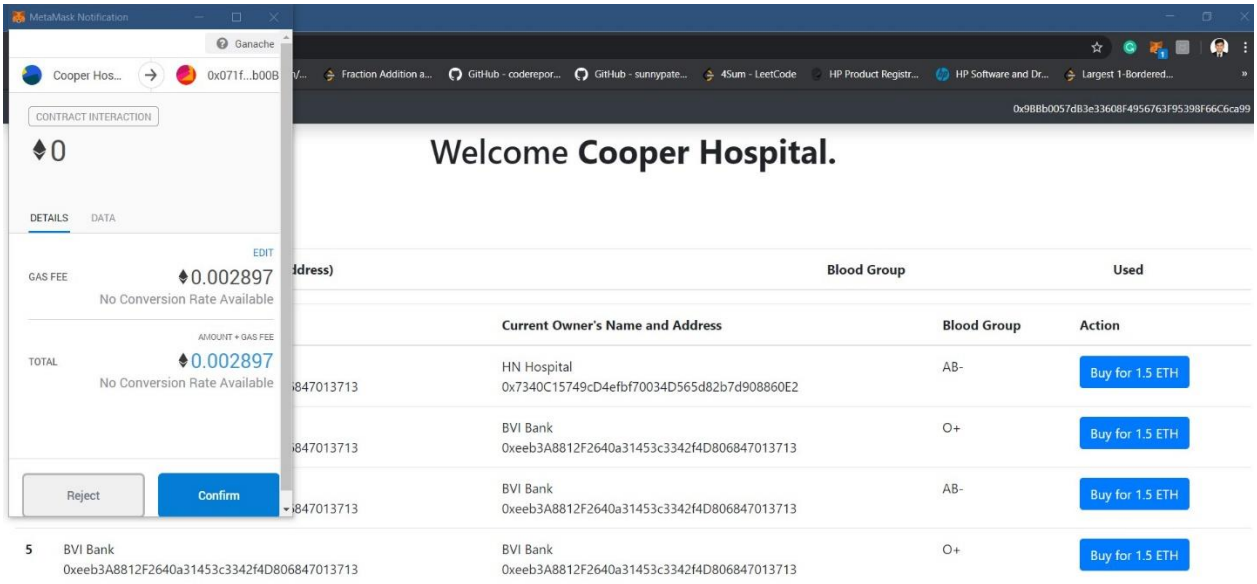

Fig 5.4.7 Hospital can buy blood bags by paying amount in ETH

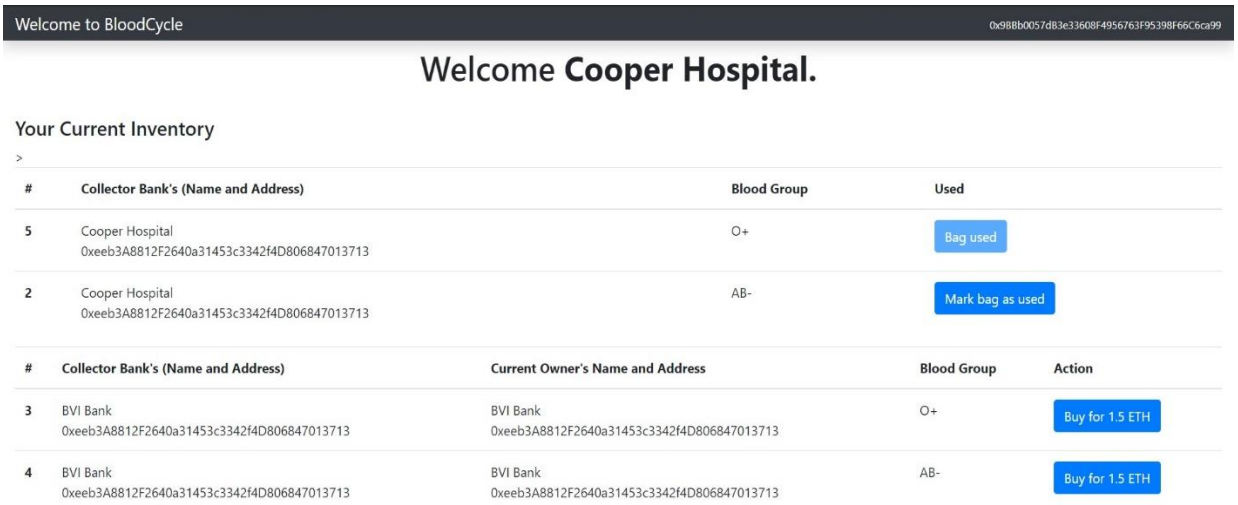

Fig 5.4.8 The blood bags bought will be added in hospital's inventory

CHAPTER 6

Implementation

6.1 Algorithms / Methods Used

Our algorithm, as previously stated is heavily dependent on the specific features of solidity, such as the 'Smart Contracts'. Shown below are the snippets of our main contract – '*BloodManagement.sol*'.

```
mapping(uint => Bloodbag) public bloodbags;
mapping(address => uint[]) public donors;
mapping(address => uint[]) public hospitals;
mapping(address => User) public usertype;
mapping(address => uint[]) public notification;
mapping(uint => User) public users;
```

Fig 6.1.1 Mappings in Contract

Mappings: Mappings can be seen as hash tables which are virtually initialized such that every possible key exists and is mapped to a value. Here we have 5 prime mappings:

- **Bloodbags** – This is a very general list of bloodbags. It stores and returns the entire structure(object) of a Bloodbag, whose '*id*' will be equal to the uint (integer) provided to the mapping.

- **Donors** – The Donors mapping will store and return an array, consisting of the bloodbag id(s) corresponding to each and every donor for a particular Blood Collection Center. Based on the address of the Bank provided, the list of donors for that center will be retrieved.

- **Hospitals –** This mapping stores and returns an array, consisting of the bloodbag id(s) that are currently in possession and owned by the given address's hospital.

- **Usertype –** This is the type of mapping used for administrative purposes. It is used specifically to provide the client's information, such as the type of user. For example –
    1. '1' is for Donor
    2. '2' is for Blood Bank
    3. '3' is for Hospitals

This '*usertype*' mapping helps the system to dynamically recognize the type of client, who is visiting the webpage. Based on the usertype and the address of the client, we can provide a fluent access to the data belonging to the user/client.

- **Notification –** This mapping stores and returns an array, consisting of the bloodbag id(s) that have been used by the hospital(s). It is then used to notify respective donors about their donated blood. It is initially set to *false* by Blood Bank admin and updated to *true* by Hospital Admin.

- **Users –** Lastly, the '*users*' mapping is used to contain the list of entire user network, including the donors, hospitals, and the banks. It helps to keep the access of *id* of every user in a simplistic yet efficient manner.

```
struct User {
    uint id;
    uint user_type;
    string user;
    address payable user_address;
    string name;
}

struct Bloodbag {
    uint id;
    bool used;
    bool first;
    uint donation_date;
    address payable donor;
    address payable bank;
    string blood_group;
    uint expiry;
    string owner_name;
    address payable owner;
    // uint price;
}
```

Fig 6.1.2 Structs in Contract.

Structs: Every system has to use a type of datastructure that is efficient enough to handle the application of the system and suffice with its use-case. Our system comprises of two different '*structs*'.

- **User** – The user struct is a skeleton holding certain parameters for each and every user that is registered on our network. These value are –
    1. **Id -** Used for listing and identification purposes.
    2. **user_type –** It helps denote the category of the user (Bank/Hospital/Donor).
    3. **user –** Contains the description of the given user.
    4. **user's network address –** The blockchain network enabled address of the user.
    5. **unique name –** For listing and Identification Purposes.

- **Bloodbag** – The bloodbag struct holds the information unique to each and every bag created/donoted to the blood bank, that enters into our network.
    1. **Id -** Used for listing and identification purposes
    2. **Donation date –** It is factored in for the authenticity of the bloodbag created. It helps in calculating the expiry date of the blood bag.
    3. **Used –** A boolean field to check if bag is used or not.
    4. **Address (Donor) –** Used to maintain the link to donor, inorder for the donor certificate's tracking.
    5. **Address(bank) –** Used for maintaining the relation to the bank that has generated the bag.
    6. **Blood Group –** Used for Medical purposes.
    7. **Expiry –** Used for secure medical Purposes.
    8. **Owner name –** Used to track the current dynamic ownership/holder of the bag in the system.
    9. **Address Owner –** Used to track the purchase details for the owner where the payment is transacted.

6.2 Working of the Project

```
function createBank(address payable _bank, string memory _name) public {
    require(msg.sender == admin,"Not an admin");
    require(_bank != address(0),"No bank address");
    require(usertype[_bank].user_type != 2,"Bank already exists");
    userCount++;
    usertype[_bank] = User(userCount, 2, "Bank",_bank, _name);
    users[userCount] = User(userCount, 2, "Bank",_bank, _name);
}

function createHosp(address payable _hosp, string memory _name) public {
    require(msg.sender == admin,"Not an admin");
    require(_hosp != address(0),"No hosp address");
    require(usertype[_hosp].user_type != 3,"Hospital Already exists");
    userCount++;
    usertype[_hosp] = User(userCount, 3, "Hospital",_hosp, _name);
    users[userCount] = User(userCount, 3, "Hospital",_hosp, _name);
}

function h_placeOrder(uint bag_id) public payable {
    // Fetch the owner
    address payable _seller = bloodbags[bag_id].owner;
    bool used = bloodbags[bag_id].used;
    require(msg.value > 1, 'Wrong price paid');
    require(usertype[msg.sender].user_type == 3, 'Unauthorized transaction originator');
    address(_seller).transfer(msg.value);
    //Transfer ownership
    bloodbags[bag_id].owner = msg.sender;
    bloodbags[bag_id].owner_name = usertype[msg.sender].name;
    Bloodbag memory _bloodbag = bloodbags[bag_id];
    hospitals[msg.sender].push(_bloodbag.id);
    emit bagPurchased(_bloodbag.id, used, _bloodbag.donation_date, _bloodbag.donor, _bloodbag.
     _bloodbag.blood_group,_bloodbag.expiry,_bloodbag.owner_name,_bloodbag.owner);
}

function useBag(uint bag_id) public {
    require(usertype[msg.sender].user_type == 3, 'Unauthorized transaction originator');
    bloodbags[bag_id].used = true;
    address donor = bloodbags[bag_id].donor;
    notification[donor].push(bag_id);
}
```

Fig 6.2.1 Functions in Contract – I

```
function createBloodbag(uint _donation_date, address payable _donor,
string memory _donor_name,string memory _blood_group, uint _expiry) public {
    // Require valid params
    string memory d_name = _donor_name;
    require(usertype[msg.sender].user_type == 2, "Not a blood bank.");
    if(bytes(d_name).length == 0){
        d_name = usertype[_donor].name;
    }
    require(bytes(d_name).length > 0, "Please input name");
    require(bytes(_blood_group).length > 0, "error in blood group");
    require(_donation_date > 0, "error in d_date");
    require(_expiry > _donation_date, "error in exp_date");
    require(_donor != address(0), "error in donor");
    // Increment bag count
    bagCount ++;
    // Date related stuff
    // Add donor and set type = 1 if not already exists.
    if (usertype[_donor].id == 0){
        userCount++;
        usertype[_donor] = User(userCount, 1, "Donor", _donor, d_name);
        users[userCount] = User(userCount, 1, "Donor",_donor, d_name);
    }
    // Create the Blood bag
    string memory _owner_name = usertype[msg.sender].name;
    Bloodbag memory temp_bloodbag = Bloodbag(bagCount, false, false, _donation_date,
        _donor, msg.sender, _blood_group, _expiry,_owner_name, msg.sender);
    bloodbags[bagCount] = temp_bloodbag;
    // add bag to specific donor's list
    donors[_donor].push(temp_bloodbag.id);
    // Trigger an event
    emit BagCreated(bagCount, false, _donation_date, _donor, msg.sender,
    _blood_group, _expiry, _owner_name, msg.sender);
}
```

Fig 6.2.2 Functions in Contract – II

The functions used by the network are:

- **Admin Functions:** These are the functions designed for the admin panel to carry out the backend tasks. Namely:
    1. **createBank** – The admin will assign the registration 'creation' of every blood bank.
    2. **createHosp** – The admin will handle the registrations for every hospital that get affiliated with our network.

- **System Functions:** The system will perform certain desired operations for the registered entities. For example –
    1. **createBloodbag** – The banks on receiving the donations made by the donor, will log in to their systems and register the details on the blood bag.
    2. **h_placeOrder** – The hospitals, when in need of new blood bags, will order for the purchase of bloodbags from the nearby Banks and other Hospitals. This transaction will be validated and performed securely through this function.
    3. **useBag** – Whenever the hospital uses a blood bag from its inventory, the donor of that blood bag will be notified. The bag ID will be pushed at the back of the notification array corresponding to the donor ID of same bag.

CHAPTER 7

Testing

7.1 Test - Cases

Testing is performed on all the functions mentioned in smart contract in-order to check the proper working of the system in all scenarios. Various data structures and variables are tested too. Following is the code for testing the smart contract.

Blood.test.js

test > Blood.test.js > ...

```
1  const Blood = artifacts.require("./Blood.sol");
2
3  require('chai')
4    .use(require('chai-as-promised'))
5    .should()
6
7  contract('Blood', ([donor, bank, hosp, admin, bank2, hosp2]) => {
8    let blood
9
10   before(async () => {
11     blood = await Blood.deployed()
12   })
13
14   describe('deployment', async() => {
15     it('deployed successfully', async() =>{
16       const address = await blood.address
17       assert.notEqual(address, 0x0)
18       assert.notEqual(address, '')
19       assert.notEqual(address, null)
20       assert.notEqual(address, undefined)
21     })
22   })
23
24   describe('Bloodbags', async () => {
25     let result, bagCount, donation, expiry, event
26
27     before(async () => {
28       const a = await blood.usertype(admin).user_type
29       console.log(await blood.usertype(admin))
30       // console.log(a.toNumber())
31       await blood.createBank(bank, "Lilavati blood bank", {from: admin}) // I
32       await blood.createBank(bank2, "Jamnavati blood bank", {from: admin})
33       await blood.createHosp(hosp, "Bachao mereko hospital", {from: admin})
34       await blood.createHosp(hosp2, "Ye Marega hospital", {from: admin})
```

Fig 7.1.1 Test-Cases – I

Blood.test.js

test > Blood.test.js > contract('Blood') callback > describe('Bloodbags') callback > it('creates bags') callback

```
46
47      it('creates bags', async () => {
48        // SUCCESS
49        assert.equal(bagCount, 5)
50        const event = result.logs[0].args
51        assert.equal(event.id.toNumber(), bagCount.toNumber(), 'id is correct')
52        assert.equal(event.donation_date.toNumber(), donation, 'donation_date is correct')
53        assert.equal(event.donor, donor, 'donor is correct')
54        assert.equal(event.bank, bank, 'bank is correct')
55        assert.equal(event.blood_group, "A+", 'blood group is correct')
56        assert.equal(event.expiry.toNumber(), expiry, 'expiry is correct')
57        assert.equal(event.owner, bank, 'owner is correct')
58
59        // FAILURE: Product must have a name
60        await blood.createBloodbag(donation, "0x0", "A+", expiry,
61        "Lilavati blood bank", { from: bank }).should.be.rejected;
62        // FAILURE: Product must have a price
63        await blood.createBloodbag(donation, donor, "", expiry,
64        "Lilavati blood bank", { from: bank }).should.be.rejected;
65      })
66
67      it('get donors bags', async () => {
68        const arr = await blood.getDbags(donor);
69        const len = arr.length;
70        var i;
71
72        for(i=0; i<len; i++) {
73          const ind = arr[i];
74          event = await blood.bloodbags(ind);
75          assert.equal(event.donor, donor, 'donor is correct')
76          console.log(event.id);
77        }
78      })
79    })
```

Fig 7.1.2 Test-Cases – II

```
81    describe('Hospital', async() =>{
82      let event, donation, expiry, result
83
84      before(async () => {
85        await blood.h_placeOrder(3, { from: hosp, value: web3.utils.toWei('1', 'Ether')})
86      })
87
88      it('shows inventory', async() => {
89        console.log(await blood.getHbags(hosp))
90      })
91
92    })
93
94  })
95
```

Fig 7.1.3 Test-Cases – III

7.2 Type of Testing

Unit testing is used in **Truffle** (an Ethereum development framework) using plugins *Chai* and *Mocha* to test various transactions taking place in the smart contract. Each test function is executed as a single transaction, in order of appearance in the test file (as shown in the previous section). Assertion functions provided by truffle/Assert. ... This is to make writing solidity unit tests easier, and should allow for more extensibility in the future with less hassle.

We decided to go with the Truffle framework as it comes standard with an automated testing framework to make testing the contracts a breeze. This framework lets us write simple and manageable tests in two different ways:

- In JavaScript and TypeScript, for exercising your contracts from the outside world, just like your application.
- In Solidity, for exercising your contracts in advanced, bare-to-the-metal scenarios.

The transactions and blocks can be viewed on Ganache while conducting the tests. It is included in the truffle suite to develop applications and run tests.

Given below is the output of the testing code shown in the previous section

Fig 7.2.1 Testing in Truffle – I

Command Prompt

```
> Artifacts written to C:\Users\admin\AppData\Local\Temp\test-2020330-22844-mcc1nw.ns6gb
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



  Contract: Blood
    deployment
      √ deployed successfully
    Bloodbags
Result {
  '0': <BN: 1>,
  '1': <BN: e>,
  '2': 'Admin',
  '3': '0x0935234F7D0851753dC4017985b622AC768EDd42',
  '4': 'Pranav',
  id: <BN: 1>,
  user_type: <BN: e>,
  user: 'Admin',
  user_address: '0x0935234F7D0851753dC4017985b622AC768EDd42',
  name: 'Pranav' }
      √ creates bags
<BN: 1>
<BN: 2>
<BN: 3>
<BN: 4>
<BN: 5>
      √ get donors bags (1531ms)
    Hospital
[ <BN: 3> ]
      √ shows inventory (77ms)


  4 passing (11s)
```

Fig 7.2.2 Testing in Truffle – II

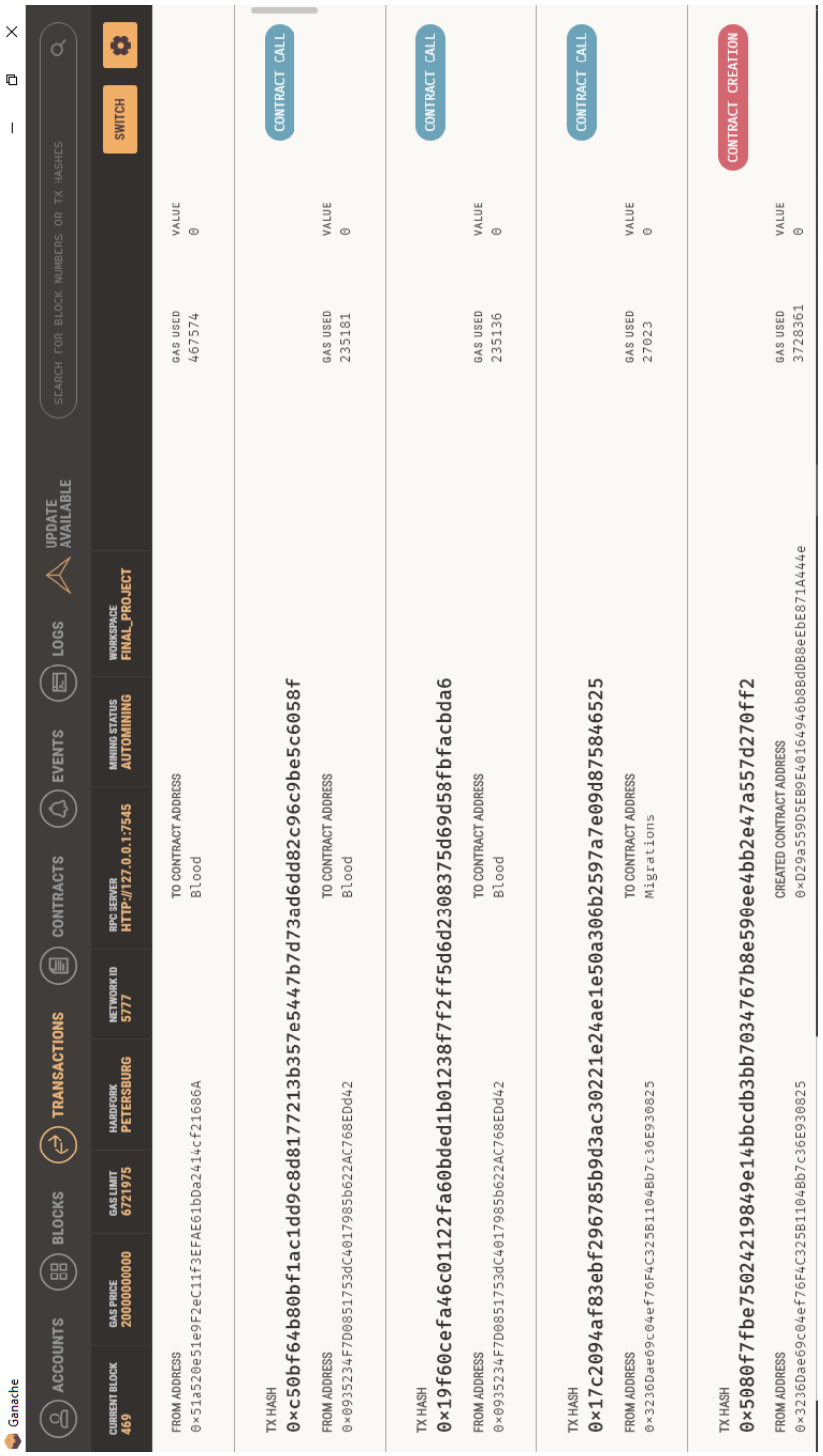

Fig 7.2.3 Test Transactions on Ganache

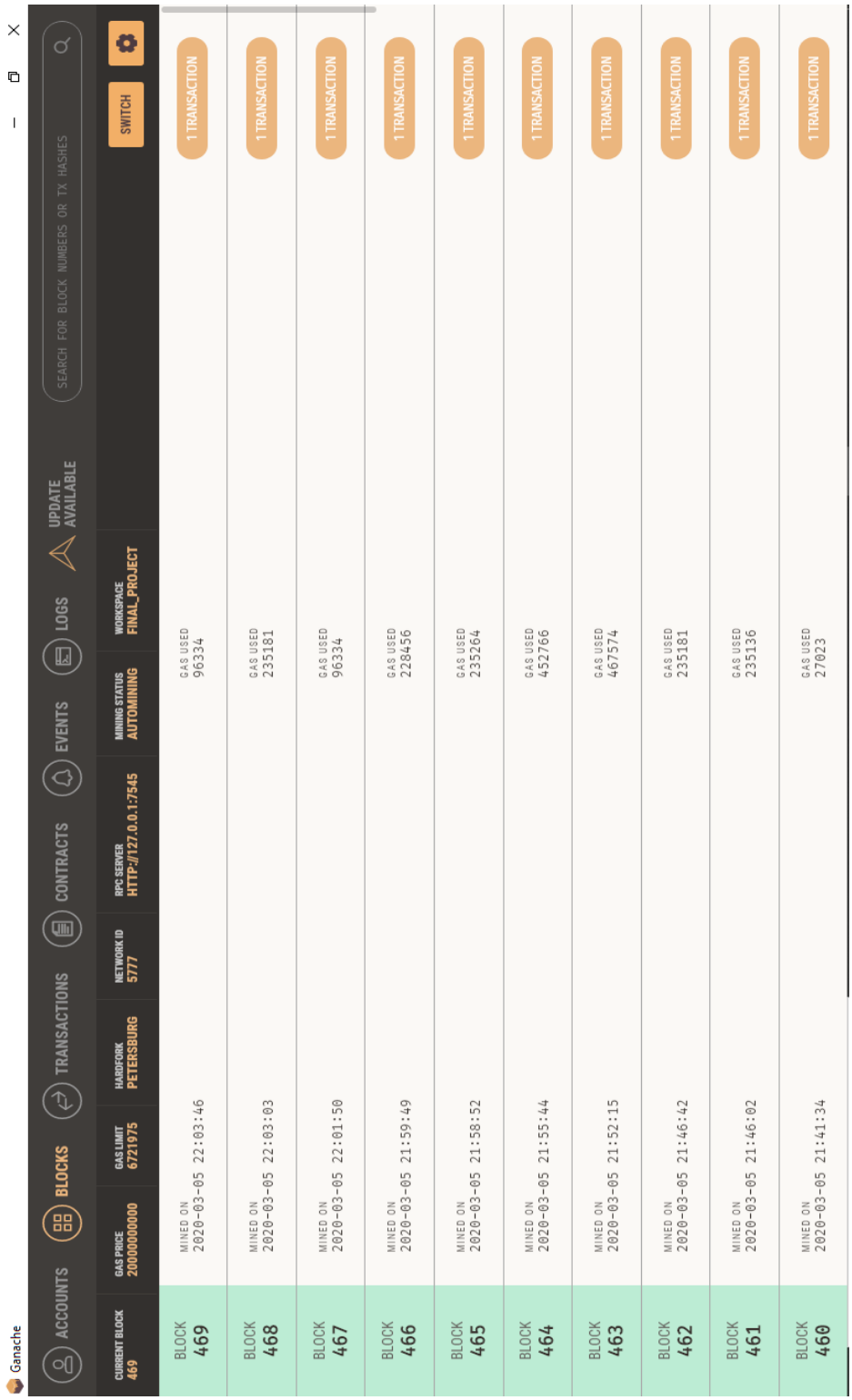

Fig 7.2.4 Test Blocks on Ganache

CHAPTER 8

Results & Discussions

8.1 Results

Our proposed system will track the Blood throughout the supply chain process. It achieves secure and reliable blood supply in following way:

- Firstly, it authorizes the data collection and updating process by validating the users and the data entered by them.
- Secondly, the data integrity is maintained as the transactions on network are immutable. There is no database structure to manipulate the data even after the user has been authorized, which provides additional security from attackers.
- Transportation time can be reduced in critical cases by asking the hospitals in near range.
- Blood wastage can be reduced by supplying near expiration blood first.

The web application will provide the following features for four entities viz. Admin, Donor, Blood bank and Hospital :

- *Admin* will add blood banks and hospitals to the system.
- *Blood bank* on collecting the blood will enter blood bag details along with donor details and submit the data on the Blockchain. Banks will have access to the list of all the bags collected by it.
- *Donors* will be able to track the ownership along with other details of all his/her donated blood bags.
- *Hospitals* will get a list of all the blood bags available at blood banks as well as other hospitals and will have an option to buy them.

8.2 Discussions

Previous papers [5] and [7] propose a traceability system but do not show the implementation of the same. We developed “Blood Cycle”, an Ethereum based system which keeps tracks of the blood bags and allows the actors to make transactions on the website with the help of a smart contract. Results can be seen as the relevant info of the blood bag gets updated after each authentic transaction and is visible to every actor including the donor.

We have presented a new system for blood supply process based on Ethereum technology. In our sample use cases, we have demonstrated the process from the Blood Collection phase at BCC to the hospital storing the Blood bags. Implementation of this system would require the existing blood management systems to be linked on the network. Also, all the required actors as per the business logic would be needed to set-up machines and install Ethereum. These shall complement the system to make sure it works accordingly.

CHAPTER 9

Conclusion & Future Scope

9.1 Conclusion

By applying required checks at every stage in the Blockchain-based SCM, we have ensured that a scarce and perishable resource like blood is properly collected, stored and dispatched according to their priority for patients. Since the whole process is transparent with the help of the shared ledger maintained, the donors can easily track their donations. It would make a great impact on increasing the trust factor and consequently, the donation rate.

The existing cold blood chain system lacks reliability, proper waste management system, and immutable transactions. With the addition of Blockchain to Supply Chains, it now records the transactions which bolster reliability, maintain logs and provide traceability. The security of payments and waste management factors have been considered by including smart contracts.

9.2 Future Scope

With respect to moving this project in the future, we would aim to implement two key aspects that might further enhancing the entire chain. Firstly, we would design the system to handle the splitting of blood in to different key components such as plasma, WBC's and RBC's are medically recommended. Secondly, in order to generate a better view against waste management, we would design a data visualizer. This visualizer would track the details regarding where each blood bag is stored and for how long, thus analyzing the cause of unexpected expiration, if any. It would also track the nominal data, such as how regular is a donor donating, how much blood is present in zones that are more prone to accidents and disasters.

REFERENCES

[1] Times of India, Article on Blood Wastage in Maharashtra. (2017).

[2] el Maouchi, Mourad & Ersoy, Oğuzhan & Zekeriya, Erkin. "TRADE: A Transparent, Decentralized Traceability System for the Supply Chain". (2018).

[3] Sylim, Patrick & Liu, Fang & Marcelo, Alvin & Fontelo, Paul. "Blockchain technology for detecting falsified and substandard drugs in the pharmaceuticals distribution system". In: JMIR Research Protocols. Vol. 7. (2018).

[4] Feng Tian. "An agri-food supply chain traceability system for China based on RFID & blockchain technology". In: Service Systems and Service Management (ICSSSM), 13th International Conference on. IEEE, pp. 1–6, (2016).

[5] Davis, R. & Geiger, Bradley & Gutierrez, Alfonso & Heaser, Julie & Veeramani, Dharmaraj. "Tracking blood products in blood centers using radio frequency identification: A comprehensive assessment". In: Vox Sanguinis, vol. 91, pp. 50-60, (2009).

[6] Jabbarzadeh, F. Behnam and S. Stefan. "Dynamic supply chain network design for the supply of blood in disasters: A robust model with real-world application". In: Transportation Research Part E: Logistics and Transportation Review, vol. 70, pp. 225-244, (2014).

[7] Kim S & Kim D. "Design of an innovative blood cold chain management system using blockchain technologies". In: ICIC Express Letters, Part B: Applications. 9. (2018), pp. 1067-1073.

PUBLICATIONS

Gor, Pranav & Gosalia, Deep & Jhaveri, Rahil & Gawade, Aruna. **"Validity and Integrity check in Supply Chain Management using Blockchain"** , Published by :International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181, Vol. 9 Issue 02, February-2020, IJERTV9IS020421.

ACKNOWLEDGMENTS

We would like to thank Prof. (Mrs.) Aruna Gawade, our project guide for her constant support and guidance without which our project could have been stuck at many points. We would also like to thank Dr. (Mrs.) Meera M. Narvekar, Head of Department, Computers, Dr. Hari Vasudevan, Principal, Dwarkadas J. Sanghvi College of Engineering, for allowing us to use the college labs and classrooms for our project.