

# Design Decisions

Version 0.1 – 2019.02.25

*Created 2019.02.28*

## Charity Chain

A Blockchain based Charity donation management system

Submitted By

Akshay Srinivas ( Developer )

Mansoor O P ( Developer )

Table of Contents

**1 INTRODUCTION .....3**

1.1 DOCUMENT SCOPE..... 3

1.2 INTENDED AUDIENCE ..... 3

1.3 PROJECT OVERVIEW ..... 3

**2 HIGH-LEVEL DESIGN .....3**

2.1 GOALS AND GUIDELINES ..... 3

2.2 ARCHITECTURAL STRATEGIES ..... 4

2.3 HIGH-LEVEL COMPONENT VIEW ..... 7

2.4 HIGH-LEVEL DEPLOYMENT VIEW ..... 8

**3 DETAILED DESIGN .....8**

3.1 LOGICAL VIEW ..... 8

3.2 DETAILED COMPONENT VIEW ..... 8

3.3 PROCESS VIEW ..... 8

3.4 USER INTERFACE FLOW MODEL ..... 9

**4 APPENDIX .....10**

**5 GLOSSARY .....11**

## 1 Introduction

### 1.1 Document Scope

This document shares the information about how we came to this idea of developing software & What is the study we made behind before we started making the application. All the technologies we chose and the process flow of the application. How the application makes use of the blockchain technology efficiently so that a full-fledged application can be used for Charity organisations.

### 1.2 Intended audience

We have developed this application mostly for helping the Charity Organisations. The distribution of Charity donations is managed in this system. Also, the people who are willing to donate to charity organisations. They can also register to our application and buy some of the HHC tokens. HHC is the Token we are using here. Also the Goods Vendor. The people who give Charity organisations goods delivery. These are the main 3 audiences we are expecting in our application.

### 1.3 Project Overview

We are building a Blockchain Based Dapp on top of Ethereum Blockchain which helps the Management of Donations to Charity Organisations. That means the people who are willing to donate to charity organisations and including the Goods vendor. The application using most of the features of Blockchain which make it Transparent to all audience, Distributed and Tamperproof. We have developed the software in Angular frontend and Blockchain Backend. We used Solidity programming language to do all the Blockchain based operations and Angular for the frontend based applications.

The application got main 3 tabs, User registration, Donation, Token Request, Profile. These help the users to make Charity donations smoother and easy. The UI is really simple so that anyone even if he is not into computers can be operated heedless. We are using the popular ERC20 Token system inside our application with some of our modifications. We hope this project will change the Donation systems to Charity organisation to an all new level.

We have given a Video tutorial of Application install & Working uploaded to Youtube,  
Link: <https://youtu.be/K763uQVgYAc>

Also, we are sharing the GitLab repository for this application here. It is opensource and Licensed under Apache 2.0

Link: <https://gitlab.com/akshaysrinivas/charity-chain-angular>

## 2 High-Level Design

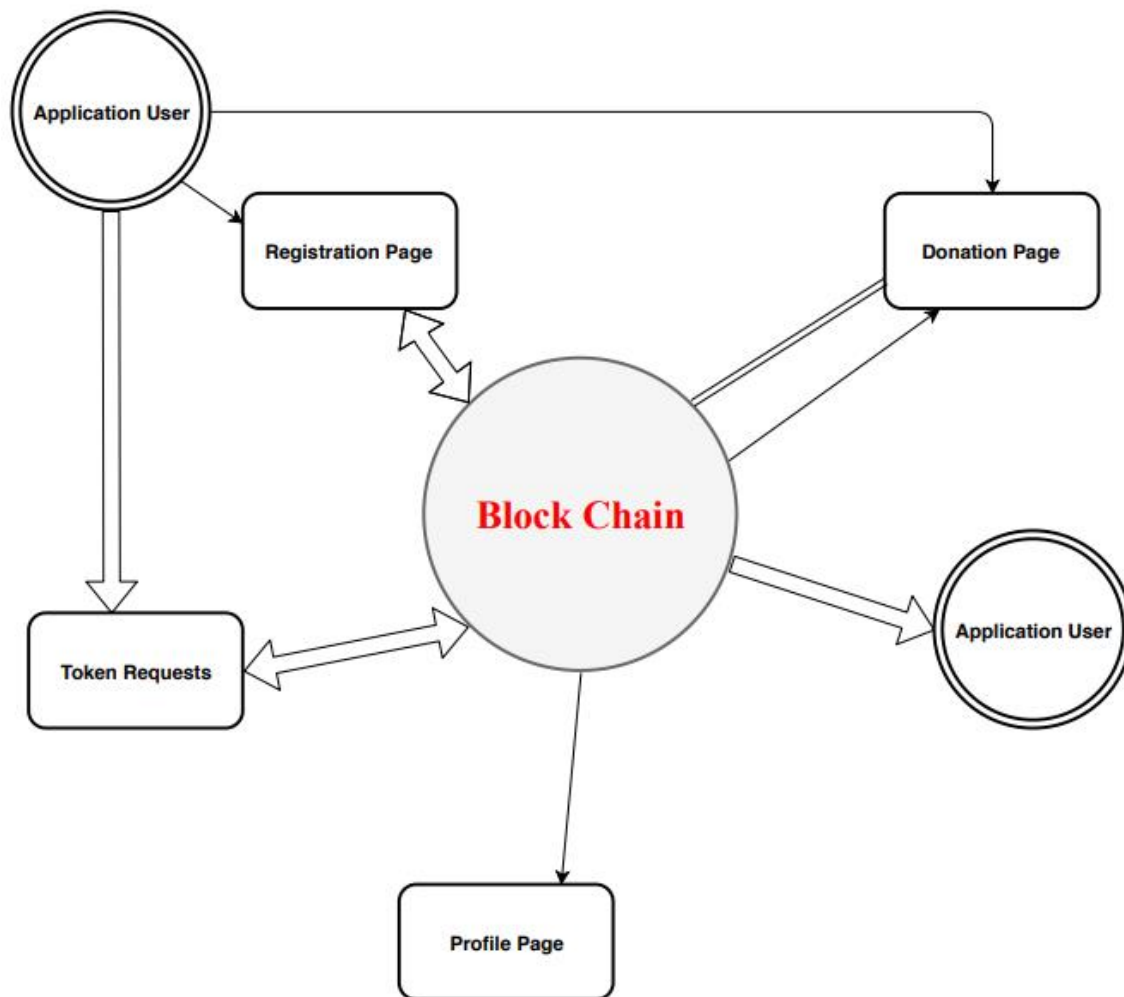
### 2.1 Goals and Guidelines

The Goals & Guidelines of our project is described below :

## CHARITY CHAIN

- The current system of Donations Like 'https://www.giveindia.org/' and many other applications, They are requesting us to donate for them, But we don't know where is the money going and how it is utilized. It's completely a centralized system & the details are not transparent.
- The Blockchain based application in the Charity industry got a big impact since We know what are the significant properties of blockchain & it's the biggest application on Charity Donations.
- Since everything is blockchain based, We can ensure that This is secure in nature and Transparent since all the people using the application can see the total token distributed to each organisation which makes it worth to donate for Charities with smaller donations.
- There is no central system, Money from the user is directly send to Charity Organisation. No middle man, User is happy in a donation.

### 2.2 Architectural Strategies



## CHARITY CHAIN

Given above is the basic architecture of the application. In the picture, we got few users, Application tabs and ultimately Blockchain. As we said we got mainly 4 tabs in the application and they are,

- Profile Page

The profile page is where the user is able to see all his details like Name, EmailID, Password, Type of User and the total amount of Token he has. For all 3 Users, the profile page is the same. Profile page also includes the list of Charity organisations. That is any user with a blockchain address, if he is registered or not, He is able to see all the Charity organisations registered in the application. The Charity organisation details include address and the total amount of Tokens they have.

- Registration Page

This is the page where the user is able to register in the application. He is able to register his Name, Email, Address, Password, What type of User,

1. A person willing to Donate for Charity Organisations
2. Charity Organisations
3. Goods Vendor

The user can choose any of the 3 options on their need. This will complete the User registration in our application.

- Donation Page

This is the page used for Donations. Here there will be 2 type field for the amount of Tokens & User Address. The current user ID and available balance will be shown.






- Token Request Page

This is the page where the user can request for Tokens. Just type in the Token amount & request for the token. Here there will be one more addition, They Token requests can only be seen by the admin. The pending token requests will only be viewed by Admin. Whenever the admin is logged into the application, He can see all Token request and either he can Accept Or Reject.

All the data will be storing in blockchain for security. We have set a limit for our Tokens. It is 31 million HHC tokens.

Now let us see how the Blockchain makes up the application,

We got 5 Solidity contracts,

 CharityFunc.sol	01/03/2019 04:31	SOL File	6 KB
 HelpHandToken.sol	01/03/2019 04:31	SOL File	1 KB
 Migrations.sol	01/03/2019 04:31	SOL File	1 KB
 Owner.sol	01/03/2019 04:31	SOL File	2 KB
 TokenFunc.sol	01/03/2019 04:31	SOL File	4 KB

## CHARITY CHAIN

- Owner.sol

Below are the points and features of Owner Contract.

1. The Owner contract has an owner address and provides basic authorization control.
2. The Owner constructor sets the original `owner` of the contract to the sender.
3. It got a modifier which throws if called by any account other than the owner.
4. transferOwnership function allows the current owner to transfer control of the contract to a newOwner.
5. returnOwnership function allows to return the address of the current Owner

- HelpHandToken.sol

This contract provides the interface for IERC20 token.

- TokenFunc.sol

This contracts imports both Owner.sol & HelpingHandToken.sol, It does the basic functionalities of HHC Token.

1. A modified ERC20 Token specifically for Charity related application.
2. totalSupply returns the total number of tokens in existence.
3. Transfer function transfer token to a specified address and specified amount of tokens.
4. balanceOf function gets the balance of the specified address.
5. transferFrom function Transfer tokens from from address to to address with the specified amount of tokens.
6. The Token constructor sets totalSupply\_ and set balance to the owner of the contract.

- CharityFunc.sol

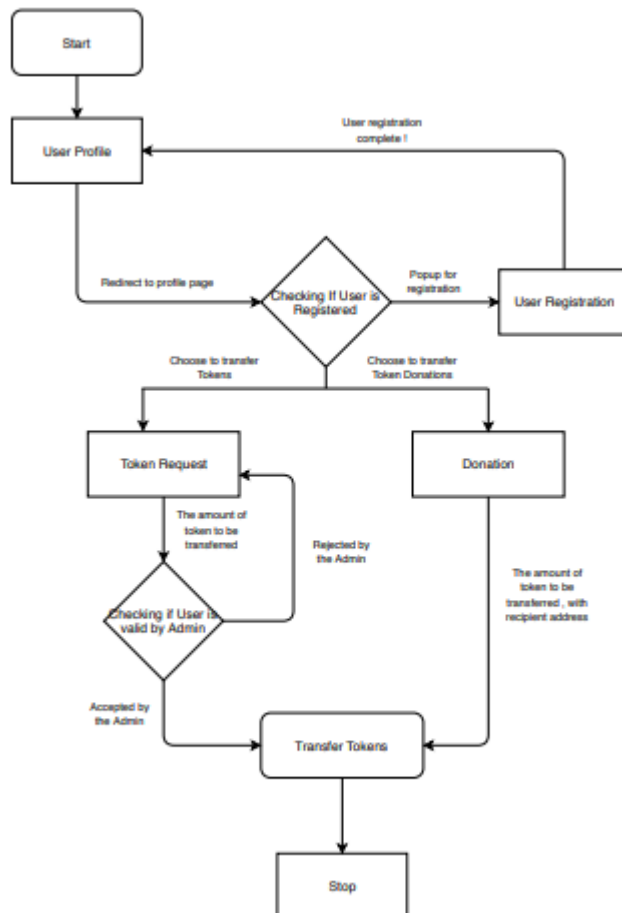
This is the main contract of the application. This contract imports TokenFunc.sol and does the main functions of the Dapp.

1. The CharityFunc contract has the basic functionalities of the application and provides user read , display, request, approval
2. The structure 'godPeople' is for user information store.
3. userID & tokenRequestCount are counters for the number of user registrations & Token requests respectively.
4. The CharityFunc constructor sets the original `owner` of the contract to the sender.
5. userDonation is the mapping of key address with value type of godPeople structure.
6. 'peopleAdd' is an array to store the address of registered users.Inorder to not get in a out of gas exception of dynamic arrays size we are given a limited amount of user for registration fixed to 50.
7. 'requestedUser' is an array to store the address of users requesting for tokens.Inorder to not get in a out of gas exception of dynamic arrays size we are given a limited amount of user for registration fixed to 10.

## CHARITY CHAIN

8. userDetails allows the user to register for new account in our application and details will be saved in blockchain.
9. readDetails allows the application to read information of registered user from blockchain.
10. arrayListGet allows the application to list all the registered users.
11. requestToken is the mapping of key address with value uint256.
12. userTokenRequest allows the user to request for HelpingHandTokens.
13. tokenReqList allows the application to list all the Users requested for Tokens.
14. tokenDetails allows the application to read information of token requested by each user from blockchain.
15. tokenReverse allows the application to delete/remove information of token requested by each user from blockchain.
16. userApproval allows the owner to transfer tokens to requested user if conditions are met & delete the request from requestedUser[].

### 2.3 High-Level Component View



Given above is the High-Level Component view of the application.

### **2.4 High-Level Deployment**

Below describes how to deploy the application in any environment.

1. Open a terminal and make a new directory: - mkdir charitychain
2. Open the folder: - cd charity chain
3. Initiate git inside the folder: - git init
4. Clone the application from our repository: - git clone <https://gitlab.com/akshaysrinivas/charity-chain-angular>
5. Open the folder: - cd charity-chain-angular
6. Install node\_modules : - npm install
7. Remove old build file from the folder: - rm -rf build
8. Compile the contracts: - truffle compile
9. Run the ethereum client 'ganache-cli' in the new terminal in the same folder location. - ganache-cli
10. Step 9 is optional, we have an online Rinkeby client with Infura
11. If you are deploying the application in local machine, Link metamask with Local machine and add ganache accounts. If you are willing to deploy the application in Rinkeby test network login in Rinkeby account in metamask.
12. For deploying in local machine run: - truffle migrate --network development
13. Or deploying in Rinkeby network run: - truffle migrate --network rinkeby
14. Now big time, Run the application: - ng serve
15. Open browser : - <https://localhost:4200>












## **3 Detailed Design**

### **3.1 Logical View**

The project is done in Angular front end. It got 5 components



## CHARITY CHAIN

	meta	01/03/2019 05:25	File folder	
	profilebar	01/03/2019 05:25	File folder	
	registerbar	01/03/2019 05:25	File folder	
	reqtokenbar	01/03/2019 05:25	File folder	
	startbar	01/03/2019 05:25	File folder	
	util	01/03/2019 05:25	File folder	
	app.component.css	01/03/2019 04:31	Cascading Style S...	1 KB
	app.component.html	01/03/2019 04:31	Chrome HTML Do...	1 KB
	app.component.ts	01/03/2019 04:31	TS File	1 KB
	app.module.ts	01/03/2019 04:31	TS File	2 KB
	app-routing.module.ts	01/03/2019 04:31	TS File	1 KB

- Meta component

This is the component where the Donation part is applied and integrated.

- Profile bar Component

This is the component where the profile section is implemented and shown.

- Register bar Component

This is the component where the User registration is implemented.

- ReqTokenbar Component

This is the component where the Request Token features are implemented.

- StartBar Component

This is where the list of Charity organisations listing.

### 3.2 Detailed Component View

Nil

### 3.3 Process View

We are using the Web3.js which is a collection of libraries which allow you to interact with a local or remote ethereum node, using an HTTP or IPC connection.

```
// Checking if Web3 has been injected by the browser (Mist/MetaMask)
if (typeof window.web3 !== 'undefined') {
  // Use Mist/MetaMask's provider
  this.web3 = new Web3(window.web3.currentProvider);
} else {
  console.log('No web3? You should consider trying MetaMask!');

  // Hack to provide backwards compatibility for Truffle, which uses web3js 0.20.x
  Web3.providers.HttpProvider.prototype.sendAsync =
Web3.providers.HttpProvider.prototype.send;
  // fallback - use your fallback strategy (local node / hosted node + in-dapp id
  mgmt / fail)
```

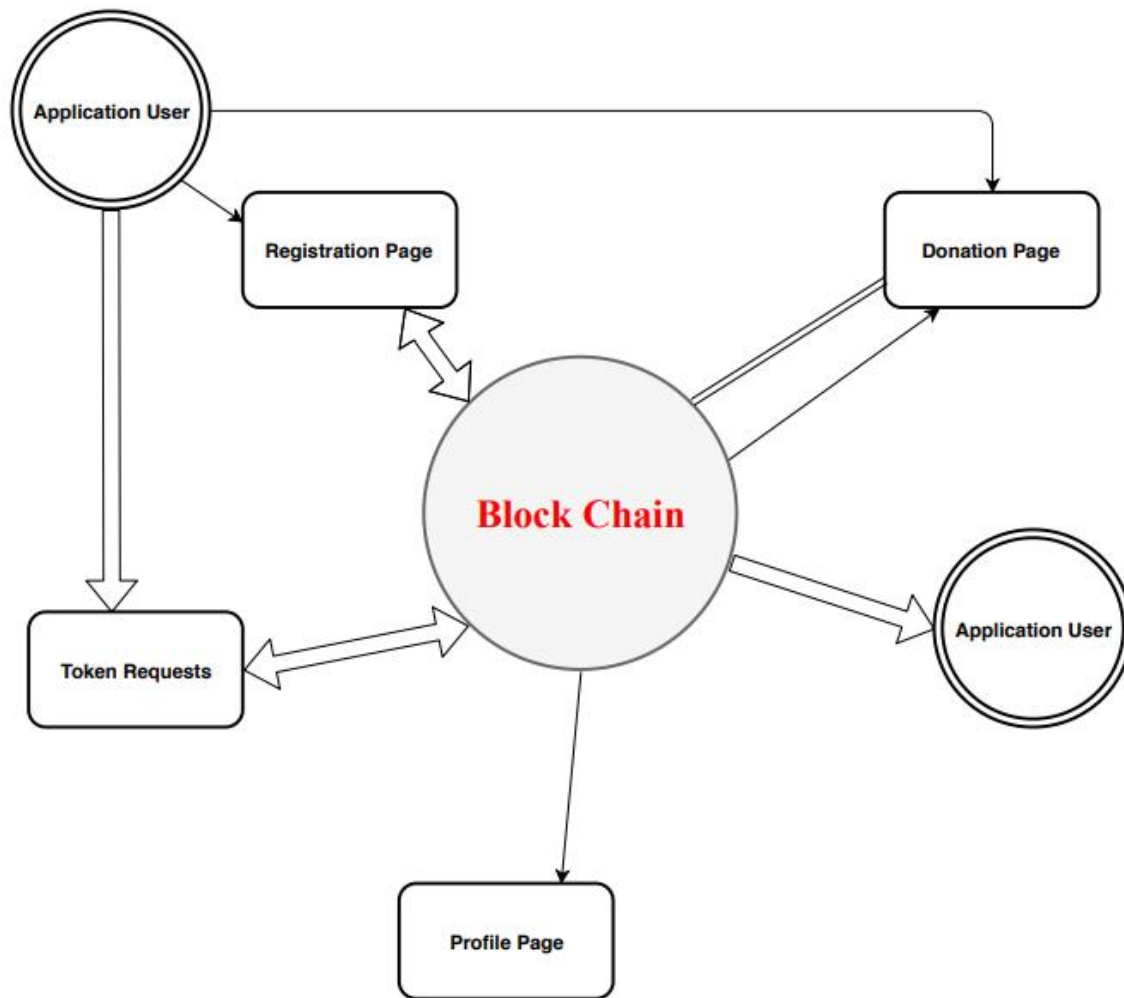
## CHARITY CHAIN

```
this.web3 = new Web3(new  
Web3.providers.HttpProvider('http://localhost:8545'));  
}
```

This is where our application connect with web3 providers. Like Metamaks or localnode. After connecting with the Web3 provide , Our application od ready to rock with ethereum blockchain. All our solidity functions described is called using this web3 interface,

### 3.4 User Interface Flow Model

As same as above,



## 4 Appendix

We have given a Video tutorial of Application install & Working uploaded to Youtube,

Link: <https://youtu.be/K763uQVgYAc>

Also, we are sharing the GitLab repository for this application here. It is opensource and Licensed under Apache 2.0

Link: <https://gitlab.com/akshaysrinivas/charity-chain-angular>

## 5 Glossary

**Solidity:** Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).

**Web3.js:** web3.js is a collection of libraries which allow you to interact with a local or remote ethereum node, using an HTTP or IPC connection.

**Angular:** Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, an end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

## 6 *Revisions to this Document*

*Template Revised by Akshay Srinivas, February 28, 2019. Rewritten in terms of 4+1 Views, UML 2.0, and User Interface flow models (with some inspiration from Agile Methods).*

*uggested (optional) usage of Use Case diagrams.*