# Investor Protector Project Report

Anuhya Gankidi , Jayanth R.Sheri , Anujot Singh, Xialu Zou
CMPE 272, San Jose State University, CA

***Abstract*—Fundraising made it easy for fundraisers to raise funds as they don't have to knock on doors of banks and financers and get the desired amount in return of interest or offering equity or even through donation, or reward. Blockchain, on the hand, is decentralizing the system of records and control which makes fundraising more transparent and secure. A smart contract is similar to a contract in the physical world, but it is digital and represented by a tiny computer program stored in a blockchain. These smart contracts can be used to implement logic. A method has been proposed here that uses smart contract to manage all the activities performed in a new startup.**

## Introduction

Online fundraising enables people to raise funds for their project. People who are interested in a project can donate by making an online transaction. The donated money goes to the project manager, which he uses to complete the project or to make a product. This existing method of online fundraising has a major drawback. It does not allow contributors to have control over the money they have contributed. Since in the existing method the project manager has all the control over the money contributed, he can very easily perform malicious activities. We address this problem faced by the existing online fundraising platforms by using Ethereum network and smart contract.

The development of Blockchain technology has allowed businesses to build decentralized models. It has derived new methods to conduct transactions and make agreements. One of the technologies that propose an alternative to the traditional model is the smart contract. A smart contract is similar to a contract in the physical world, but it is digital and represented by a tiny computer program stored in a blockchain. These smart contracts can be used to implement logic. A method has been proposed here that uses smart contract to manage all the activities performed in a new startup.

## Tools

### 1. Solidity

Solidity is an object-oriented high-level language used for writing smart contracts in the fundraising dapp. A statically typed language has features such as inheritance, libraries and complex user defined types. C++, python and JavaScript, majorly influenced it. Solidity is compiled to bytecode that is executable by Ethereum virtual machine (EVM). Using solidity developers can write dapps that implement self-enforcing business logic contained in smart contracts, leaving an undeniable and permanent record of transactions .

### 2. Remix

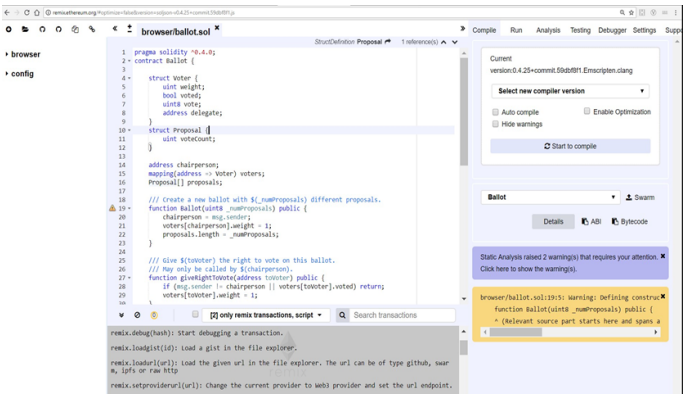
Remix is an integrated development environment (IDE) used for developing smart contracts in solidity. Developing in remix assists developers to find bugs and debug code with ease. Remix supports three different kinds of environments to deploy and run the smart contract:

· JavaScript VM: It creates a mock of blockchain environment so you can test your smart contract functionality.

· Injected Web3: This environment uses a browser plugin or a blockchain based browser such as Mist to connect to any Ethereum network (test or main).

· Web3 Provider: This environment connects to Ethereum node running at localhost and send the transactions to any network (test or main) as specified by the user.

Figure 1: Remix IDE in chrome browser



After writing a smart contract, user can compile it to bytecode and send transactions to Ethereum using Remix as shown in the Figure remix IDE in chrome browser as can be seen in figure 1.

### 3. Visual studio code

Visual studio code is a code editor developed by Microsoft. It is a feature rich editor that supports code highlighting, code debugging, intelligent code completion and code refactoring to build web applications. It supports wide range of programming languages and has a built-in terminal to execute command line commands. As fundraising is a frontend, heavy application with lots of Javascript Visual studio code was a best choice as the editor to develop the application.

### 4. Metamask

Metamask is a chrome browser plugin that acts as a bridge between your browser and Ethereum blockchain by providing a secure identity vault, a user interface to manage multiple Ethereum wallets and sign blockchain transactions [8]. It is one of the best ways to send transactions to Ethereum blockchain because it keeps a track of transaction execution and returns if any error occurs during mining or execution. It supports any ERC20 type token to be added to your wallet and trigger transaction on those ERC20 tokens. It is an Ethereum community open source project having more than million active users; hence, it is the most popular plugin to interact with blockchain.

### 5. Testnet

Test network (Testnet) is a copy of Ethereum blockchain identical in every way to main network except the fact that their Ether is worthless. There are three types of testnets public, private and GanacheCLI[9]. As names suggest, public testnet are available to everyone and connected to the internet, private testnet are similar to one's own blockchain and GanacheCLI is a simulation of Ethereum network on a single computer. For this project, we are using a public testnet called Ropsten.

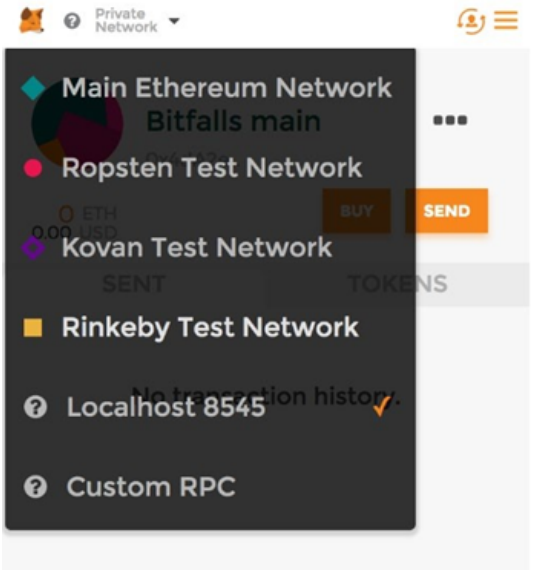



Figure 2: Testnets available in Metamask

Using Metamask we can connect to any Ethereum network as seen in the Figure 2 testnets available in Metamask. Of all the tree networks ropsten resembles to the main net the most.

### 6. Mocha

**Mocha** is a JavaScript test runner that runs both on Node.js and in the browser. It provides functionality for testing both synchronous and asynchronous code with a very simple and similar interface. Mocha provides a variety of interfaces for defining test suites, hooks and individual tests, namely: BDD, TDD, Exports, QUnit and Require.

### 7. Fs extra

fs-extra adds file system methods that aren't included in the native fs module and adds promise support to the fs methods. It also uses graceful-fs to prevent EMFILE errors. It should be a drop-in replacement for fs.

## METHODOLOGY

1. Startup manager create startup profile. To create the startup, he needs to fill the details of the startup. To convince contributors, he also may upload the proposal in PDF format.

2. The startup then will appear in startup main page. If contributors are interested with the startup, he may contribute to the startup.

3. Startup manager create expense request, which consists of list the items needed to execute the startup.

4. Contributors received notifications new expense request has been added.

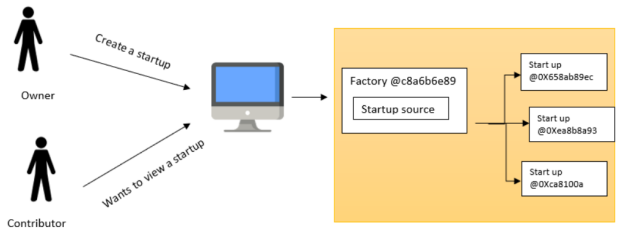
5. Contributors review either the item proposed by startup manager is appropriate or not, if it is appropriate, then contributors can vote to agree with the item listed.

6. If majority of contributors agree, smart contract will send collected funds to the respective vendors.

7. The respective vendor then sends the item agreed to the startup manager. Each transaction is recorded in blockchain and can be seen by all users in Ether scan website.

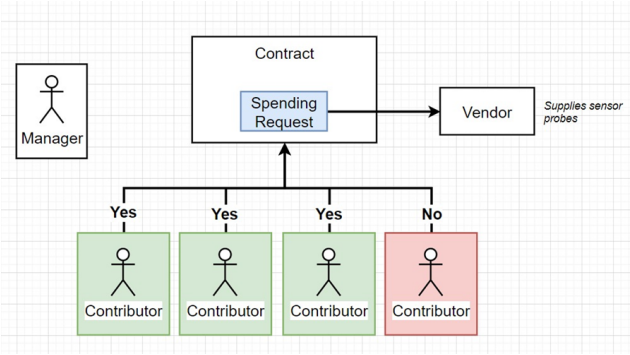
## System Architecture

Figure 3: System Architecture Design




In the above system design, the owner creates a startup profile and the contributor wants to view the same in order to help establish a successful one by contributing funds. A safe transaction is assured using the Ethereum network. The Ethereum network consists of a factory which acts as the source for startup establishment by using Ethereum token network to prevent any kind of fraud from taking place as shown in figure 3.

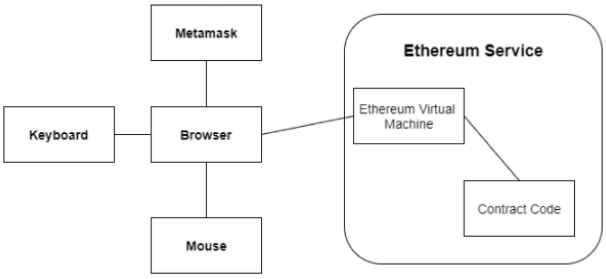
Figure 4: System Explanation

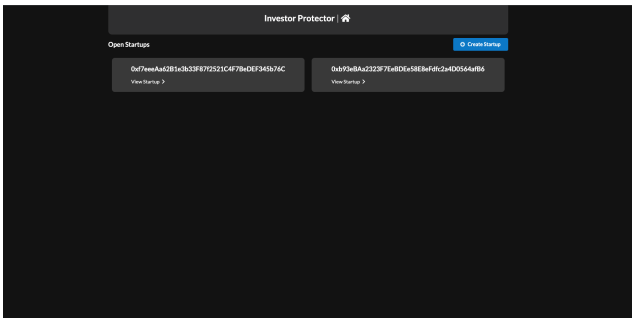


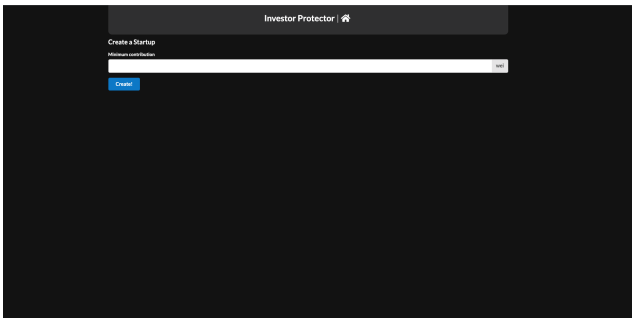

The manager is required to send the spending request and this spending request attempts to withdraw the money from the contract and send to an external address. The manager has the responsibility to create a spending request to an outside vendor. All the contributors have to vote on every spending request. Each of the contributors here, have to approve the spending request by calling a function. If enough people approve the request, then the funds are automatically sent to the outside vendor and in return, the vendor will now deliver some supplies to complete the startup. However, if too many people vote 'no' then that is an indication that it is a bad request and changes need to be undertaken for the successful approval. Yes/No shows the approval in the above diagram as shown in figure 4.

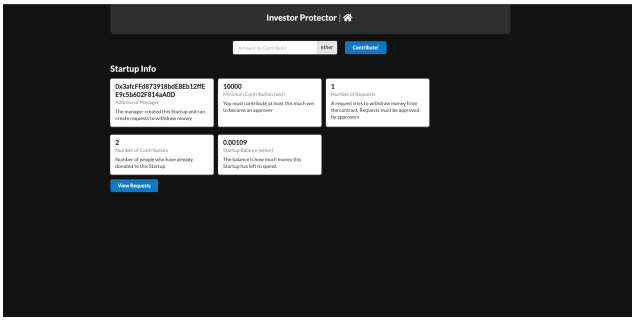
## User Interfaces

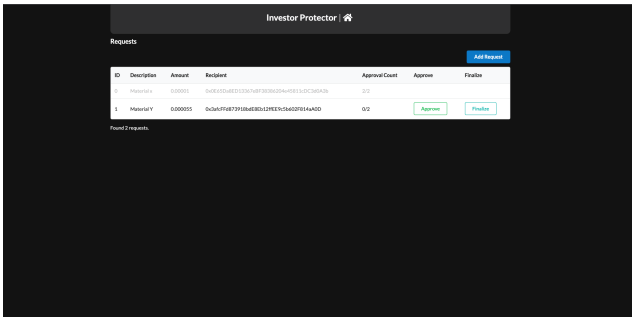
Figure 5: Interface Diagram





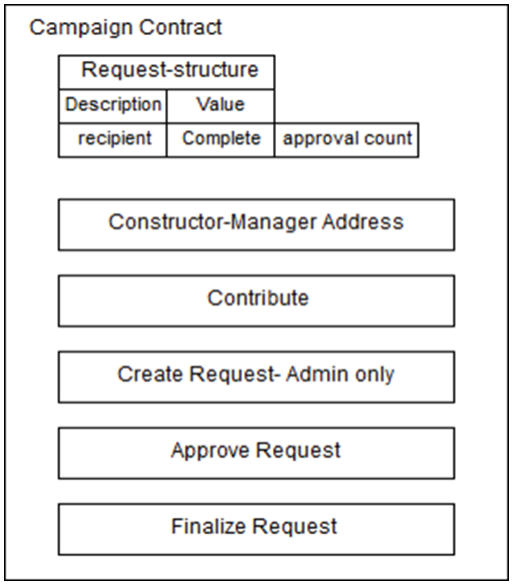

## Algorithmic Approaches Used

Fundraising contract consists of five basic functionalities. These functionalities represent the way in which processes take place. The functions of the proposed smart contract are as following:

- constructor-manager address
- contribute
- create request-admin only
- approve request
- finalize

The detailed structure of a fundraising contract is shown in figure 6.

Figure 6: Structure of fundraising contract




A detailed description of the functionality of different modules in the proposed smart contract has been listed here.

· In the ‘constructor-manager address' function, the contract stores address of its manager (creator). When the generator creates an instance of the startup contract, it also stores the address of the manager of the startup in the contract. This enables the contract to differentiate between manager and contributor.

· The ‘contribute' function, allows people to contribute to the startup. The function defined allows a set of users to contribute to the startup. The minimum contribution value is specified by the manager of the contract.

· The ‘create request-admin only' function in the startup contract allows the manager to generate a payment request. The function stores data regarding payment request in a data structure. The Data stored is describing about the payment request, address of vendor, amount (in ether) to be paid to the vendor, and approvers count.

· In the ‘approve request' function, a voting mechanism for approval of payment request is defined. The mechanism is to be chosen wisely since a wrong mechanism may lead to the execution of more statements and hence the more money requirement.

· The last function is ‘finalize’. It transfers money specified in the payment request to the vendor's cryptocurrency wallet. Only the manager has control over this functionality.

## Project Cost Analysis

Table 1: Cost Analysis

| Service Used | Description | Cost |
|---|---|---|
| Website Hosting | Web hosting is a service that allows organizations and individuals to post a website or web page onto the Internet. | Up to 500 INR/month |
| Gas Cost | Every transaction performed on the blockchain network costs Gas which is effectively paid in Ether tokens (which is money). For each function executed on the blockchain network, there are transaction and execution gas costs. | 22.40 INR for all the functions in a smart contract (Will vary with change in market price of tokens.) |

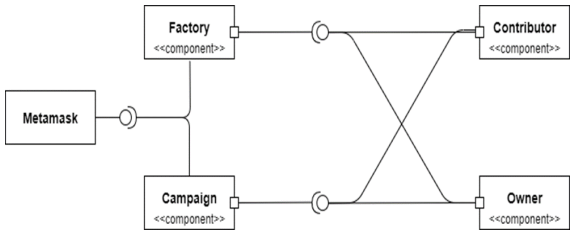
## Project Deployment

The project consists of two phases of deployment as follows:
1. Smart Contract deployment: To deploy a smart contract, you merely send an Ethereum transaction containing the code of the compiled smart contract without specifying any recipients. Deploying a contract also costs ETH, so you should be familiar with gas and fees on Ethereum. Finally, you'll need to compile your contract before deploying it.
What you'll need

· Your contract's bytecode – this is generated through compilation.
· Ether for gas – you'll set your gas limit like other transactions so be aware that contract deployment needs a lot more gas than a simple ETH transfer.
· A deployment script or plugin.
· Access to an Ethereum node, either by running your own, connecting to a public node, or via an API key using a service like Infura or Alchemy.

2. Deploying the dapp itself: the dapp is very similar in deployment to a web app and can simply be deployed on platform like heruko for trial purposes. Deployment is moving a website from a local environment to live servers. What seems like a simple thing can actually be quite complex. There are absolutely loads of ways to go about it. They range from user friendly software and services, to more complex command line tools, to full blown systems with lots of moving parts.

Figure 7: Component Diagram




## Conclusion

Use of blockchain in developing a decentralized platform has a lot of benefits . The use of smart contract and Ethereum has enabled us to implement logic in the blockchain. The use of blockchain architecture makes the proposed system resistant to non-repudiation attacks. Further it minimizes scams, frauds and increase transparency in a fundraising project and will also help the product developer to get the review of their product from the consumer before development.

For the proposed method to be effective we need to verify that its implementation is feasible. To prove the concept, we developed the proposed system and tested it on Rinke by test network. We are going to compile and deploy the Generator and Fundraising contract in the Rinke by test network. As stated earlier every transaction performed in a blockchain network can be monitored and this makes our system resistant to malicious activities.

## APP LINK

http://18.118.226.159:3000/

To use the functions of this project Chrome metamask extension is required

## REFERENCES


W. Kenton, "Donation Based Crowdfunding", Dec 2017, [online] Available: https://www.investopedia.com/terms/d/donationbased-crowdfunding.

N. Szabo, The Idea of Smart Contracts, Oct. 1997, [online] Available: https://www.nakamotoinstitute.org/the-idea-of-smart-contracts/.

R.G. Brown, A Simple Model for Smart Contracts, Oct. 2015, [online] Available: http://www.gendal.me/2015/02/10/a-simple-model-for-smart contracts/.

Solidity Documentation, Oct. 2018, [online] Available: http://www.solidity.readthedocs.org/en/latest/.

C. Cachin, S. Schubert and M. Vukolić, Non-determinism in Byzantine fault-tolerant replication, Oct. 2016, [online] Available: http://www.arxiv.org/abs/1603.07351.

G. Wood, Ethereum: A Secure Decentralized Generalized Transaction Ledger, Oct. 2018, [online] Available: https://www.gavwood.com/paper.pdf.

A.M. Kurian, Interacting with Ethereum Smart Contracts through Web3.js, Oct. 2018, [online] Available: https://www.medium.com/coinmonks/interacting-with-ethereum-smart-contracts-through-web3-jse0efad17977.

Array and Mapping in Solidity, Oct. 2018, [online] Available: https://www.media.readthedocs.org/pdf/solidity-kr/latest/solidity-kr.pdf.

J. Zhang, A Study on Application of Digital Signature Technology, pp. 498-499.

Remix Tool, Oct. 2018, [online] Available: https://www.github.com/ethereum/remix.

Web3, Oct. 2018, [online] Available: https://www.web3js.readthedocs.io/en/1.0/.