Major Project report on

CREATING SMARTCONTRACT FOR CROWDFUNDING

A Dissertation submitted to JNTUH in partial fulfillment of theacademic requirements for the award of the degree.

Bachelor of Technology

In

Computer Science & Engineering

Submitted by

KEESARI ABHINAV
(18H51A05N3)

KUSUMA SHIVANI
(18H51A05N4)

M.HARI KISHAN
(18H51A05N5)

Under the esteemed guidance of
M.Kamala
Assistant Professor

Department of Computer Science & Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC & JNTUH, approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NAAC with 'A' Grade.)

2018 - 2022

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Major Project phase -1 report entitled **"CREATING SMARTCONTRACT FOR CROWDFUNDING "** being submitted by **Keesari Abhinav** ***(18H51A05N4)***, **Kusuma Shivani** ***(18H51A05N4)***, ***M.Harikishan(18H51A05N5)*** in partial fulfillmentfor the award of **Bachelor of Technology in Computer Science and Engineering** is arecord of bonafide work carried out his/her under my guidance and supervision.

The results embodied in this project report have not been submitted toany other University or Institute for the award of any Degree.

M.Kamala	**Dr. K Vijaya Kumar**
Assistant Professor	**Professor and HOD**
Project Guide	**Dept. of CSE**
Dept.of CSE	

Submitted for viva voce Examination held on ____________________

External Examiner

Acknowledgment

We are highly indebted and grateful to our guide **M.Kamala,** Asst. Professor, Department of CSE, CMRCET for his excellent guidance and constant encouragement throughoutfor the successful completion of the Project.

I would like to thank **Dr. K. Vijaya Kumar**, Head of the Department of Computer Scienceand Engineering, for his moral support throughout the period of mystudy in CMRCET.

I am highly indebted to **Dr. V. A. Narayana**, Principal CMRCET for givingpermission to carry out this technical seminar in a successful and fruitful way.

I would like to thank the Teaching & Non- teaching staff of Department ofComputerScience and Engineering for their co-operation

Finally, I express my sincere thanks to **Mr. Ch. Gopal Reddy**, Secretary, CMRGroup of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this technical seminarwork.

SIGNATURE

KEESARI ABHINAV (18H51A05N3)
KUSUMA SHIVANI (18H51A05N4)
M.HARIKISHAN (18H51A05N5)

ABSTRACT

Globally, there is a huge craze for crowdfunding and we have seen so many ideas which are so popular now where crowdfunded in past. Last year in North America itself raised over 17Billions. There are over 3000 crowdfunding platforms in the world and the basic problem in it is to not let the money of the donars into the hands of scammers and the money is still refundable if the donars want there, money in the middle of the project if they think the project will not be succeed So, we have taken the problem statement as mentioned above and decided to introduce the smart contracts into the crowdfunding. So, we have described the existing solutions success rate and working model for the betterment of the existing system with including a new technology in it. By doing so the wastage of money reduces, increasing the trust to the platform.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ NOTATIONS/ NOMENCLATURE

Variables

Manager:	Address of the person who is managing this campaign
Minimum contribution:	Minimum donation required to be considered a contributor or approver
Approves:	List of address for every person who has donated money
Requests:	List of requests that the manager has created

Functions

Campaign:	Constructor function that sets the minimum contribution and the owner
Contribute:	Called when someone wants to donate money to the campaign and become an approver
Create request:	Called by the manager to create a new spending request
Approve request:	Called by each contribution to approve a spending request
Finalize Request:	After a request has gotten enough approvals, the manager can call this to get money sent to the vendor.

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1 ABOUT THE PROJECT

Crowdfunding is a way to finance your business through, loans, donations or exchanging money for rewards or shares in your business. You generally do this through a crowdfunding website. To begin, you'll need to post your business idea (as a campaign) on the website. Crowdfunding is the use of small amounts of capital from a large number of individuals to finance a new businessventure. Crowdfunding makes use of the easy accessibility of vast networks of people through and crowdfunding websites to bring investors and entrepreneurs together, with the potential to increase entrepreneurship by expanding the pool of investors beyond the traditional circle of owners, relative. Although similar concepts can also be executed through mail-order subscriptions, benefit events, and other methods, the term crowdfunding refers to Internet- mediated registries. This modern crowdfunding model is generally based on three types of actors
the project initiator who proposes the idea or project to be funded, individuals or groups who support the idea, and a moderating organization (the "platform") that brings the parties together to launch the idea. Crowdfunding has been used to fund a wide range of for-profit, entrepreneurial ventures such as artistic and creative projects, medical expenses, travel, and community-oriented social entrepreneurship projects. Though crowdfunding has been suggested to be highly linked to sustainability, empirical validation has shown that sustainabilityplays only a fractional role in crowdfunding. Its use has also been criticised for funding quackery, especially costly and fraudulent cancer treatments. Crowdfunding has been around for a while now (the word was first used in 2006). It’s fundamentally a pretty simple idea - raising money from multiple people towards one wider cause or project. And as a concept it predates anything digital; communities all over the world have been working together for hundreds of years and chipping in to fix the community centre’s roof, put up a statue, or even publish literature by subscription. But today, crowdfunding mostly happens online, and plenty of platforms have sprung up to support it as a fundraising method.

1.2 OBJECTIVE

The main objective of the project is Automating the process of investment There is no need for third party Secure data storage Immutability Speed and efficiency Increasing transparency. Making transaction histories more transparent and secure through the use of blockchain technology. Because Blockchain is a type of distributed ledger, all network participants share the same documentation as opposed to individual copies. Contributors can decide Where to invest and can Acknowledge the requests for money made by the Project Creators through their votes. The creator can only use the money if a minimum number of contributors approve a certain request. It will make sure the money is used for Necessities rather than Luxuries.

CHAPTER 2

SYSTEM ANALYSIS

SYSTEM ANALYSIS

2.1 EXISTING CROWDFUNDING SYSTEM

In existing system if all the funds collected from the campaign are spent fairly on the business model, then the funds will not gets wasted and this happens in ideal world shown in figure 1.

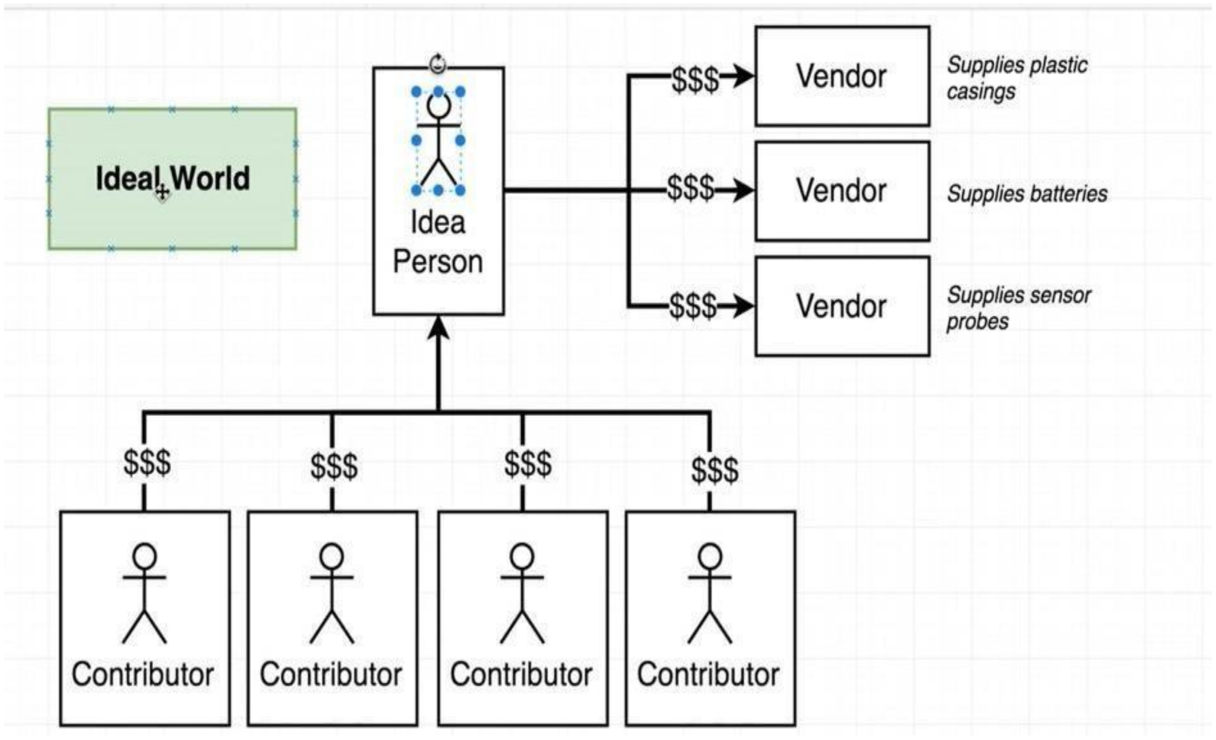


Figure 1: Existing System Block Diagram in Ideal World

But so many situations proved that there can be many scams to take place. The situation in which scamstakes place is called non ideal world situation and it is shown in figure 2.

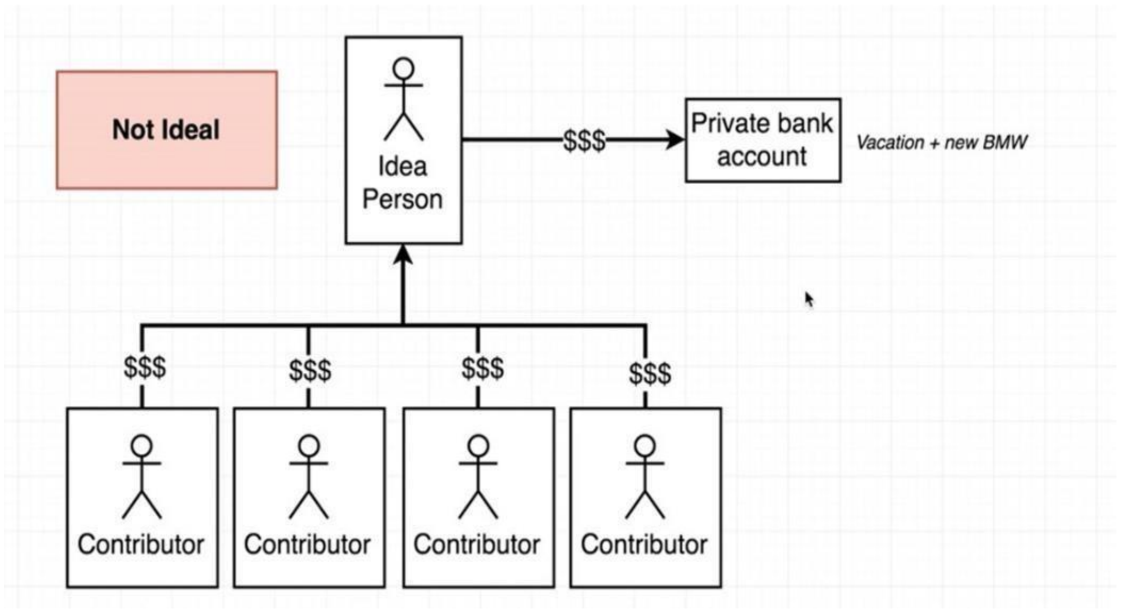


Figure 2: Existing System in Non Ideal World

2.2 PROPOSED CROWDFUNDING SYSTEM

In proposed system we solve the major problem which is not letting the scammers to take away the money which are collected from the campaigns. Here the whole control of the funds will be in the hands of the donors so the funds will not get wasted. Here the smart contract playsa vital role by implementing a voting system which allows the donors to vote and if 51% or more contributers approves then the money will be released and this lets the funds to be spent correctly and wil not give any chance to the managers to waste the funds for their personal use.

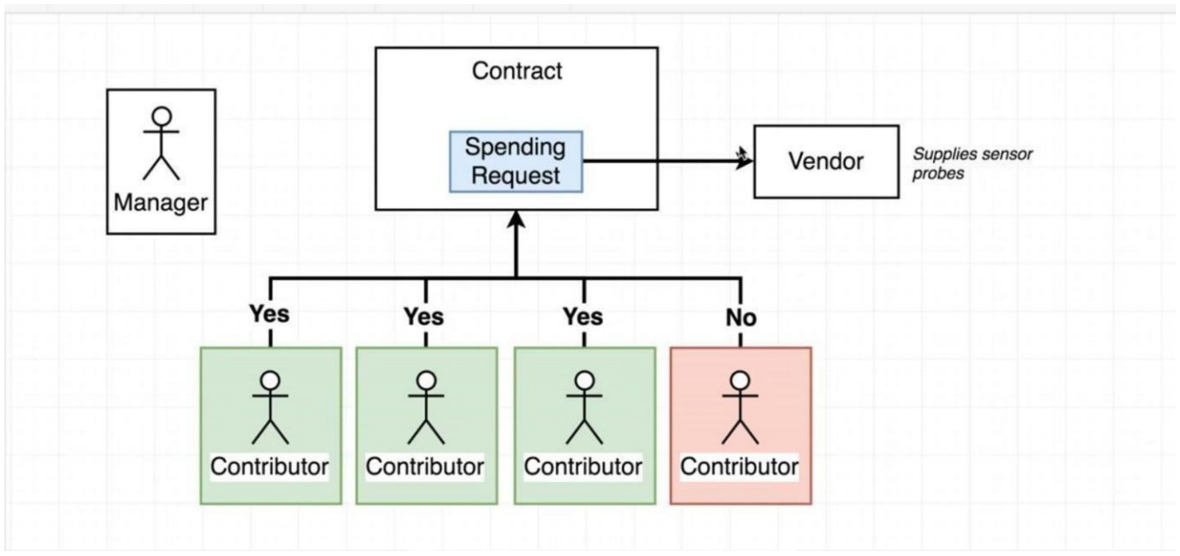


Figure 3: Proposed System Block Diagram

CHAPTER 3

SYSTEM AND REQUIREMENTS

SYSTEM AND REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

- Windows 10
- Git CMD

3.2 HARDWARE REQUIREMENTSMemory

- 2-GB RAM(Minimum)

Processor

- 1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set (Minimum)
- 3.3 gigahertz (GHz) or faster 64-bit dual core processor with SSE2 instruction set (Recommended)

Display

- Super VGA with a resolution of 1024 x 768 (Minimum)

3.3 NETWORK REQUIREMENTS

Model-driven apps are designed to work best over networks that have the following elements:

- Bandwidth greater than 50 KBps (400 kbps)
- Latency under 150 ms

CHAPTER 4

LITERATURE SURVEY

LITERATURE SURVEY

4.1 REVIEW OF LITERATURE SURVEY

4.1.1 **APPLYING ETHEREUM SMART CONTRACTS -** NIK ALINA NIK AHMAD

∗Crowdfunding is one of the most popular method to collect capital. However, various concerns and obstacles still need to be overcome in order to entirely gain benefit from crowdfunding. The information asymmetry is one of the crowdfunding problems where stakeholders receive different information. In response to this problem, this study aims to increase contributors' trust by providing more transparent transactions and reduce the information asymmentry through blockchain implementation in crowdfunding platform. This study reviewed the effects of blockchain implementation in crowdfunding system using Ethereum smart contract. To study the results, a blockchain-based crowdfunding system was developed. The results of this study include comparison between crowdfunding system that implemented blockchain and traditional crowdfunding system in terms of transparency of transactions, immutability of data, CRUD operations, speed of transactions and degree of information symmetry. Trust attributes were also assessed against the developed system. Overall result showed that implementation of blockchain in crowdfunding system provide higher transparency which can reduce the information asymmetry and increase stakeholder's trust to contribute and raise money via crowdfunding platform.

This paper discussed the practicality of blockchain technology in increasing users' trust and decreasing the information asymmetry in blockchain-based crowdfunding system to benefit both fundraisers and contributors. Therefore, in order to increase users' trust through a more transparent transaction, a decentralized blockchainbased crowdfunding system was proposed. Having the information copied and spread across a network will ensure a direct change reflection at all locations whenever a new block is added into the blockchain. By utilizing smart contract to automatically execute the contract based on certain conditions lead to transaction transparency, allow the contributors to be aware about how the fund are invested as every transaction made in blockchain is transparently recorded. The results showed that implementation of blockchain in crowdfunding system positively affect most of the trust attributes and able to minimize fraud cases and information asymmetry as data cannot be easily tampered. As a result, contributors are able to donate to the campaigns confidently without worrying about the funds exploitations. In this study, the usage of smart contract tokens was not covered. However, with its implementation, it is able to provide better controls on how Ether can be processed. Therefore, future study could involve introducing other technology which can be used in blockchain to query data and more research should be conducted on improving the overall user experience on how Ethereum transfers can be accelerated.

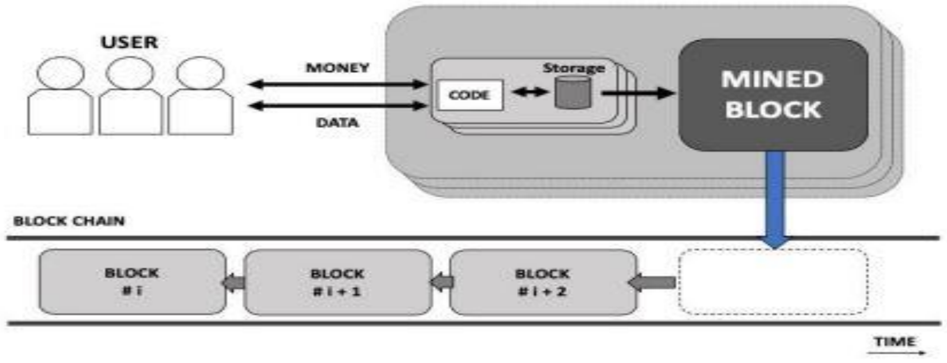
4.1.2 SMART CONTRACT FOR CROWD FUNDING

- FIRMANSYAH ASHARI1 , TETUKO CATONSUKMORO2 , WILYU MAHENDRA BAD3 , SFENRANTO4 , GUNAWAN WANG5

In the situation of the Covid-19 plague, many organizations are raising funds to help local governments to obtain additional sources of funds that will be distributed to those in need. Trust is an important factor for the parties involved in raising these funds, both in terms of the funder, the service provider of fundraising platform and even the fundraiser. On this occasion, the author tries to analyze how to implement blockchain technology and smart contracts in the dominant schemes of crowdfunding process. the results of this study indicate that blockchain-based smart contracts can be applied to the dominant schemes of crowdfunding process.

Smart Contracts based on Blockchain technology are very suitable to be implemented in 3 dominant crowdfunding process schemes. Beside increasing trust due to blockhain, with smart contract also shorten main process in fundraising. Implementation of Smart Contract based on Blockchain technology, requires high-cost if the organization takes the initiative to implement this technology using their own resources. In general, smart contract service providers use cryptocurrency, where not all governments legally recognize the use of these currencies.

Smart Contract A smart contract is a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code. The code and the agreements contained therein exist across a distributed, decentralized blockchain network. The code controls the execution, and transactions are trackable and irreversible. Smart contracts permit trusted transactions and agreements to be carried out among disparate, anonymous parties without the need for a central authority, legal system, or external enforcement mechanism. While blockchain technology has come to be thought of primarily as the foundation for bitcoin, it has evolved far beyond underpinning the virtual currency



Smart contract consists of account balance, personal storage and an executable code. the smart contract status is stored in one interconnected block and is updated every time the contract is called. the code in the blockchain cannot be entered into after the contract is entered. The contract can be executed by sending a transaction to a unique address 20 bytes from the contract. Subsequently the contract is executed by the miners in the network to reach consensus output then the status of the contract will be renewed. contracts can read or write transactions to personal storage and save money into their accounts and receive messages or money from other users or can make new contracts. Deterministic and non-deterministic are types of smart contracts. Deterministic Smart Contract is a contract that when executed does not require information from external parties (from outside the blockchain). Non-deterministic contracts are contracts that depend on information are contracts that depend on information (ex: Oracle or data feed) from external parties. For example a contract to get weather information which is one example of information that is not available on the blockchain.

4.1.3 DECENTRALIZED PLATFORM FOR CROWD FUNDING

- SIDDHESH JADYE1, SWARUP CHATTOPADHYAY2, YASH KHODANKAR3, DR. NITA PATIL4

In today's world, blockchain-based systems are in demand across various industries, because of its secure, trusted, and decentralised network as well as for being more efficient than the traditional methods. However, the traditional ways these days are facing a lot of issues and challenges because of the complex and less secure network. Blockchain network integration overcomes the problems faced by traditional methods across industries. The Blockchain integrated network provides benefits such as increased security, increased transparency, increased efficiency and decreased chances of fraud. Although the blockchain-based systems provide various benefits, due to lack of knowledge about this technology, the implementation rate is low. In this work, we have highlighted the distinction between the traditional crowdfunding platform as well as blockchain network-based crowdfunding platform and the benefits of implementing blockchain network in other sectors. This work highlights the issues and challenges faced by the industries, as mentioned earlier, by using the traditional methods as well as the solutions to the problems provided by the blockchain network-based systems to those industries. This work helps the people to understand the benefits of blockchain network-based systems in their respective industries as well as execute it to improve the transparency, efficiency, and security of the system altogether.

We can conclude that the implementation of blockchain can improve many drawbacks that area unit gift within the ancient crowdfunding platforms. This includes, increased security, increased potency, fraud protection. In gist, we are able to say we've got achieved the subsequent things:

- Decentralization
- Fraud interference victimisation e-Voting
- Secure systems with Smart Contract
- increased potency
- Tokenization
- Having the ability to serve high demand
- Having the ability to retain all the positives of the standard Crowdfunding Platforms

Working of Smart Contract:

A new technique for crowdfunding supported blockchain technology has been advised during this section. Ethereum is associate ASCII text file, public, blockchain-based distributed computing platform and OS that includes good contract (scripting) practicality. The good contract permits us to implement business logic and runs on blockchain network. Solidity is the most well-liked language for writing a wise contract good go for associate ethereum network permits America to exchange cash, share, or something of import in an exceedingly clear and conflict-free manner. This property of good contract enables America to use it in varied eventualities. Two good contracts are developed as delineated in the primary good contract deploys the second contract every time a replacement fundraising campaign is initiated. The second contract consists of all the logic that's needed for running the campaign. The primary contract has been noted as a generator and the second contract is noted as a campaign contract. When a personal starts a replacement campaign the generator deploys an associate instance of the campaign contract on the ethereum network. The deployed contract stores and manages the money that is contributed to the campaign. The leader of the campaign is noted because the manager of the campaign. The manager, when having collected the specified quantity of cash (in the shape of ether) generates a payment request. This payment request must be approved by quite fifty % of its contributors. If the payment request has the specified number of approvers then the manager will end the request and transfer needed cash to the seller from the fund collected. As delineated the manager when making a campaign has two functionalities. The in which the manager requests the payment from the contributors. The payment request is finalized when the campaign has gathered the required range of positive approvers.

4.2 SYSTEM AND METHODS

4.2.1 System 1: Kickstarter

Kickstarter is another popular choice. As of 2021, since it was founded in 2009, Kickstarter has successfully funded nearly 200,000 projects, with more than $5.7 billion pledged across all Kickstarter projects. Kickstarter is the most popular crowdfunding site for aspiring businesses hoping to raise capital and reach a larger audience. In fact, unlike GoFundMe, Kickstarter can only be used for creating projects that can be shared with others. Kickstarter is one of a number of crowdfunding platforms for gathering money from the public, which circumvents traditional avenues of investment. Project creators choose a deadline and a minimum funding goal. If the goal is not met by the deadline, no funds are collected (a kind of assurance contract).

The kickstarter platform is open to backers from anywhere in the world and to creators from many countries, including the US, UK, Canada,[1] Australia, New Zealand, The Netherlands, Denmark, Ireland, Norway, Sweden, Spain, France, Germany, Austria, Italy, Belgium, Luxembourg, Switzerland and Mexico.

Kickstarter applies a 5% fee on the total amount of the funds raised. Their payments processor applies an additional 3–5% fee. Unlike many forums for fundraising or investment, Kickstarter claims no ownership over the projects and the work they produce. The web pages of projects launched on the site are permanently archived and accessible to the public. After funding is completed, projects and uploaded media cannot be edited or removed from the site.

There is no guarantee that people who post projects on Kickstarter will deliver on their projects, use the money to implement their projects, or that the completed projects will meet backers' expectations. Kickstarter advises backers to use their judgment on supporting a project. They also warn project leaders that they could be liable for legal damages from backers for failure to deliver on promises. Projects might also fail even after a successful fundraising campaign when creators underestimate the total costs required or technical difficulties to be overcome.

When asked what made Kickstarter different from other crowdfunding platforms, co-founder Perry Chen said: "I wonder if people really know what the definition of crowdfunding is. Or, if there’s even an agreed upon definition of what it is. We haven’t actively supported the use of the term because it can provoke more confusion. In our case, we focus on a middle ground between patronage and commerce. People are offering cool stuff and experiences in exchange for the support of their ideas. People are creating these mini-economies around their project ideas. So, you aren’t coming to the site to get something for nothing; you are trying to create value for the people who support you. We focus on creative projects—music, film, technology, art, design, food and publishing—and within the category of crowdfunding of the arts, we are probably ten times the size of all the others combined. To maintain its focus as a funding platform for creative projects, Kickstarter has outlined three guidelines for all project creators to follow: creators can fund projects only; projects must fit within one of the site's 13 creative categories; and creators must abide by the site's prohibited uses, which include charity and awareness campaigns

Additionally, Kickstarter cannotbe used to raise funds to donate to a charity or cause, projects can't offer incentives like equity, revenue sharing, or investment opportunities, nor can any project involve the site's list of prohibited items such as "any item claiming to diagnose, cure, treat, or prevent an illness or condition," political fundraising, drugs or alcohol, or any contests, coupons, gambling, and raffles. Kickstarter is one of number of a crowdfunding platform for gathering money from the public which circumvents traditional avenues of investments. It applies 5% fee on the total amount of funds raised. Project creators choose a deadline and a minimum funding goal, if the goal is not met by the deadline no funds are collected. Kickstarter is a normal web application fora crowdfunding platform which don't use smart contract. A project is a finite work with a clear goal that you'd like to bring to life. Think albums, books, or films. The funding goal is the amountof money that a creator needs to complete their project. Funding on Kickstarter is all-or-nothing. No one will be charged for a pledge towards a project unless it reaches its funding goal. This way, creators always have the budget they scoped out before moving forward. A creator is the personor team behind the project idea, working to bring it to life. Backers are folks who pledge money to join creators in bringing projects to life. Kickstarter is not a store, backers support a creative process.Rewards are a creator's chance to share a piece of their project with their backer community. Typically, these are one-of-a-kind experiences, limited editions, or copies of the creative work being produced.

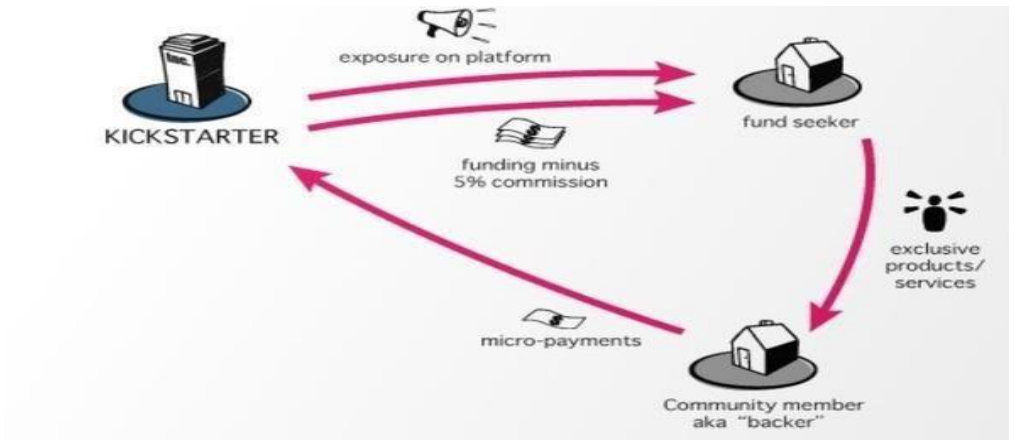


Figure 4: Working Model of Kickstarter

BASIC PROBLEM IN EXISTING SYSTEM

Type of Project	Number of Projects
Unsuccessfully Funded Projects	328518
0% Funded Projects	55991
The total projects that wasted money	328518 – 55991 = 272527

Table 1: Kickstarter Statistics

So the total projects which are unsuccessful along with wasting the amount of contributors is 272527.

This is because lack of control of money in the hands of contributors so using smart contracts we can solve this issue.

4.2.2 SYSTEM 2: FUNDSURFER

Fundsurfer is a crowdfunding platform with different types of money can be donated. Both fiat and bitcoin can be donated by the backers.

The fees are only taken from a projects funding total when it is successfully funded and your backers have made payment.The charges are for platform, bank transfer, crypto transfer, non European cards .As bitcoin can also be funded, we can transfer the funds to every corner in the world.Fundsurfer is not only for creative projects and also for charity. Analysis was performed on a sample dataset containing data about 4121 campaigns launched over Kickstarter platforms in the month of April 2014. Out of 4,121 projects, 1,899 (46%) are successful and 2,232 (54%) are unsuccessful. This dataset comprises of projects in all fifteen categories as classified by Kickstarter such as music, dance, etc. Tweet messages were collected using Twitter API. To collect a corpus of messages related to above mentioned campaigns, all the text messages posted on Twitter platform between 1st April 2014 to 5th June 2014 mentioning hashtags “#Kickstarter” and “@Kickstarter” were retrieved. Around 135,375 tweet messages were obtained. These tweetmessages consist of the text message posted, date of posting of message, number of retweets, favorites' and hash tags. These tweet messages were then processed and all noisy data such as stop words, tags were removed and stored in a single csv file in proper format. Messages other than in English language were also removed.

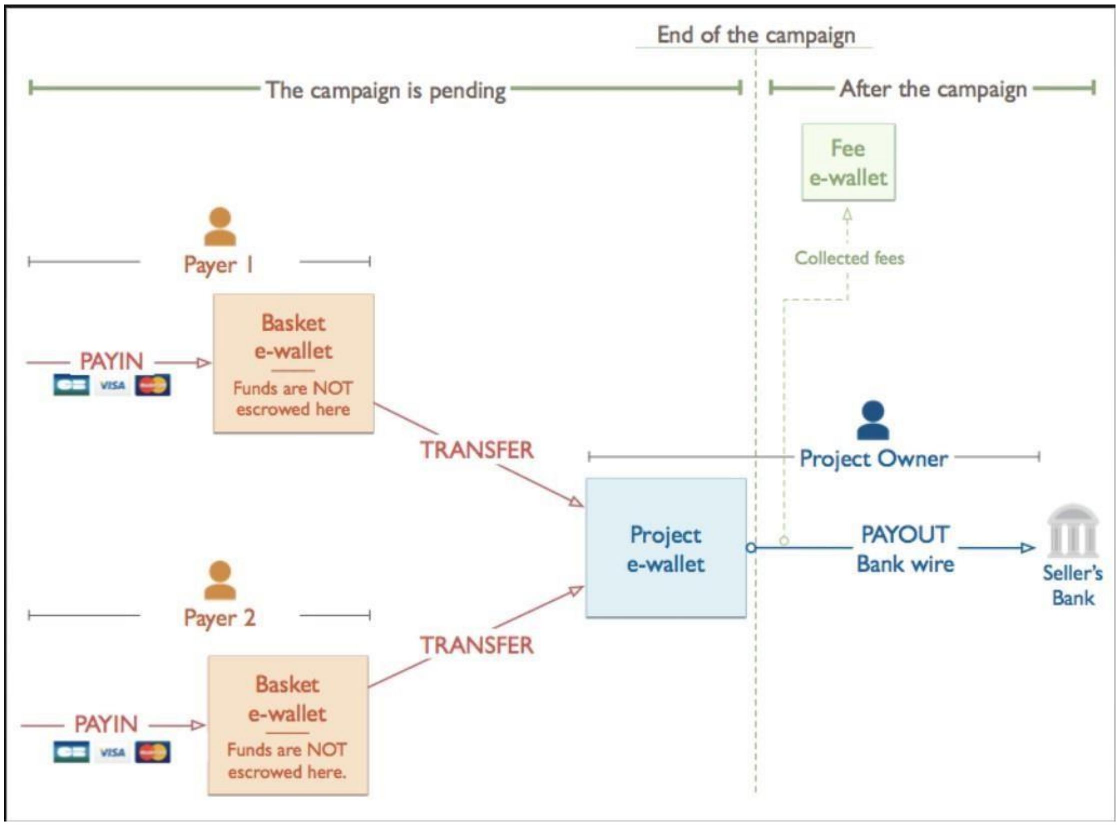


Figure 5: Working of Fundsurfer

4.2.3 SYSTEM 3: INDEGOGO

Indiegogo is a American crowd funding website, it allows people to solicit funds for an idea, charity and start up business.Charges 3% for bank transaction.The site runs on a reward based system.Indiegogo launched Indiegogo Life , which is a fund raising platform for medical expenses and emergencies later acquired by GoFundMe.The campaign owners may not create a campaign that rise funds for illegal activities such as alcohol, drugs weapons .Indiegogo also funds for creative projects.Indiegogo is one the two most popular crowd funding websites.It has looser guidelines than kickstarter , letting user funds campaign that kickstarter would not.

Ex: kickstarter would not allow funding for healthcare projects.

from any category a year after its launch in 2007.Indiegogo is seen as a less strict and more flexible platform than Kickstarter, as it gives backers control over whether they want fixed or flexible models this is probably the most significant difference between the two crowdfunding platforms. Kickstarter releases funds only after the campaign has reached its funding goal, whereas Indiegogo allows the campaigner to receive funding pro-rata, or wait until their target is hit.As a campaigner, it might be easier and less risky to go with flexible funding (i.e., receiving funds as they come); however, regardless of the amount raised, campaigners must still deliver on any promises made. For a backer, fixed funding is more attractive as it is associated with muchless of a risk.

Get set up

Our platform gives you freedom and flexibility to customize your campaign as you want: You get to decide on your fundraising target, create perks to reward backers, and set a campaign deadline (up to 60 days)

Launch your campaign

You decide when to make your campaign live – no approvals necessary. Once your campaign is launched, be sure to tweet, email, and shout from the rooftops – tell everyone!

Continue raising with In Demand

After your crowdfunding campaign is over, continue raising money and building your communityfor as long as you’d like. No fundraising target, no time limits.

CHAPTER 5

TOOLS AND LIBRARIES

TOOLS AND LIBRARIES

5.1 TOOLS:

REMIX:

Remix IDE is an open source web and desktop application. It fosters a fast development cycle and has a rich set of plugins with intuitive GUIs. Remix is used for the entire journey of contract development as well as being a playground for learning and teaching Ethereum.

VISUAL STUDIO CODE:

Visual Studio has the open tools and flexibility you need to create and deploy modern web applications

SEMANTIC UI:

Semantic UI is a front-end development framework similar to bootstrap designed for theming. It contains pre-built semantic components that helps create beautiful and responsive layouts using human-friendly HTML.

RINKEBY:

Ethereum Rinkeby Faucet. This faucet uses a public test network where you can receive or send transactions without spending real money.

SOLIDITY:

Solidity. Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state. Solidity is a curly-bracket language . It is influenced by C++, Python and JavaScript, and is designed to target the Ethereum Virtual Machine (EVM).

METAMASK:

Meta Mask is a Crypto Wallet and Your Gateway to Web3 Buy, store and send tokens globally Explore blockchain applications at lightening speed Choose what to share and what to keep private. Trusted by over 1 million users worldwide.

NEXT.JS:

The Next.js is React Based framework with server side rendering capability. It is very fast and SEO friendly. Using Next.js, you can create robust react based application quite easily and test them.

5.2 LIBRARIESWEB3:

The Web3.js library is the principal JavaScript library you will use in Web3applicationdevelopment Primarily, you will use this library for interacting with the Ethereum blockchain.

REACT:

It is designed specifically for building user interfaces.

CHAPTER 6

PROJECT PIPELINE

PROJECT PIPE LINE

6.1 SMART CONTRACTS:

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. Smart contracts are a type of Ethereum account. This means they have a balance and they can send transactionsover the network. However they're not controlled by a user, instead they are deployed to the network andrun as programmed. User accounts can then interact with a smart contract by submitting transactions thatexecute a function defined on the smart contract. Smart contracts can define rules, like a regular contract, and automatically enforce them via the code. Smart contracts cannot be deleted by default, and interactions with them are irreversible.
Globally 34B raised by crowdfunding because its not that donars are willing to donate the money its because donars are trusting the platform and the product. We should not take it for granted and raise money and put the money into vain, there is a responsibility to be taken by the creators and the platformfor the trust the donars have put on them.

6.2 SMART CONTRACT DESIGN:

We implemented two Smart Contracts typologies using the ERC721 token that we call JobOfferManager and Employment which are already registered and ready in the webApp. The platform can customize theSmart contracts by the data inserted from the employer according to the previous section. Once the eventof creation of a new job offer is launched the two Smart Contracts are created, configured according to the information inserted by the employer, and deployed into the blockchain, linked to each other. The JobOfferManager implements various functions, such as the insertion of the deposit in ETH, the creationof a new job offer, the hiring and the payment. The Employment Smart Contracts contains all the information related to candidates. It also allows to the employer to increase and certify the working hoursas worked by the employee, to end the job once the working hours are completed. The worker can apply

to different job offers, withdraw a candidacy, send a request for a job. The features of the two

Smart Contracts are described by user stories which are reported as a list or according to the diagram shown infig. 6 The user stories are:

- ETH deposit: the employer put a deposit into the contract
- Announce: the employer sets a new job offer with description and features.
- Candidacy: workers send a job application to non expired offers
- Candidacy Withdraw: a worker can withdraw a candidacy
- Visualization: the employer can examine the candidacies
- Hiring: the employer selects a candidate and hires him/her.
- Hour registering request: the employee send a request to the employer to record and certify all workedhours up to that day.
- Confirm: the employer examines the request and can decide to asseverate the worked hours.
- Announcements Visualization: the employer visualizes the job offers he/she created
- Candidacy Visualization: the applicant visualizes the job offers he/she applied to
- Work Visualization: the employee visualizes the job he/she is working on.
- Career: the worker visualizes his/her work history
- Offers Visualization: any user visualizes all job offers (expired or active)
- Expired Offers: any user visualizes the expired job offers

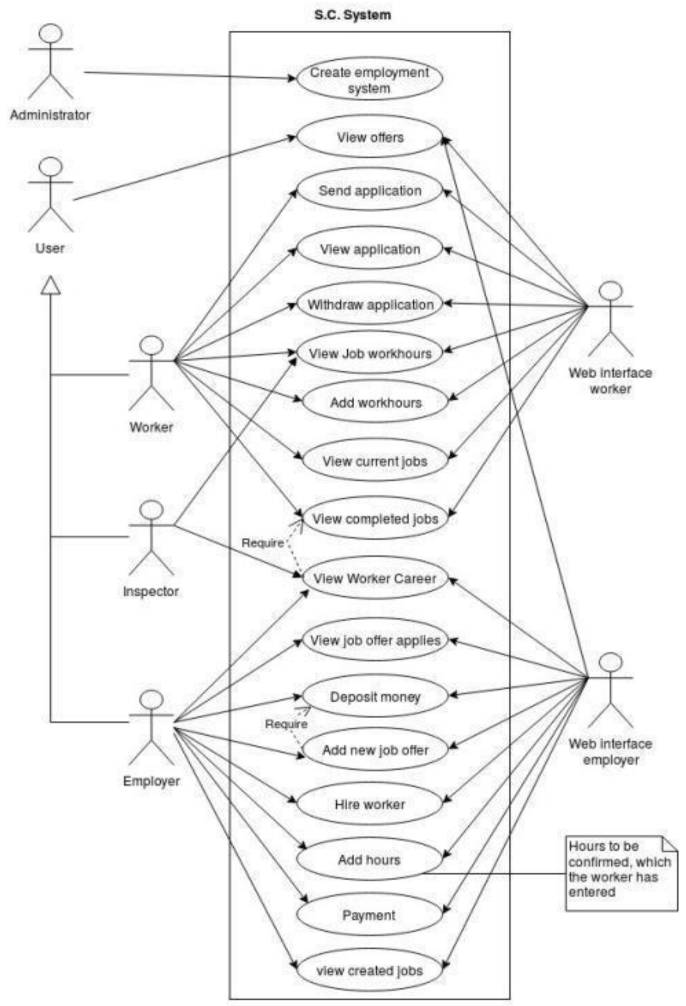


Figure 6: User Stories diagram for the Smart Contract implementation

- Current Offers Visualization: any user visualizes all job offers not yet expired For each user story we implemented a corresponding web page in the prototype. Once the use case diagram for actors has been designed it is easy to proceed to the design of the Smart Contracts. According to what reported above, we implemented into the JobOfferManager Smart Contracts the following state variables: - address owner: address of contract's creator - uint32 lastid: token holding the number of created offers - struct
- jobOffer (data representing one job offer)
- uint256 expirationDate: the expiration day
- address payable worker: Ethereum employee's account address
- address employer: Ethereum employer's account address
- string name: offer's name

- uint8 workhours: working hours to be worked
- uint salary: salary offered - struct Jobs: – uint32[] jobs: array holding all the job offers created
- struct OnGoingJobs – uint32[] onGoingJobs: array holding all the jobs a specific worker is working onat the moment
- mapping(address => uint256) internal depositOf : mapping returning the ETH deposited in the SmartContract by a specific address (associated to an employer)
- mapping(address => Jobs) internal offersBy: mapping returning the job offers created by a specificaddress
- mapping(address => OnGoingJobs) private hiredinjobs: mapping associating the employee address to allthe job he/she is actually working on
- mapping(uint32 => jobOffer) private jobs: mapping returning the description of a job offer given theoffer id
- mapping(uint32 => bool) public moneyIsReturn: mapping returning a boolean stating if the deposit hasbeen withdrawn given an offer id.
- The ABI interface is reported below according to the user stories diagram devised above.
- Constructor
- getNumberOfOffers(): outputs the number of offers created
- getName(uint32): outputs the offer’s name
- getExpirationDate(uint32): outputs the expiration date
- getSalary(uint32): outputs the offer’s salary
- getAddressWorker (uint32): outputs the employee’s address
- getAddressEmployer (uint32): outputs the employer’s address
- getInfo(uint32): outputs the offer’s description
- getAmountHours(uint32): outputs the number of minutes to be worked for a job
- getJobOffer(uint32): outputs all the features of a job offer
- getArrayActiveOffer (): outputs all non expired offers
- getDepositedAmount(): outputs the amount deposited in the contract from a given address
- getOffersBy (address): outputs the offers created by a give address
- getApplicantOf (address): outputs an array containing all the jobs a worker is hired for
- getTokenId(): outputs the token’s actual value
- getBalance(): outputs the amount deposited in wei
- getIsActiveOffer(uint32): verifies if an offer is expired or not
- getIsMoneyIsReturn (uint32): outputs if, given a job offer, the deposit has been withdrawn

- newJob(uint256, string memory, string memory, uint8, uint): creates a new job offer with all the details hireWorker(address payable, uint32): starts the work relationship. It is called by the employer to hire theemployee for a given job offer
- payment(uint32): once the job duties are completed this function is called by the employer to pay the employee.
- moneyReturnsEemployer(uint32): it reimburses the employer once a job offer has expired without any worker hired. Next we report as well the state variables for the Employment Smart Contract:
- address owner: address of contract's creator
- address payable scJobOfferManager: address referring to the Smart Contract managing the job offersstruct Applicant:
- address[] applicant: array containing all candidacies
- uint32[] jobsDone: array containing all the jobs done by a worker
- struct RequestHours (data realted to the hours requested by a worker)
- uint32[] idOffer: array containing all id of job offers without requests of adding hours —
- uint[] numberHours: array containing the number of requested work hours
- mapping (uint32 => uint) internal workhours: mapping providing the number of hours worked by anemployee given the offer id
- mapping (uint32 => Applicant) internal applicantsOf: mapping providing the candidates of an offergiven its id
- mapping (uint32 => uint) internal requestHours: mapping providing the number of hours requested forbeing added given the offer id
- mapping (address => RequestHours) internal requestHoursForEmployer: mapping providing the offerid and the number of requested hours given the employer's address
- mapping (address => JobDone) internal jobsDone: mapping providing the work concluded given theworker's address

Finally we report the ABI Interface for the Smart Contract Employment

- Constructor
- setJobOfferAddress(address payable): the function for setting the address of JobOfferManager SmartContract
- getJobOfferAddress(): outputs the address of JobOfferManager contract
- getIsSetJobOfferAddress(): returns if the address of JobOfferManager contract has been set or not
- getIsEqualToJobOfferAddress(address payable): verifies if the value of variable scJobOfferManager matches the address of the contract provided in input

- single contract managing the job offers)
- getApplicantOf(uint32): outputs the array containing candidates for an offer
- getJobsDone(): outputs the array containing all concluded jobs
- getRequestHours(uint32): outputs the number of hours required for an offer
- getRequestHoursForEmplyer(): outputs two arrays containing the offers' ids and the number of hoursrequested by each offer
- getHoursDone(uint32): outputs the number of hours worked up to that moment by an employee
- getHourMissing(unit32): outputs the number of remaining working hours to be completed
- jobCompleted(uint32): verifies if the work has been executed and performs the payment
- addWorkdays(uint32, uint): it is called by the employer to update the hours worked by an employee andactivates the payment procedure (calling jobCompleted) once if the number of working hours agreed hasbeen reached
- requestAdditionalHours(uint32, uint): is called by the employee to ask for adding the worked hours
- workerApplies(uint32): called by a user to apply for a job offer
- withdrawCandidacy(uint32): called by a user to withdraw from a job offer

Contracts Diagrams

Solidity allows to define Smart Contract in a way very similar to how classes are defined in Object Oriented (OO) programming languages. It supports inheritance, it includes a Constructor to deploy the Smart Contract on the blockchain, it supports interactions among contracts by means of transactions messages, so that a contract can call and activate functions of another

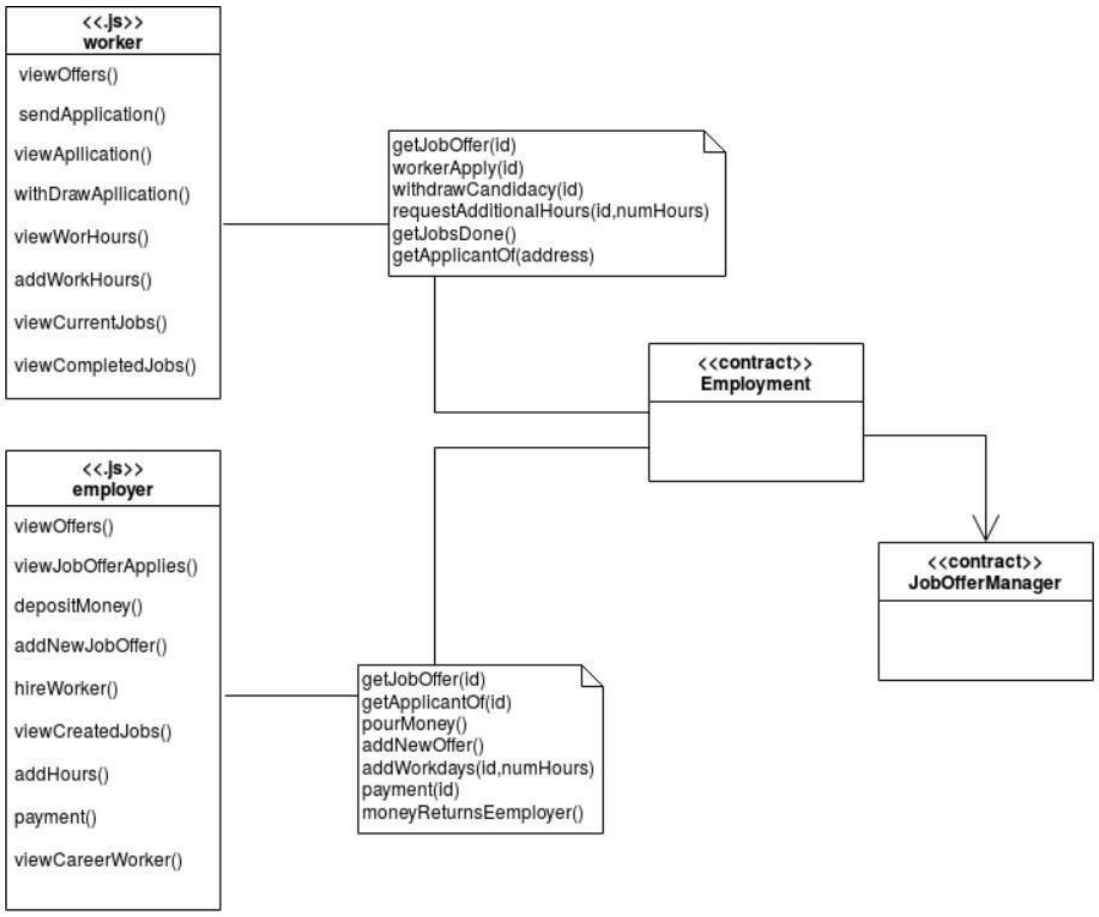
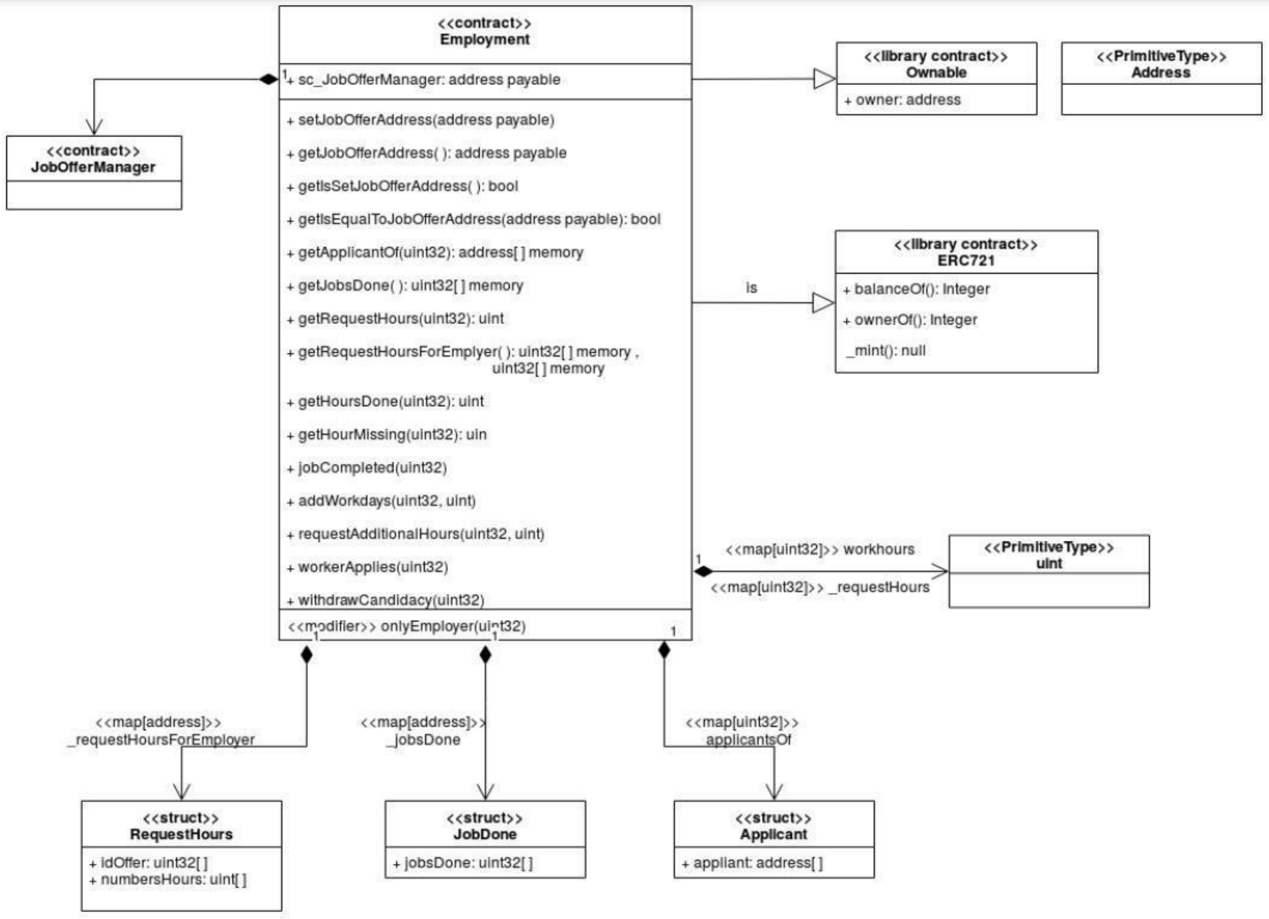


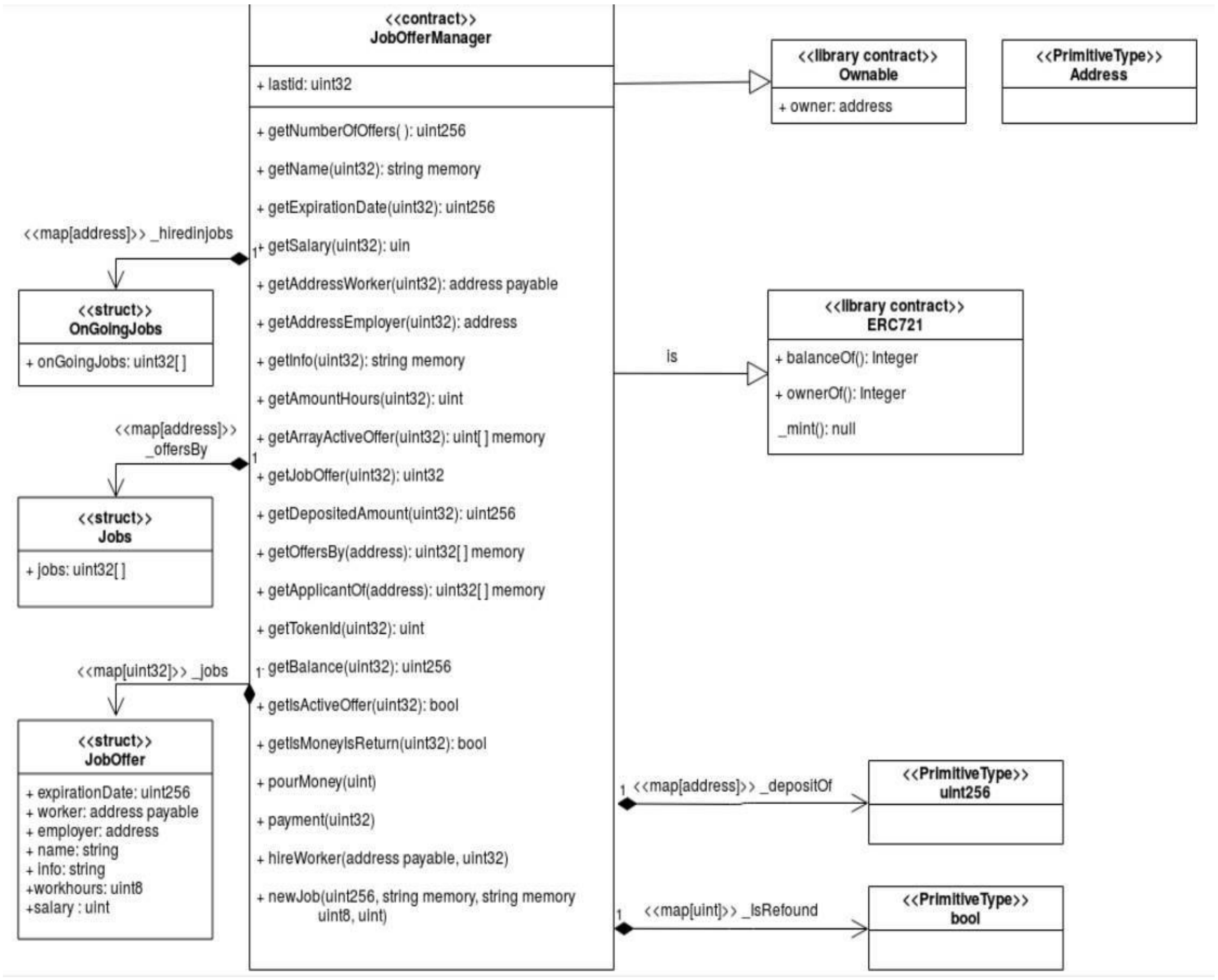
Figure 7: Smart Contract - Class diagram for the DApp implementation

one, it supports events managing. According to BOSE and ABCDE methodology it's convenient to develop diagrams for Smart Contracts design and interaction which adopt the same schemes used in OO Software Engineering such as UML diagrams. We already presented the diagrams for Blockchain SoftwareEngineering for users stories and for state diagrams in the previous section. In this section we describe the devised Smart Contracts and their interaction using the Contracts Diagram. Figures 7, 8 and 9 show the Smart Contracts diagrams and the relationships among them. These are described as in the usual UML diagrams, with similar meanings and with the use of stereotypes to characterize specific aspects of blockchain software. In particular the use of data structures and of libraries is easily described and allows for a clear and fast development of the smart contract software code in Solidity acting as a guide for developers. Fig. 7 also shows how it is possible to represent the interaction among in-chain and out-of- chain components.



<<contract>> Employment
+ sc_JobOfferManager: address payable
+ setJobOfferAddress(address payable)
+ getJobOfferAddress( ): address payable
+ getIsSetJobOfferAddress( ): bool
+ getIsEqualToJobOfferAddress(address payable): bool
+ getApplicantOf(uint32): address[ ] memory
+ getJobsDone( ): uint32[ ] memory
+ getRequestHours(uint32): uint
+ getRequestHoursForEmplyer( ): uint32[ ] memory , uint32[ ] memory
+ getHoursDone(uint32): uint
+ getHourMissing(uint32): uin
+ jobCompleted(uint32)
+ addWorkdays(uint32, uint)
+ requestAdditionalHours(uint32, uint)
+ workerApplies(uint32)
+ withdrawCandidacy(uint32)
<<modifier>> onlyEmployer(uint32)
<<contract>> JobOfferManager
<<library contract>> Ownable
+ owner: address
<<PrimitiveType>> Address
is
<<library contract>> ERC721
+ balanceOf(): Integer
+ ownerOf(): Integer
_mint(): null
<<map[uint32]>> workhours
<<map[uint32]>> _requestHours
<<PrimitiveType>> uint
<<map[address]>> _requestHoursForEmployer
<<map[address]>> _jobsDone
<<map[uint32]>> applicantsOf
<<struct>> RequestHours
+ idOffer: uint32[ ]
+ numbersHours: uint[ ]
<<struct>> JobDone
+ jobsDone: uint32[ ]
<<struct>> Applicant
+ appliant: address[ ]
1

Figure 8: Smart Contract Class diagram for the Employer Smart Contrac



<<contract>>
JobOfferManager
+ lastid: uint32
+ getNumberOfOffers( ): uint256
+ getName(uint32): string memory
+ getExpirationDate(uint32): uint256
+ getSalary(uint32): uin
+ getAddressWorker(uint32): address payable
+ getAddressEmployer(uint32): address
+ getInfo(uint32): string memory
+ getAmountHours(uint32): uint
+ getArrayActiveOffer(uint32): uint[ ] memory
+ getJobOffer(uint32): uint32
+ getDepositedAmount(uint32): uint256
+ getOffersBy(address): uint32[ ] memory
+ getApplicantOf(address): uint32[ ] memory
+ getTokenId(uint32): uint
getBalance(uint32): uint256
+ getIsActiveOffer(uint32): bool
+ getIsMoneyIsReturn(uint32): bool
+ pourMoney(uint)
+ payment(uint32)
+ hireWorker(address payable, uint32)
+ newJob(uint256, string memory, string memory uint8, uint)
<<library contract>>
Ownable
+ owner: address
<<PrimitiveType>>
Address
<<map[address]>> _hiredInjobs
1
<<struct>>
OnGoingJobs
+ onGoingJobs: uint32[ ]
is
<<library contract>>
ERC721
+ balanceOf(): Integer
+ ownerOf(): Integer
_mint(): null
<<map[address]>> _offersBy
1
<<struct>>
Jobs
+ jobs: uint32[ ]
<<map[uint32]>> _jobs
1
<<struct>>
JobOffer
+ expirationDate: uint256
+ worker: address payable
+ employer: address
+ name: string
+ info: string
+workhours: uint8
+salary : uint
1
<<map[address]>> _depositOf
<<PrimitiveType>>
uint256
1
<<map[uint]>> _IsRefound
<<PrimitiveType>>
bool

Figure 9: Smart Contract Class diagram for the JOBOfferManager Smart Contract.

Activity Diagram

In order to define and analyze the dynamic behavior of the system we designed the activity diagrams for the employer and for the employee. This is particularly important in Blockchain Software Engineering since the activity flow correspond to dynamic behaviours on the blockchain as well for the messages flow.In fact activities can match transactions in the blockchain which may occur between smart contracts. On the other hand, for activities that are performed internally to a single smart contract there is the possibilityof generating the events related to these activities (this is typical in solidity). Thus in the design phase theactivity diagrams provide a useful analysis tool to understand which operations need to be executed by means of blockchain transactions and which operations need to be recorded and tracked by means of events. The various activities in the diagram in general correspond to code execution or message

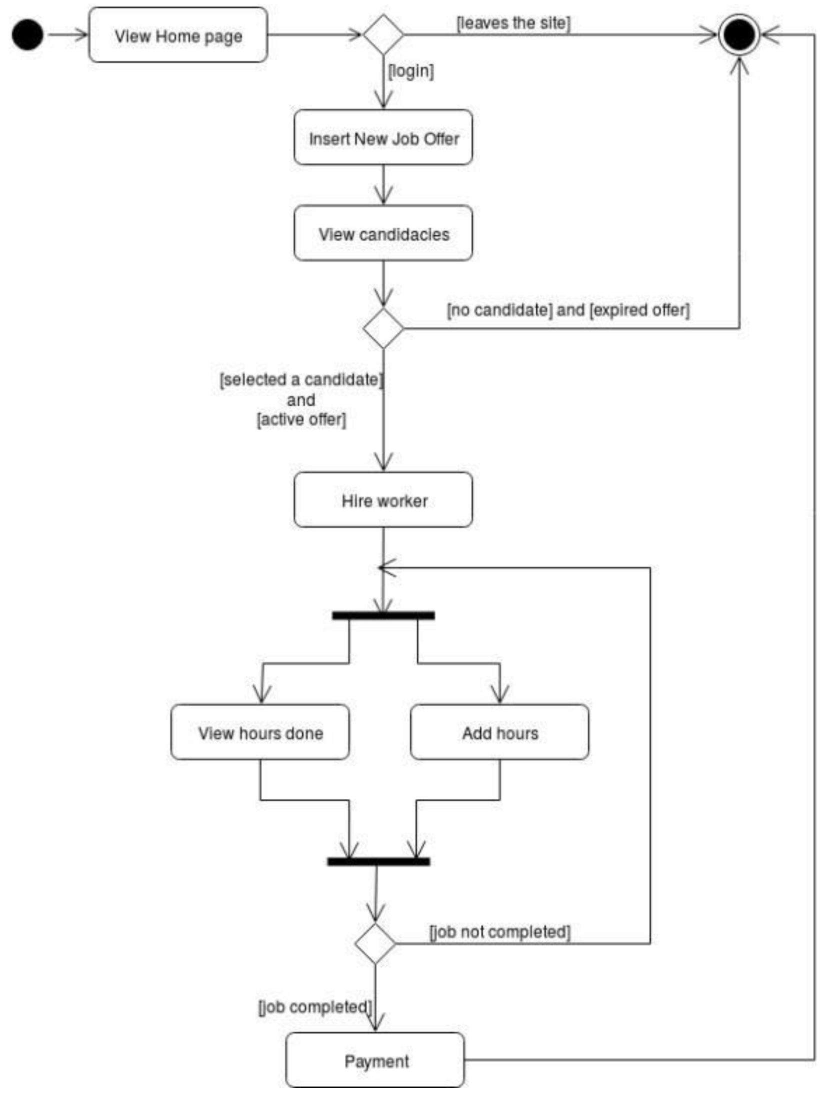


Figure 10: Activity diagram for the Employer - 1

calls. In figures 10 11 we report the activity diagram for the employer and for the employee. The diagramsprovide useful insights for identifying blocks and pieces of working software code for the in and out of blockchain components.

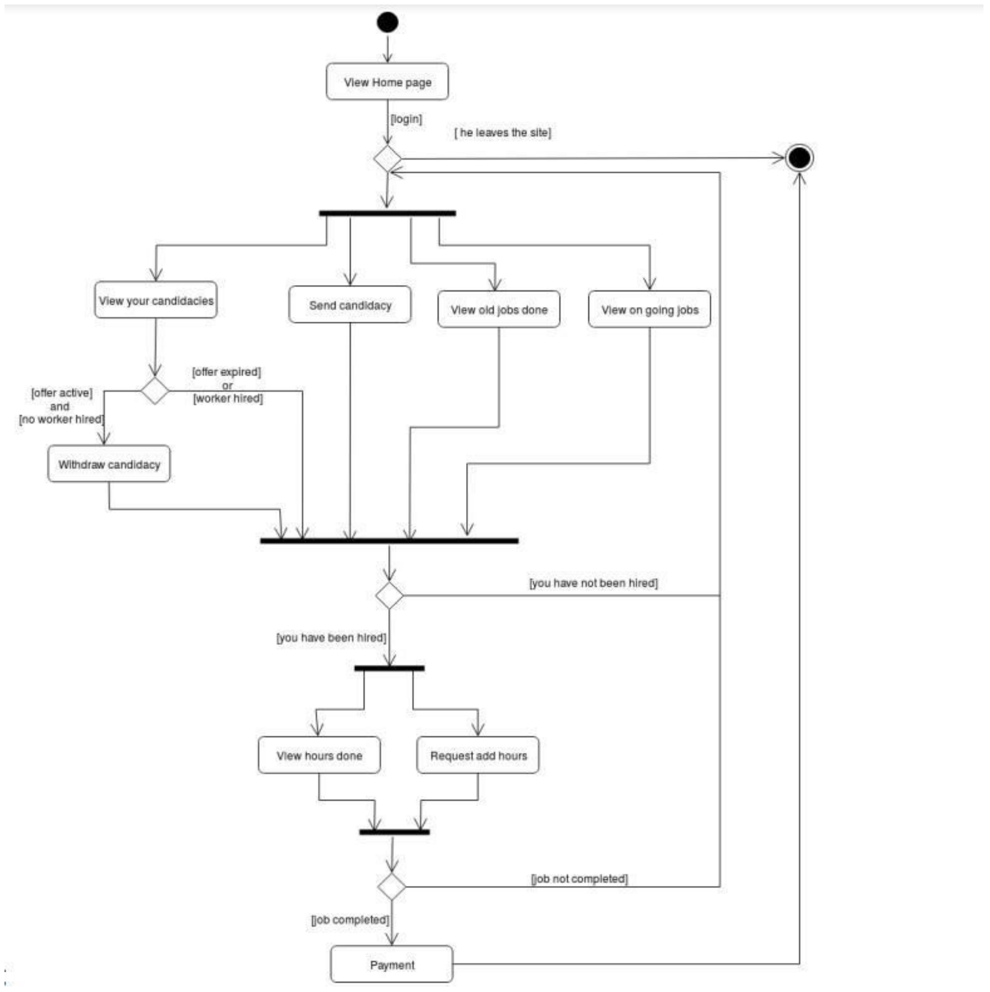


Figure 11: Activity diagram for the Employer - 2

Sequence diagram

In the diagram 12 we represent the sequence of operations that the actors and the system perform to add work hours for a given job until the completion of the agreed number of hours. The worker accesses the workers' web interface and adds work hours to a job. He provides the job id. The web interface collects the request and calls the function "requestAdditionalHours (id, n)" of the employment smart contract which verifies that the worker (the sender of the request) is actually registered as a worker of the given job(with a given id) and the job is active. If so, it adds the N work hours as requested. The employer can checkthe amount of work hours to be confirmed calling the function viewHours() of the employers' web interface. The web interface loops in the job created by the employer and requests the employment smart

contract to return the work hours waiting to be confirmed. Once data are collected, the web interface showsthe list to the employer. The employer can now certify the work hours of a worker, given

A job id .he accesses the employers' web interface and calls the function "selectJobOffer" to request.

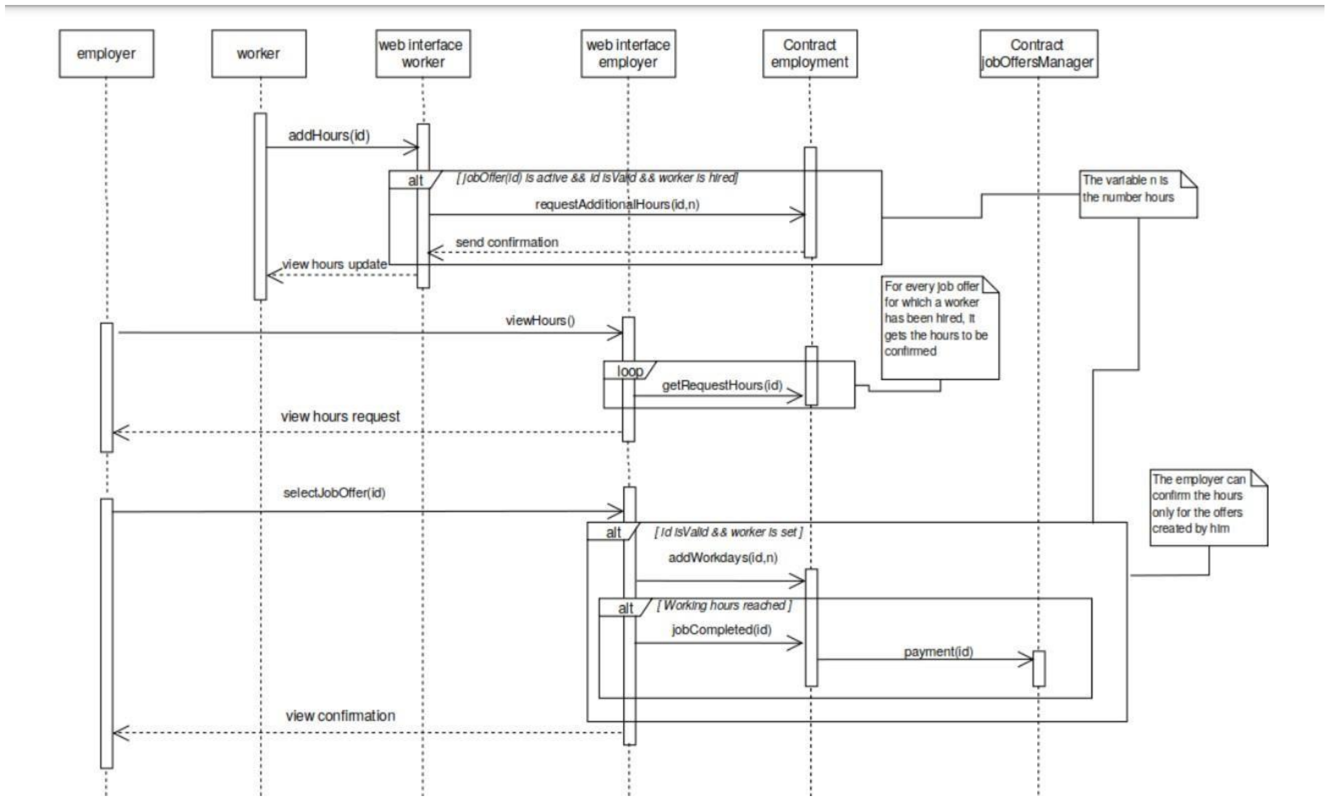

employer
worker
web interface worker
web interface employer
Contract employment
Contract jobOffersManager
addHours(id)
alt
[jobOffer(id) is active && id isValid && worker is hired]
requestAdditionalHours(id,n)
send confirmation
view hours update
The variable n is the number hours
viewHours()
For every job offer for which a worker has been hired, it gets the hours to be confirmed
loop
getRequestHours(id)
view hours request
selectJobOffer(id)
The employer can confirm the hours only for the offers created by him
[Id isValid && worker is set]
addWorkdays(id,n)
[Working hours reached]
jobCompleted(id)
payment(id)
view confirmation

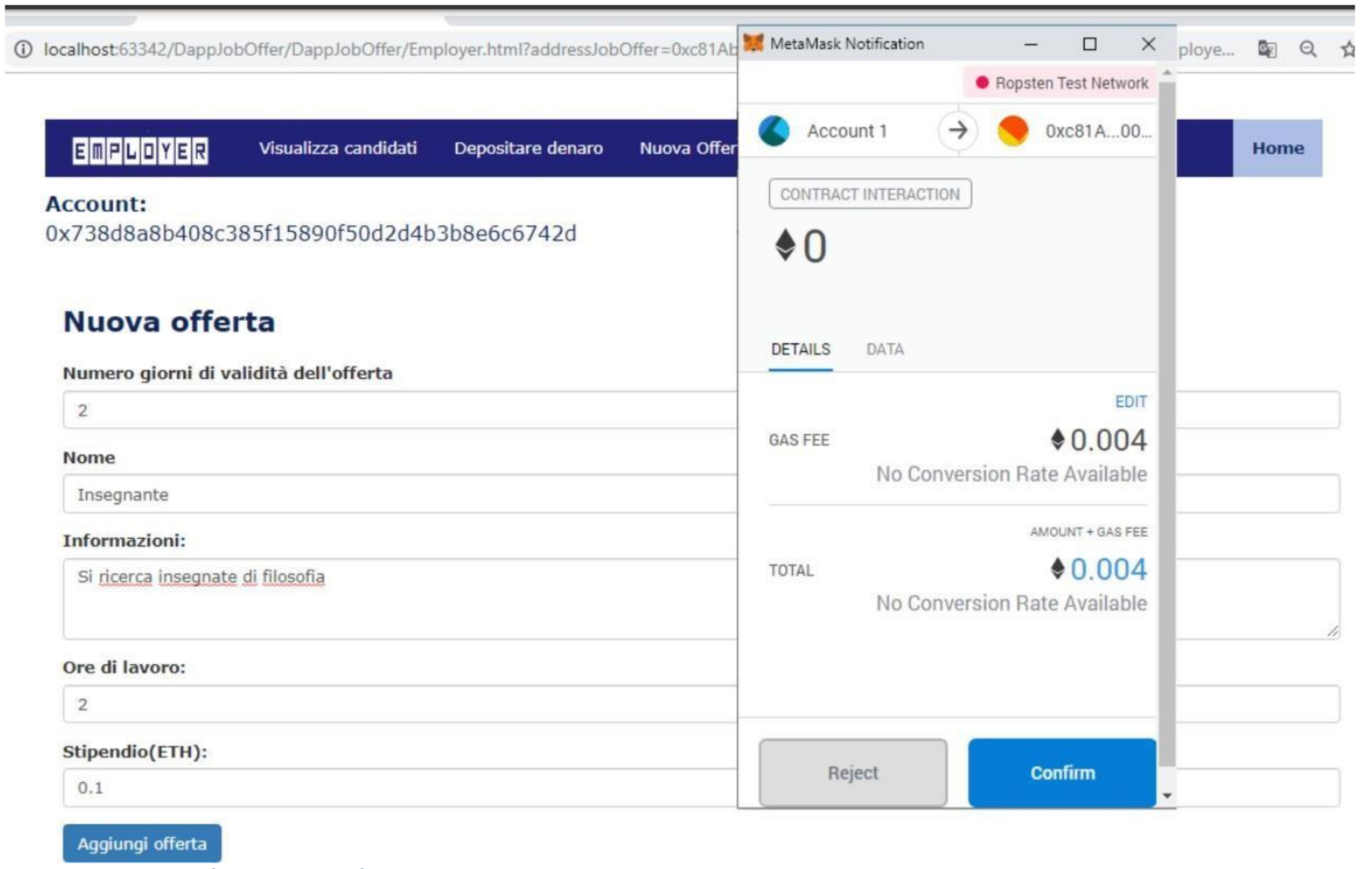
Figure 12: Sequence diagram of the hours registration and of the payment phases

the validation of the requested work hours. The interface calls the function ”addWorkdays” for a given idandan amount N. If the id is valid, the work hours are validated. Now, if the contractual limit of work hours is reached, the system calls the jobCompleted function which runs the ”payment” function of the jobOfferManager contract.

The prototype

Based on the application of all principles and methods adopted from BOSE and ABCDE methodologies we developed the Solidity code for the Smart Contracts (not reported here) and built the DApp system which provides the users with a user friendly web interface enabling the implementation of all the features described. In fig. 13 we report as an example the web interface providing the functionality for the insertion of a new job offer by the employer. The web interface uses "metamask", a bridge to run Ethereum DAppsright in your browser with-out running a full Ethereum node, for providing the communication channel between DApp and blockchain. The web interface allows the various actors of the system to interact with the blockchain through the DApp so that no actor needs to understand or to know the working principles of the blockchain system. Every feature is provided by a user friendly web page were input and output data can be inserted and read and events and function calls can be managed. We devised various interfaces to provide specific features

depending on the users roles with menus adapted to the different roles. We deployed the Smart Contractson the Ropsten test net in order to test our prototype under all working conditions.


localhost:63342/DappJobOffer/DappJobOffer/Employer.html?addressJobOffer=0xc81Ab

EMPLOYER | Visualizza candidati | Depositare denaro | Nuova Offer | Home

Account:
0x738d8a8b408c385f15890f50d2d4b3b8e6c6742d

Nuova offerta

Numero giorni di validità dell'offerta: 2

Nome: Insegnante

Informazioni: Si ricerca insegnate di filosofia

Ore di lavoro: 2

Stipendio(ETH): 0.1

Aggiungi offerta

MetaMask Notification

Ropsten Test Network

Account 1 → 0xc81A...00...

CONTRACT INTERACTION

0

DETAILS | DATA

EDIT

GAS FEE 0.004
No Conversion Rate Available

AMOUNT + GAS FEE
TOTAL 0.004
No Conversion Rate Available

Reject | Confirm

Figure 13: Web interface developed for the interaction with the DApp system.

All these works present two weakness. They focus only on the creation of the contractual relationship and does not include a specific automation in payment when work is completed nor provide additionalprotection to the worker or they do it only marginally.

6.3 WORKING OF SMART CONTRACTS:

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions aremet. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss.

Step 1: Find an open source Solidity contract as a starting point.

Step 2: Define the abstract token contract.

Step 3: Define the abstract store contract.

Step 4: Write test cases for use with TDD.

Step 5: Implement the smart contract code.

Step 6: Create a Custom dev chain testing.

6.4 BLOCK DIAGRAM

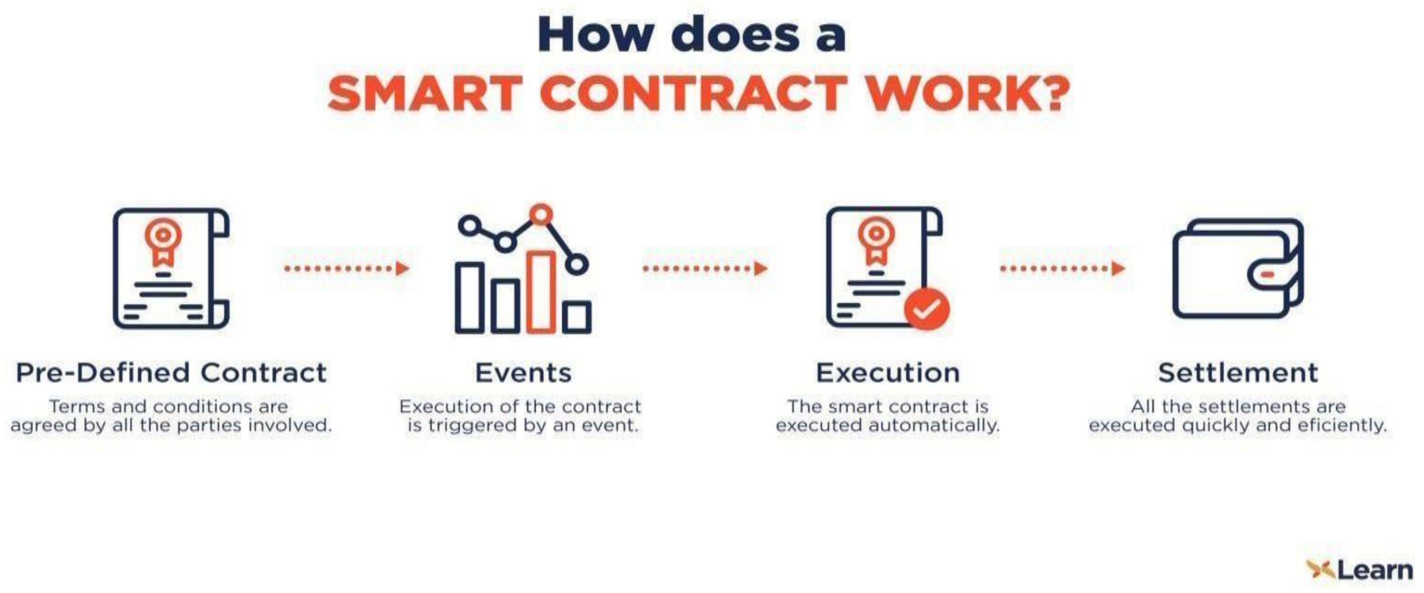

How does a
SMART CONTRACT WORK?
Pre-Defined Contract
Terms and conditions are agreed by all the parties involved.
Events
Execution of the contract is triggered by an event.
Execution
The smart contract is executed automatically.
Settlement
All the settlements are executed quickly and eficiently.
Learn

Figure 14: Working of Smart Contract

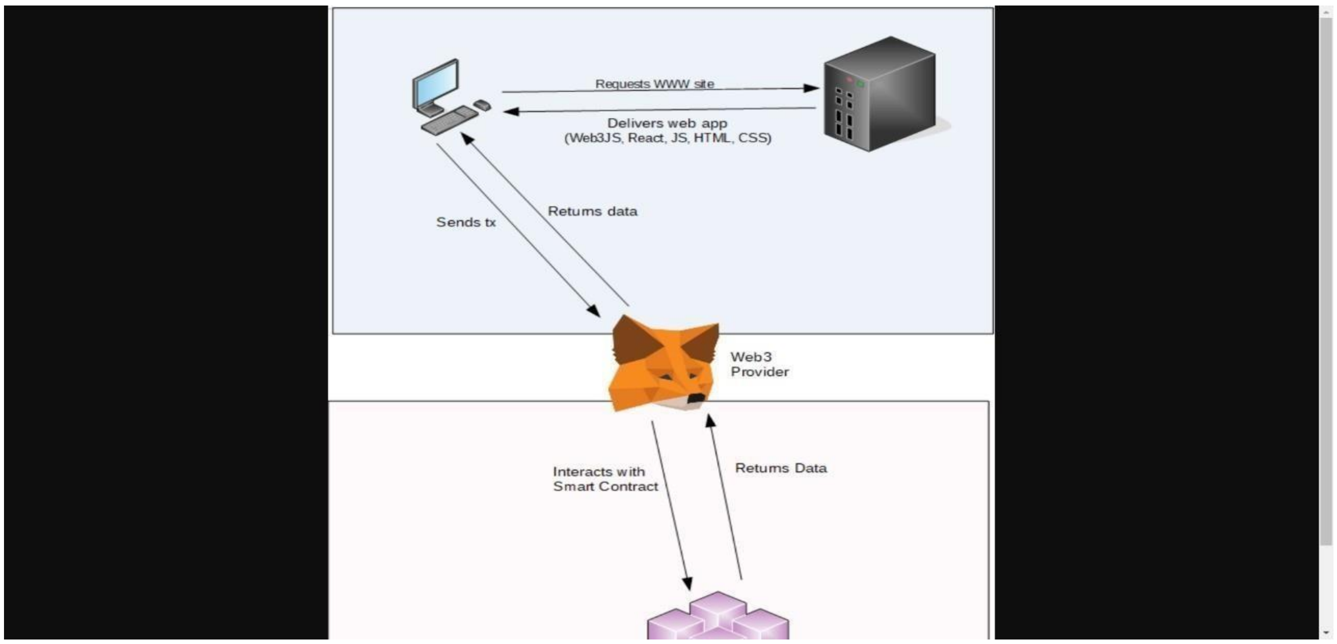

Requests WWW site
Delivers web app (Web3JS, React, JS, HTML, CSS)
Sends tx
Returns data
Web3 Provider
Interacts with Smart Contract
Returns Data

Figure 15: Working of Metamask

6.5 APPLICATIONS:

- Government voting system. Smart contracts provide a secure environment making the votingsystem less susceptible to manipulation.
- Healthcare. Blockchain can store the encoded health records of patients with a private key.
- Supply chain.
- Financial services.

6.6 ADVANTAGES:

- Accuracy. One of the primary requirements of a smart contract is to record all terms andconditions in explicit detail.
- Transparency.
- Clear Communication.
- Speed.
- Security.
- Efficiency.
- Paper Free.
- Storage & Backup.

6.7 FUTURE WORK:

The blockchain technology is still in its nascent stage and it will take time to go mainstream.Before using/embracing the technology, the legal and regulatory aspects in relation to the technology and smart contract needs to be looked into before it is recognized as a valid alternative to traditional contracts.Smart contract platforms are intended to be more self-sufficient, self-governing, accurate and transparent.The benefits of taking one's business into the digital era are wide-reaching and fraud prevention, reduced cost and immutability of the smart contracts are unquestionably huge. Smart contracts can be used in all spheres of lives, from payment gateways to electricity bills etc.This practically guarantees that smart contracts will be a foundation of the future global economy and a part of every person's life.

6.8 BLOCKCHAIN:

Blockchain technology and distributed ledgers are attracting massive attention and trigger multiple projects in different industries. However, the financial industry is seen as a primary user of the blockchain concept. This is not only due to the fact that the most well-known application of this technology is the crypto-currency Bitcoin, but it is also driven by substantial process inefficiencies and a massive cost base issue specifically in this industry. On top of this, the financial crisis revealed that even in financial services it is not always possible to identify the correct present owner of an asset. It is even more of a problem to retrace ownership over a longer chain of changing buyers in global financial transaction services: when, e.g., the US investment bank Bear Stearns failed in 2008 and was completely acquired by JP Morgan Chase, the number of shares offered to the acquirer was larger than the shares outstanding in the books of Bear Stearns. It was not possible to clarify the accounting errors and JP Morgan Chase had to bear the damage from excess (digital) shares.

While the problem of tracing back ownership in long transaction chains is already a critical aspect in financial markets, it is also important for physical goods, e.g., (blood) diamonds or broccoli. US retailerWal-Mart with more than 260 million customers per week is in search for a technology that helps to identify precisely those batches of vegetables that in a given case, e.g., are infected by coliform bacteria. Intermediation is today's dominating solution for verifying ownership of assets and transaction processing. Intermediaries perform the careful checking of each involved party along a chain of intermediaries. However, this is not only time consuming and costly but an intermediary fails. The blockchain technology promises to overcome these critical aspects, representing ‘‘a shift from trusting people to trusting math’’ (Antonopoulos 2014) since human interventions are no longer necessary. In the past few years, the crowdfunding business has become one of the major ways of raising money. It typically involves raising money from a large number of people contributing relatively small amounts (who expect a financial or non-financial Campaigns can be created by companies, charities, and individuals for a specific goal. It has become a popular alternative to venture capital and allows non-traditional projects to pitch their ideas and raise funds for startups.

Another development of recent times is Blockchain technology which has quite a several applications, now including crowdfunding. A Blockchain is a decentralized distributed ledger that makes a record of data in a way that is transparent and difficult to change. It has the advantage of privacy; the level of security it provides makes it highly resistant to malicious activities.

6.9 APPLICATIONS OF BLOCKCHAIN:

Type	Applications	Description	Examples
Financial Applications	Crypto – currencies	Networks and medium of exchange using cryptography to secure transactions	Bitcoin Litecoin Ripple Monero NASDAQ private equity Medici Blockstream Coinsetter Everledger
	Securities issuance, trading and settlement	Companies going public issue shares directly and without a bank syndicate. Private, less liquid shares can be traded in a blockchain-based secondary market. First projects try to tackle securities settlement	
	Insurance	Properties (e.g., real estate, automobiles, etc.) might be registered using the blockchain technology. Insurers can check	

		thetransaction history Central authorization by notary is not necessary anymore.	
	Notary public	Central authorization by notary is not necessary anymore	
Non-Financial applications	Music industry	Determining music royalties and managing music rights ownership.	Stampery Viacoin Ascribe Imogen heap Storj Filament ADEPT Namecoin
	Decentralized proof of existence of documents	Storing and validating the signature and timestamp of a document using blockchain.	
	Decentralized storage	Sharing documents without the need of a third party by using a peer-to-peer distributed cloud storage platform.	
	Decentralized internet ofthings	The blockchain reliably stores the communication of smart devices within the	

		internet of things.	
	Anti-counterfeit solutions	Authenticity of products is verified by the blockchain network consisting of all market participants in electronic commerce (producers, merchants, marketplaces).	
	Internet applications	Instead of governments and corporations, Domain Name Servers (DNS) are controlledby every user in a decentralized way.	

Table 2: Applications of Blockchain

Crosby et al. (2016) distinguish between financial and nonfinancial applications that could potentially be addressed by the blockchain (Table 1). This disruptive innovation has not only the potential to change thenature of interactions in Finance, but also in many other areas of our everyday life. For instance, the Britishsinger Imogen Heap sells her songs using the blockchain.

The application fields for blockchains seem to be manifold, especially in areas that have historically reliedon third parties to establish a certain amount of trust. Atzori (2015) suggests that politics and the entire society might be restructured by the blockchain. Many functions might become obsolete if people startedto organize and protect the society using decentralized platforms. He concludes that ‘‘decentralization of government services through permissioned blockchains is possible and desirable, since it can significantly increase public administration functionality’’. Reorganizing societies is of prime importance in poor countries. Wealth can be protected more effectively using the blockchain. Especially in the third world, landowners have problems to prove the ownership if for example the local government aims to expropriatethe population. These existential threats can be controlled by integrating land titles into the blockchain. However, as pointed out by Glaser (2017), the interface between the digital realm and the physical worldcould turn out to be the weak link which damages the digital trust established by a blockchain system.

There is also currently a debate among researchers and regulators if crypto-currencies relying on the blockchain can fulfill the functions of real money (European Central Bank 2012; Federal Bureau of Investigation 2012). Money has been defined by Mishkin (2004) as ‘‘anything that is generally accepted in payment for goods or services or in the repayment of debts’’. Luther and White (2014) argue that today crypto-currencies are only rarely used as a medium of exchange. Glaser et al. (2014) provide empirical insights that Bitcoin is indeed primarily used as a speculative asset. However, spending and accepting might become easier due to innovative approaches by entrepreneurs, establishing crypto-currencies as a substitute for fiat money. The blockchain might therefore contribute to change the way people pay for goods in the real world. Homeowners face significant transaction costs when buying property. According to Goldman Sachs, ‘‘blockchain could reduce title insurance premiums and generate \$2– \$4 billion in cost savings in the US by reducing errors and manual effort’’ (Goldman Sachs 2016).

While computer scientists mainly focus on the technical and cryptographic challenges in this area, researchers from the Business and Information Systems Engineering field have the opportunity to focus on market design, questions of trust and privacy, and the adoption respective non-adoption of the new technology. Moreover, this disruptive innovation might change many existing business models, create newones and might have severe impacts on entire industries. Therefore, research at the intersection of technology, markets and business models is certainly valuable.

6.10 BLOCKCHAIN AND SMART CONTRACTS:

The rise of the blockchain technology in recent years also supports other concepts that have been suggestedin literature. Szabo (1997) introduced the concept of ‘‘Smart Contracts’’, which combine computer protocols with user interfaces to execute the terms of a contract. Due to the blockchain, Smart Contracts are becoming more popular since they can be utilized more easily by applying blockchains in comparisonto the technology available at the time of their invention 20 years ago. This innovative approach might, for example, replace lawyers and banks that have been involved in contracts for asset deals depending onpredefined aspects (Fairfield 2014). Smart Contracts can also be used to control the ownership of properties. These properties might be tangible (e.g., houses, automobiles) or intangible (e.g., shares, access rights). A prominent example for blockchain technology that treats smart contracts as first class citizens is Ethereum, which is a decentralized system originally proposed by Buterin (2014). A taxonomy of decentralised consensus systems and an overview of different types of systems is provided.

By Glaser and Bezzenberger (2015). Ethereum can be seen as an extension of the Bitcoin blockchain to support a broader scope of applications. Thus, blockchain technology allows to establish contracts using cryptography and to replace third parties (e.g., a notary) that have been necessary to establish trust in the past. Blockchain might disrupt the entire transaction process by automatically executing contracts in a cost-effective, transparent and secure manner (Fairfield 2014). The architectural components of blockchain technology, their interaction as well as a framework for implication analysis of blockchain systems for digital ecosystems is proposed by Glaser (2017).

The financial industry is even wondering if large parts of their current business might be replaced by the blockchain. This can be illustrated by the payment process. If people pay goods by credit card today, the settlement occurs after a delay of several days. Utilizing the blockchain, this delayed settlement would become redundant since payment can be done in real time by adjusting the ledger.

CHAPTER

CONCLUSION AND FUTURE SCOPE

CONCLUSION AND FUTURE WORKS

7.1 CONCLUSION:

Based on the literature survey we came up with an idea to use smartcontracts in crowdfunding transactions instead of traditional way. By using smartcontracts transactions would be autonomous, increases transparency, security, speed, efficiency while keeping the transaction cost low.The benefits tobe included in the existed solutions should be full control of the money to the donor, less money to the middleman.A smartcontracts aims to provide additional security and trust to backers.As per the contractthe campaign funds are released or even withdrawal, the decision based on approval of more than 50% of the contributors.

7.2 FUTURE SCOPE

Smart contract technology is reshaping conventional industry and business processes. Being embedded in blockchains, smart contracts enable the contractual terms of an agreement to be enforced automatically without the intervention of third party. Smart contracts can be used in several other fields , ex: Land registration.

7.3 LIMITATIONS

1. Difficult to change:

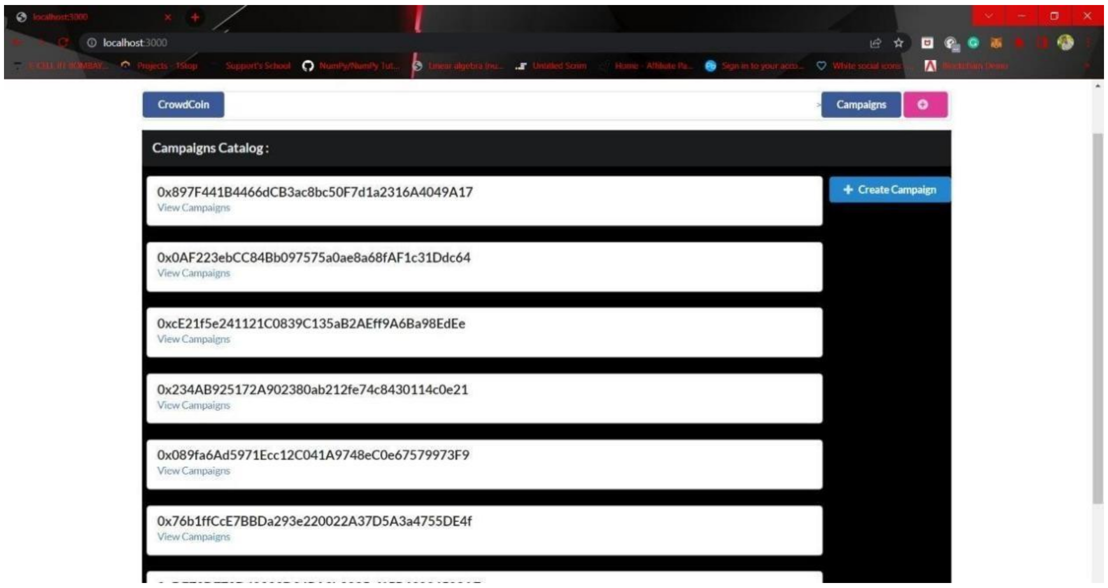
   Changing smart contract processes is almost impossible, any error in the code can be time-consuming and expensive to correct.

2. Possibility of loopholes:

   According to the concept of good faith, parties will deal fairly and not get benefits unethically from a contract. However, using smart contracts makes it difficult to ensure that the terms are met according towhat was agreed upon.

CHAPTER-8
RESULTS

RESULTS



Campaigns Catalog :

0x897F441B4466dCB3ac8bc50F7d1a2316A4049A17
View Campaigns

0x0AF223ebCC84Bb097575a0ae8a68fAF1c31Ddc64
View Campaigns

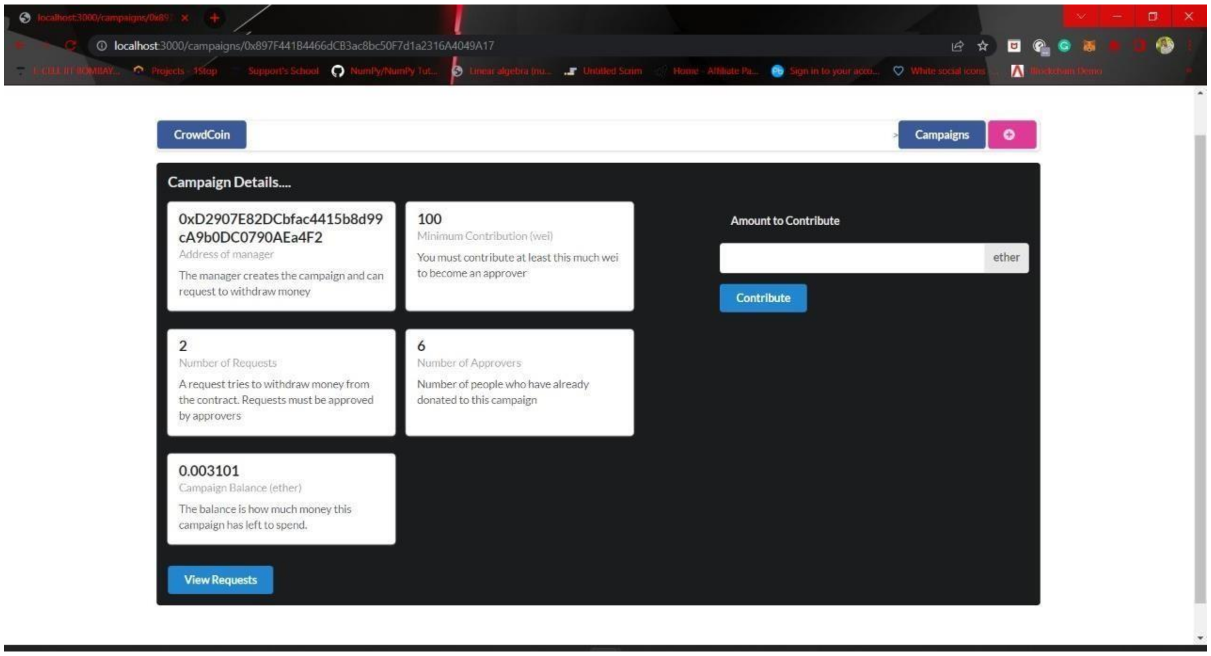
0xcE21f5e241121C0839C135aB2AEff9A6Ba98EdEe
View Campaigns

0x234AB925172A902380ab212fe74c8430114c0e21
View Campaigns

0x089fa6Ad5971Ecc12C041A9748eC0e67579973F9
View Campaigns

0x76b1ffCcE7BBDa293e220022A37D5A3a4755DE4f
View Campaigns

Figure 16: Screenshot of Campaign Catalog


localhost:3000/campaigns/0x897F441B4466dCB3ac8bc50F7d1a2316A4049A17

Campaign Details....

0xD2907E82DCbfac4415b8d99cA9b0DC0790AEa4F2
Address of manager
The manager creates the campaign and can request to withdraw money

100
Minimum Contribution (wei)
You must contribute at least this much wei to become an approver

2
Number of Requests
A request tries to withdraw money from the contract. Requests must be approved by approvers

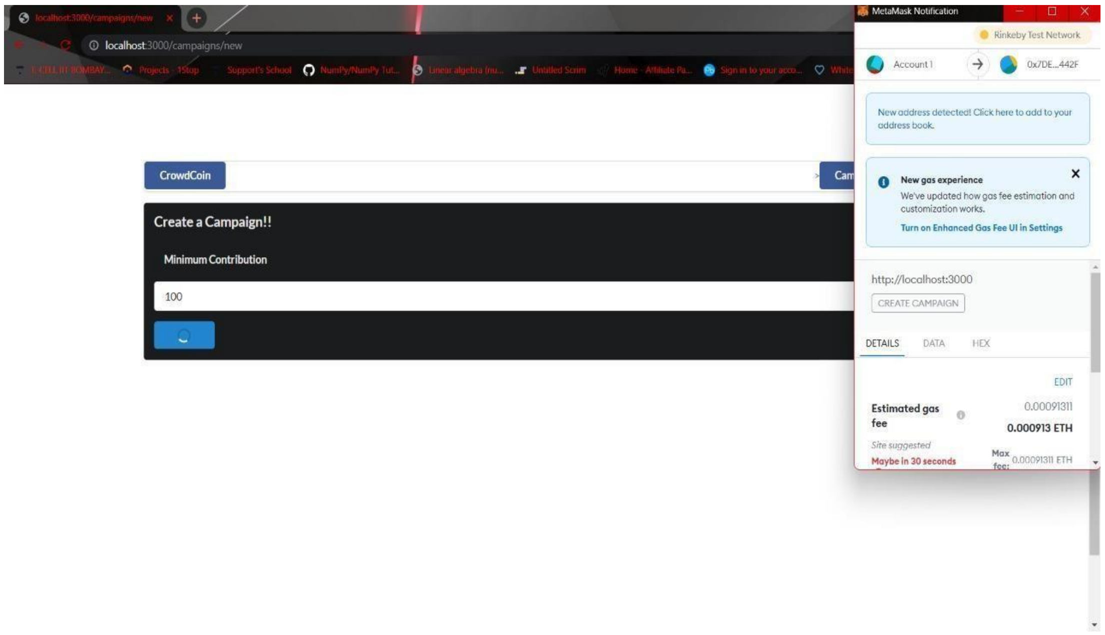
6
Number of Approvers
Number of people who have already donated to this campaign

0.003101
Campaign Balance (ether)
The balance is how much money this campaign has left to spend.

Amount to Contribute

ether

Figure 17: Screenshot of Campaign Details



Create a Campaign!!

Minimum Contribution

100

Rinkeby Test Network

Account 1 → 0x7DE...442F

New address detected! Click here to add to your address book.

New gas experience

We've updated how gas fee estimation and customization works.

Turn on Enhanced Gas Fee UI in Settings

http://localhost:3000

CREATE CAMPAIGN

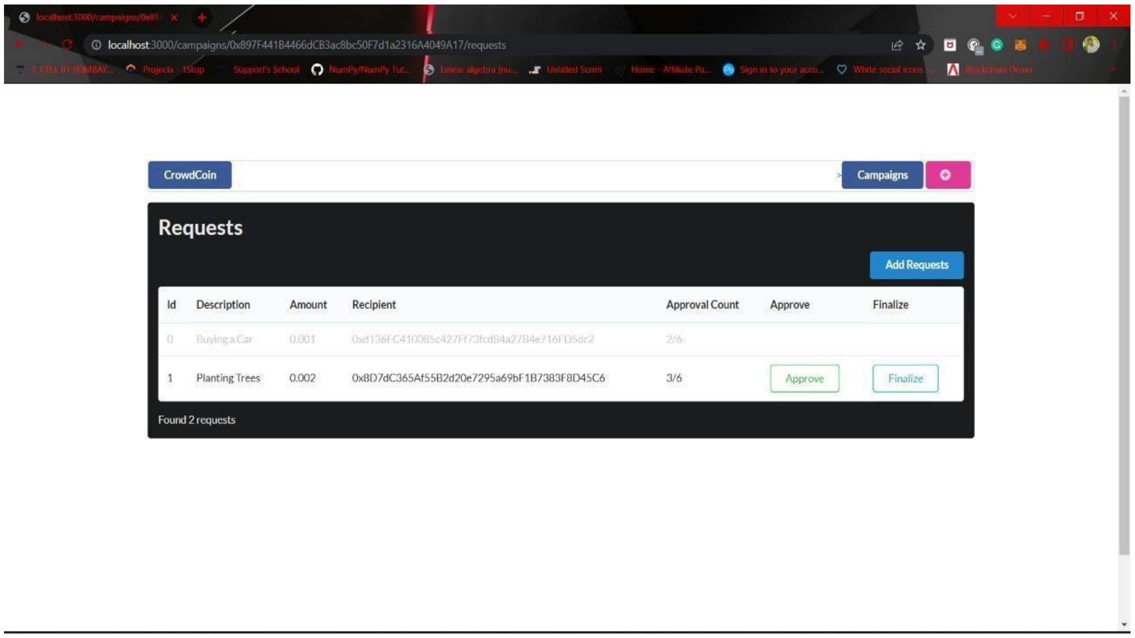
Estimated gas fee 0.00091311 0.000913 ETH

Site suggested

Maybe in 30 seconds

Max fee: 0.00091311 ETH

Figure 18: Screenshot While Contributing


localhost:3000/campaigns/0x897F441B4466dCB3ac8bc50F7d1a2316A4049A17/requests
CrowdCoin
Campaigns
Requests
Add Requests

Id	Description	Amount	Recipient	Approval Count	Approve	Finalize
0	Buying a Car	0.001	0xd136FC410085c427Ff73fcd84a27B4e716FD5dc2	2/6		
1	Planting Trees	0.002	0x8D7dC365Af55B2d20e7295a69bF1B7383F8D45C6	3/6	Approve	Finalize

Found 2 requests

Figure 19: Screenshot of Adding Requests

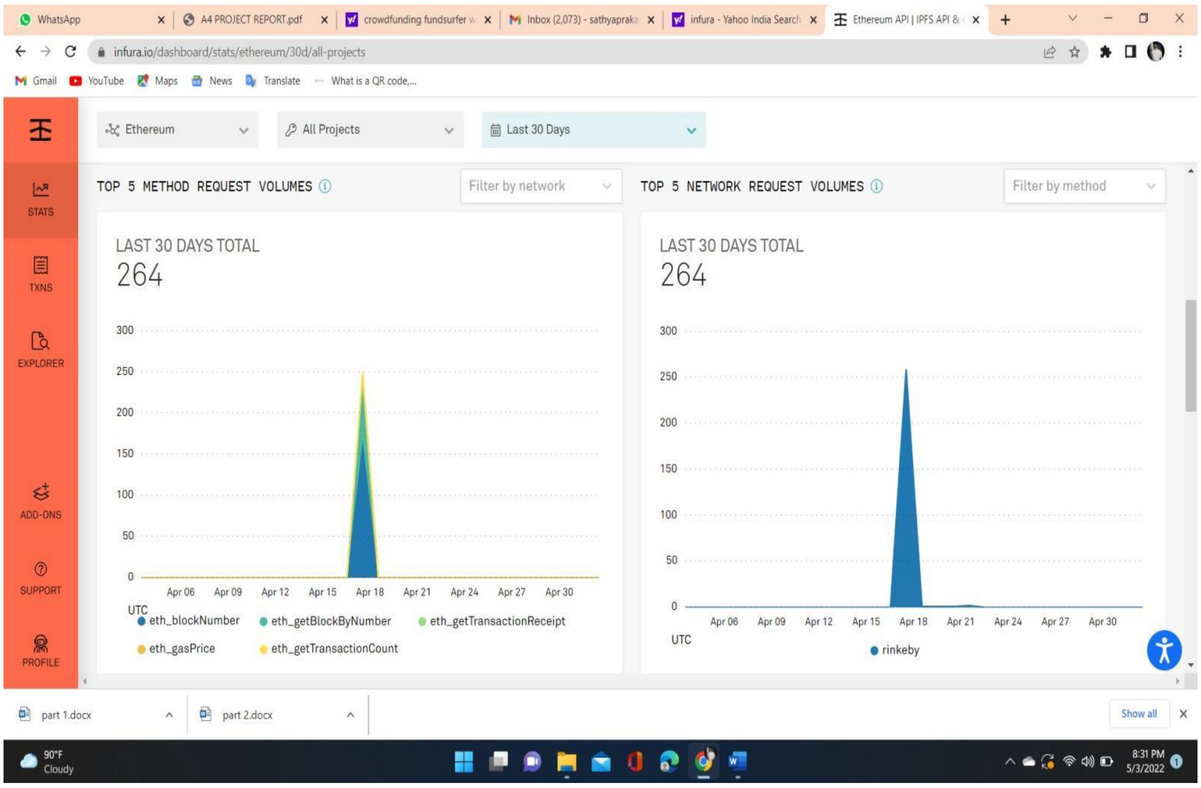
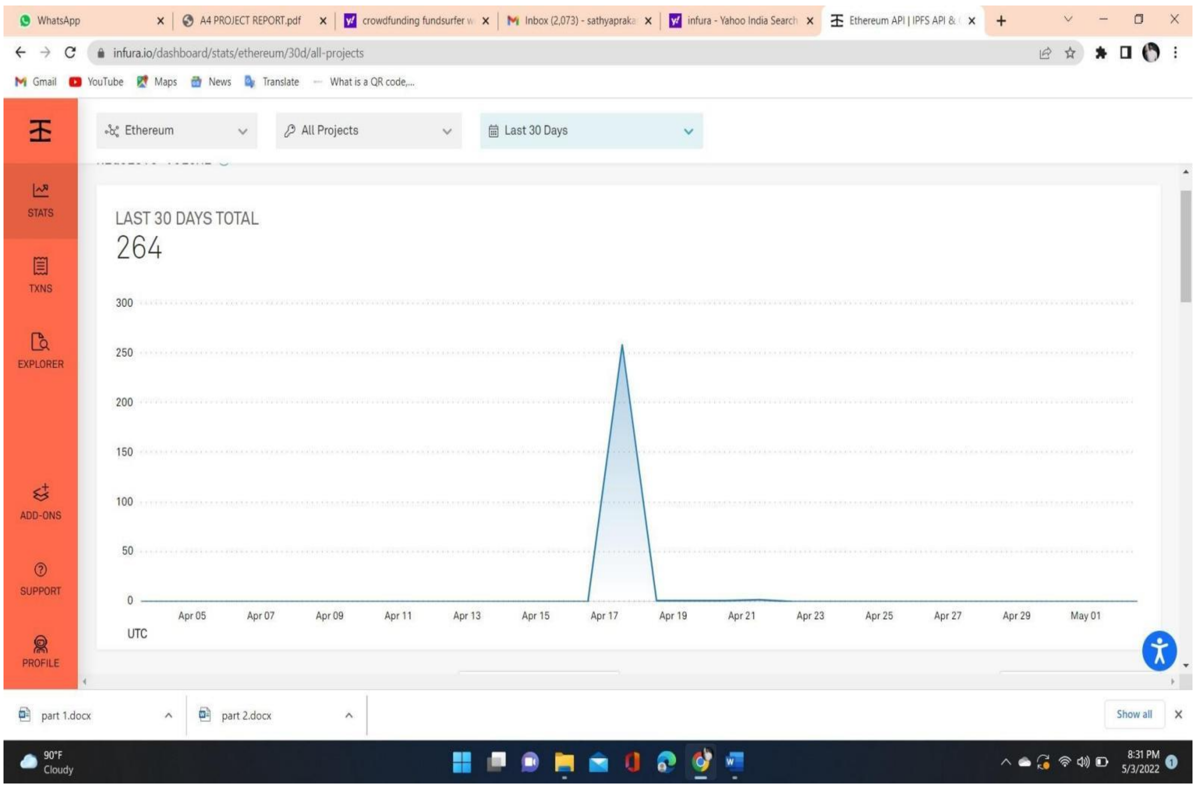
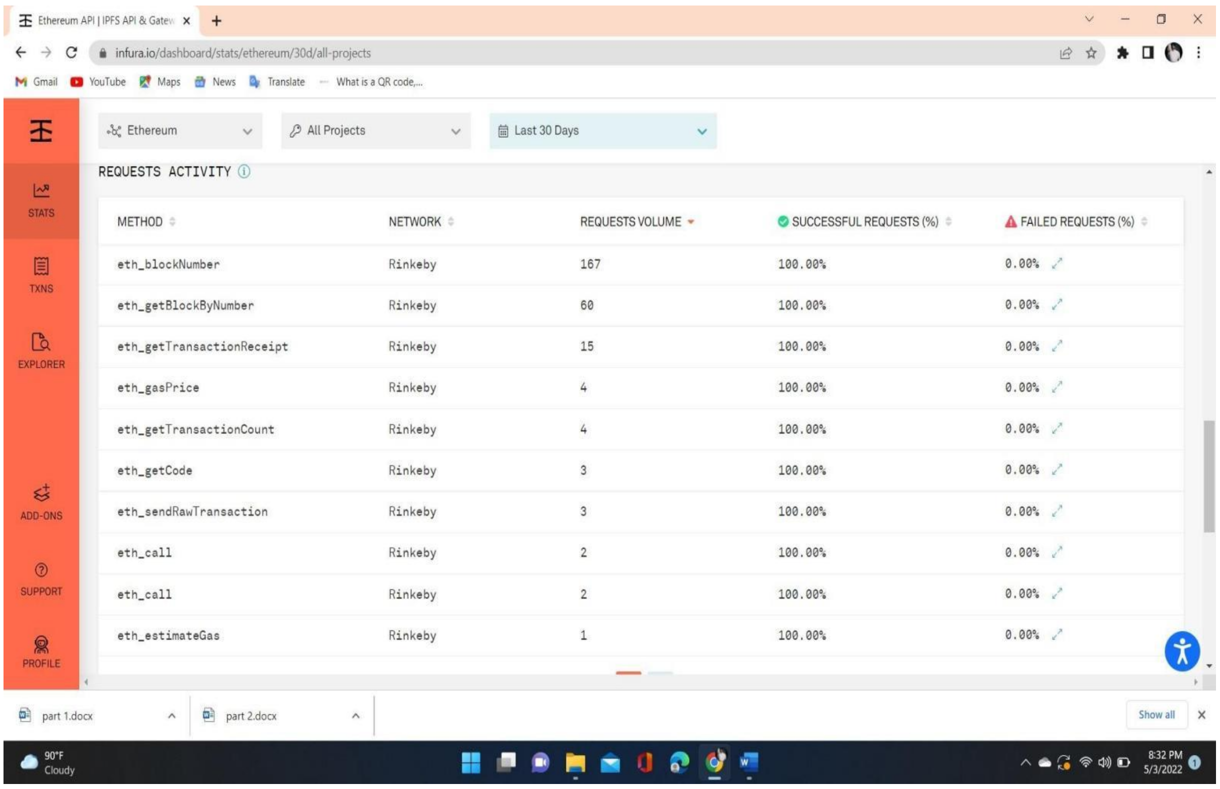


Figure 20: Statistics of Transactions - 1



LAST 30 DAYS TOTAL
264

Figure 21: Statistics of Transactions - 2



Ethereum | All Projects | Last 30 Days

REQUESTS ACTIVITY

METHOD	NETWORK	REQUESTS VOLUME	SUCCESSFUL REQUESTS (%)	FAILED REQUESTS (%)
eth_blockNumber	Rinkeby	167	100.00%	0.00%
eth_getBlockByNumber	Rinkeby	60	100.00%	0.00%
eth_getTransactionReceipt	Rinkeby	15	100.00%	0.00%
eth_gasPrice	Rinkeby	4	100.00%	0.00%
eth_getTransactionCount	Rinkeby	4	100.00%	0.00%
eth_getCode	Rinkeby	3	100.00%	0.00%
eth_sendRawTransaction	Rinkeby	3	100.00%	0.00%
eth_call	Rinkeby	2	100.00%	0.00%
eth_call	Rinkeby	2	100.00%	0.00%
eth_estimateGas	Rinkeby	1	100.00%	0.00%

Figure 22: Requests Activity

CHAPTER 9

SOURCE CODE

SOURCE CODE

```
    pragma solidity ^0.4.17;

    contract CampaignFactory {
    address[] public deployedCampaigns;

    function createCampaign(uint minimum) public {
    address newCampaign = new Campaign(minimum, msg.sender);
    deployedCampaigns.push(newCampaign);
    }

function getDeployedCampaigns() public view returns (address[]) {return
    deployedCampaigns;
    }
    }

contract Campaign {struct Request {
    string description;uint value; address recipient;bool complete;
    uint  approvalCount; mapping(address => bool) approvals;
    }

    Request[] public requests;address public manager;
    uint public minimumContribution;
    mapping(address => bool) public approvers;uint public approversCount;

modifier restricted() { require(msg.sender == manager);
    _;
    }

function Campaign(uint minimum, address creator) public {manager = creator;
    minimumContribution = minimum;
    }

function contribute() public payable { require(msg.value > minimumContribution);

    approvers[msg.sender] = true;approversCount++;
    }

function createRequest(string description, uint value, address recipient) public restricted {
    Request memory newRequest = Request({
    description: description,value: value,
    recipient: recipient,complete: false, approvalCount: 0
    });

    requests.push(newRequest);
    }

function approveRequest(uint index) public {Request storage request = requests[index]


                        require(approvers[msg.sender]);
                         require(!request.approvals[msg.sender]);

    request.approvals[msg.sender] = true;request.approvalCount++;
    }

function finalizeRequest(uint index) public restricted {Request storage request =
    requests[index];
    require(request.approvalCount > (approversCount / 2));require(!request.complete);


    request.recipient.transfer(request.value);request.complete = true;
    }

  function getSummary() public view returns (uint, uint, uint, uint, address
    ) {
  return ( minimumContribution,this.balance, requests.length, approversCount, manager
    );
    }

function getRequestsCount() public view returns (uint) {return requests.length;
    }
```

REFERENCES

[1] Shivansh Pandey, 2018, “Crowdfunding Fraud Prevention using Blockchain”, JSS Academy of Technical Education,Noida, INDIA

[2] Hasnan Baber, 2019, “Blockchain-Based Crowdfunding: A ‘Pay-it-Forward’ Model of WHIRL,” International Journal of Recent Technology and Engineering (IJRTE).

3] Nik Ahmad, N.A and Syed Zamri, S.A (2014). The cross platform application development adapted Spring framework to support front-end tendering services. 2014 International Conference on Computer, Communications, and Control Technology (I4CT), Langkawi, 2014, pp. 58-62.

[4] Ahmad, N. A. N., and Kasirun, Z.M. (2011), ‘Elicitation strategies for Web application using activity theory. Journal of Advances in Computer Research. Volume 2, Number 3 (5), pp. 1-13 [online]. Available: https://www.sid.ir/en/Journal/ ViewPaper.aspx?ID=260575

[5] Sultan, K., Ruhi, U., and Lakhani, R. (2018). Conceptualizing Blockchains: Characteristics & Applications, 11th IADIS International Conference Information Systems 2018, ISBN: 978-989-8533-74-6, pp 49-57.

[6] Yaga, D., Mell, P., Roby, N. and Scarfone, K. (2018). Blockchain Technology Overview. National Institute of Standards and Techology, U.S Department of Commerce, NISTIR 8202, doi: 10.6028/NIST.IR.8202

[7] Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. 2017 IEEE International Congress on Big Data (BigData Congress), 557-564.

[8] Puthal, D., Malik, N., Mohanty, S., Kougianos, E. and Das, G. (2018). Everything You Wanted to Know About the Blockchain: Its Promise, Components, Processes, and Problems. IEEE Consumer Electronics Magazine, 7(4), pp.6-14.

[9] Mohanta, B. K., Panda, S. S., & Jena, D. (2018). An Overview of Smart Contract and Use Cases in Blockchain Technology. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-4.

[10] Mik, Eliza. (2017). Smart contracts: terminology, technical limitations and real world complexity. Law, Innovation and Technology. 9. 1-32. 10.1080/17579961.2017.1378468.

[11] Alharby, M., & Moorsel, A.V. (2017). Blockchain-based Smart Contracts: A Systematic Mapping Study. CoRR, abs/1710.06372.

[12] Delmolino, K., Arnett, M., Kosba, A. E., Miller, A. & Shi, E. (2015). Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab.. IACR Cryptology ePrint Archive, 2015, 460.

[13] Parizi, R.M., Amritraj, & Dehghantanha, A. (2018). Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security.

[14] Buterin, V. (n.d). A Next Generation Smart Contract & Decentralized Application. Ethereum White Paper, ethereum.org, pp. 1-36.

[15] M. Freedman, D. and R. Nutting, M. (2015). A Brief History of Crowdfunding Including Rewards, Donation, Debt and Equity Platforms in the USA.

[16] Gabison, G. A. (2015). Understanding Crowdfunding and its Regulation: How Can Crowdfunding Help ICT Innovation? JRC Science and Policy Report. European Commission's Joint Research Centre (JRC), Report EUR 26992 EN.

[17] Macht, Stephanie & Weatherston, Jamie. (2014). The Benefits of Online Crowdfunding for Fund-Seeking Business Ventures. Strategic Change. 23. 10.1002/jsc.1955.

[18] Mollick, Ethan R., The Dynamics of Crowdfunding: An Exploratory Study (2013). Journal of Business Venturing, Volume 29, Issue 1, January 2014, pp. 1–16.

APPENDIX 1

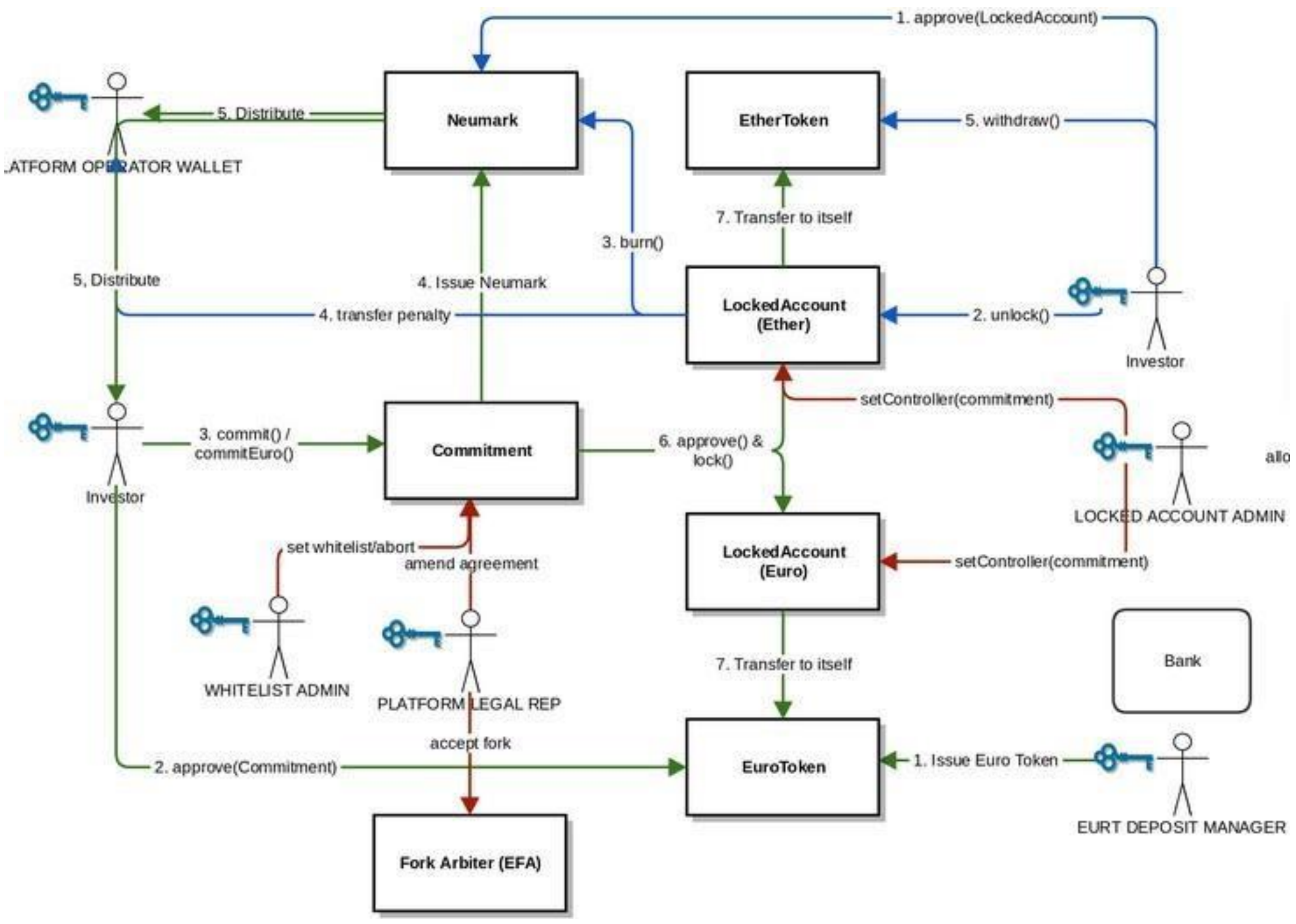


Figure 23: Flow Chart of Smart Contract