GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

# DONATION SYSTEM USING BLOCKCHAIN

**A Report for the Evaluation 3 of Project 2**

*Submitted by*

**GAUTAM UPADHYAYA**

(1613105046)

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION OF CLOUD COMPUTING AND VIRTUALIZATION**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**Under the Supervision of**
**Dr. SPS CHAUHAN, Associate Professor**

**APRIL / MAY- 2020**

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

## BONAFIDE CERTIFICATE

Certified that this project report "DONATION SYSTEM USING THE BLOCKCHAIN TECHNOLOGY " is the bonafide work of "GAUTAM UPADHYAYA (1613105046)"who carried out the project works under my supervision

**SIGNATURE OF HEAD**

Dr. MUNISH SHABARWAL,

PhD (Management), PhD (CS)

**PROFESSOR& DEAN,**

**School of Computing Science & Engineering**

**SIGNATURE OF SUPERVISOR**

Dr. SPS CHAUHAN, **ASSOCIATE PROFESSOR & SUPERVISOR**

**School of Computing Science & Engineering**

## ACKNOWLEDGEMENT:

This is an excellent opportunity to acknowledge and to thanks, all those persons without whose support and help this project would be impossible. We might prefer to add some heartfelt words for those who were a part of this project in numerous ways.

I would prefer to because of my project guide DR.SPS CHAUHAN and Rudra Pratap Ojha, for his indefatigable guidance, valuable suggestion, moral support, constant encouragement, and contribution of your time for the successful completion of project work. I'm very grateful to him, for providing all the facilities needed during the project development. At the outset, I sincerely thank all faculty members of my institution GALGOTIAS UNIVERSITY for his extra effort to create our session online and inspire all ideas.

I would prefer to thank all those that helped me directly or indirectly. Last but not the smallest amount, I'd prefer to acknowledge the continuing support of my friends, whose patience and encouragement during these long days and nights are paramount in making this project a reality.

THANK YOU.

**GAUTAM UPADHYAYA**

## DECLARATION:

I hereby declare that this submission is my very own work which, to the simplest of my knowledge and belief, it contains no material previously published or written by another person nor material which to a considerable extent has been accepted for the award of the other degree or diploma of the university or other institute of upper learning, except where due acknowledgment has been made within the text.

I inform that every data used in this report if it's taken from any site is clearly referenced under the reference section.

SIGNATURE

GAUTAM UPADHYAYA

16SCSE105053

Date: 03-may-2020

## ABSTRACT:

Blockchain technology is considered to be the driving force of the next fundamental revolution in information technology. It helps in creating decentralized application without having any focal control. Many implementations of blockchain technology are widely available today, each having its particular strength for a specific application domain. The implementation of blockchain is done by improvising the traditional Donation system. Donation system allows you to create a button and collect donations in any cryptocurrency. To improve security, the platform implements an Antifraud system, which checks donation profiles and campaigns for suspicious patterns and informs the community. It also layout possible feature trends for blockchain.

# TABLE OF CONTENT

# LIST OF FIGURES

# CHAPTER 1
# INTRODUCTION

A blockchain is a public ledger of all cryptocurrency transactions that have ever been executed. It is constantly growing as ‘completed’ blocks are added to it with a new set of recordings. The blocks are added to the blockchain in a linear, chronological order. Each node (computer connected to the Bitcoin network using a client that performs the task of validating and relaying transactions) gets a copy of the blockchain, which gets downloaded automatically upon joining the Bitcoin network. The blockchain has complete information about the addresses and their balances right from the genesis block to the most recently completed block.

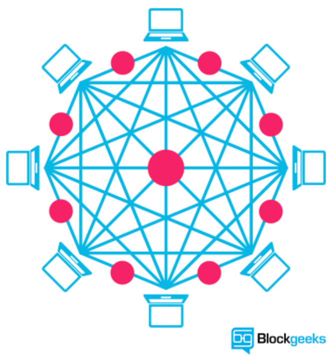
The blockchain is seen as the main technological innovation of Bitcoin, since it stands as proof of all the transactions on the network. A block is the ‘current’ part of a blockchain which records some or all of the recent transactions, and once completed goes into the blockchain as permanent database. Each time a block gets completed, a new block is generated. There is a countless number of such blocks in the blockchain. So are the blocks randomly placed in a blockchain? No, they are linked to each other (like a chain) in proper linear, chronological order with every block containing a hash of the previous block.

To use conventional banking as an analogy, the blockchain is like a full history of banking transactions. Bitcoin transactions are entered chronologically in a blockchain just the way bank transactions are. Blocks, meanwhile, are like individual bank statements.

Based on the Bitcoin protocol, the blockchain database is shared by all nodes participating in a system. The full copy of the blockchain has records of every Bitcoin transaction ever executed. It can thus provide insight about facts like how much value belonged a particular address at any point in the past.

The ever-growing size of the blockchain is considered by some to be a problem due to issues like storage and synchronization. On an average, every 10 minutes, a new block is appended to the block chain through mining.

A network of so-called computing "nodes" make up the blockchain.



Node (computer connected to the blockchain network using a client that performs the task of validating and relaying transactions) gets a copy of the blockchain, which gets downloaded automatically upon joining the blockchain network**.**

## 1.1How the donation system will work?

First of all, the donation will be done using its own coin or token. The number of tokens will depict the number of donate. Each user will get a token and they can donate using that token to whatever organization they intend to.

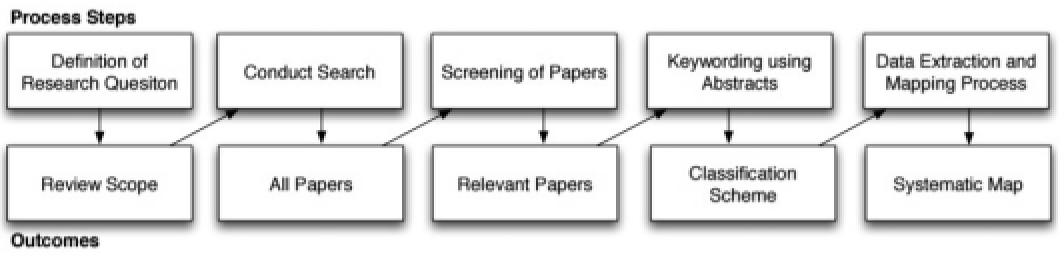
Donation system allows you to create a button or landing page in 1 click, and collect donations in any cryptocurrency. All payments are transferred to other tokens, after which the collecting party can either transfer them to cash, or exchange them for other cryptocurrencies.

The main goal of the Donate platform is to allow people and charitable funds to collect donations in cryptocurrency, making the collection process transparent and safe, and minimize corruption risks as much as possible. To improve security, the platform implements an Antifraud system, which checks donation profiles and campaigns for suspicious patterns and informs the community.

A blockchain is a distributed database that records all transactions that have ever occurred in the blockchain network. This database is replicated and shared among the network's participants. The main feature of blockchain is that it allows untrusted participants to communicate and send transactions between each other in a secure way without the need of a trusted third party. Blockchain is an ordered list of blocks, where each block is identified by its cryptographic hash. Each block references the block that came before it, resulting in a chain of blocks. Each block consists of a set of transactions. Once a block is created and appended to the blockchain, the transactions in that block cannot be changed or reverted. This is to ensure the integrity of the transactions and to prevent double-spending problem.

A smart contract is executable code that runs on the blockchain to facilitate, execute and enforce the terms of an agreement. The main aim of a smart contract is to automatically execute the terms of an agreement once the specified conditions are met. Thus, smart contracts promise low transaction fees compared to traditional systems that require a trusted third party to enforce and execute the terms of an agreement.

There is a website named Kickstarter in America that helps people in investing their money in the products and give chance to the product teams to get investment from people in their respective products. So smart contract will help in eliminating the middle men like Kickstarter and provide secure services at very nominal rate.
Smart contracts can be developed and deployed in different blockchain platforms (e.g., Ethereum, Bitcoin and NXT). Different platforms offer distinctive features for developing smart contracts. Some platforms support high-level programming languages to develop smart contracts. We will use Ethereum platform for building our smart contract.
Ethereum is a public blockchain platform that can support advanced and customized smart contracts with the help of Turing-complete programming language. Ethereum platform can support withdrawal limits, loops, financial contracts and gambling markets. The code of Ethereum smart contracts is written in a stack-based bytecode language and executed in Ethereum Virtual Machine (EVM). Several high-level languages (e.g., Solidity, Serpent and LLL) can be used to write Ethereum smart contracts. The code of those languages can then be compiled into EVM bytecodes to be run. Ethereum currently is the most common platform for developing smart contracts.



Ethereum is a public blockchain platform that can support advanced and customized smart contracts with the help of Turing-complete programming language. Ethereum platform can support withdrawal limits, loops, financial contracts and gambling markets. The code of Ethereum smart contracts is written in a stack-based bytecode language and executed in Ethereum Virtual Machine (EVM). Several high-level languages (e.g., Solidity, Serpent and LLL) can be used to write Ethereum smart contracts. The code of those languages can then be compiled into EVM bytecodes to be run. Ethereum currently is the most common platform for developing smart contracts.

## 1.2 APPLICATIONS OF BLOCKCHAIN

The applications of blockchain can be applied in every field. It is nowadays diversified in every field and helps in making secure and immutable network. The various applications of blockchain are:

Banking: Avoid risk of payment losses involved in banking transactions by adopting secure distributed ledger platform. Reduces transaction fees across cross-borders, corporate payments and remittances.

Retail and Consumer Products: Enhance your product quality, business reliability, product authenticity and customer loyalty through on-time delivery. Prove your business capability by delivering promises made.

Real Estate: Decentralized ledger platform helps to connect potential buyers with sellers. Helps to analyze the authenticity of ownership transfers, rental agreements and smart contracting through land registry management system.

Insurance: Transforms the way the insurance documents, claim settlements and fraud handlings are carried out.

Healthcare: Helps to prioritize patient health at all without compromising the quality of the health care service.

Food supply chain: Creates a tamper proof record to check the real information about expiration date, product journey from the farm to the shop.

A major advantage of blockchain technology is its distributed nature. In today’s capital markets, the transfer of value between two parties generally requires centralized transaction processors such as banks or credit card networks. These processors reduce counterparty risk for each party by serving as an intermediary but centralize credit risks with themselves. Each of these centralized processors maintains its own separate ledger; the transacting parties rely on these processors to execute transactions accurately and securely. For providing this service, the transaction processors receive a fee. In contrast, a blockchain allows parties to transact directly with each other through a single distributed ledger, thus eliminating one of the needs for centralized transaction processors. In addition to being efficient, the blockchain has other unique characteristics that make it a breakthrough innovation. Blockchain is considered reliable because full copies of the blockchain ledger are maintained by all active nodes. Thus, if one node goes offline, the ledger is still readily available to all other participants in the network. A blockchain lacks a single point of failure. In addition, each block in the chain refers to the previous blocks, which prevents deletion or reversing transactions once they are appended to the blockchain. Nodes on a blockchain network can come and go but the network integrity and reliability will remain intact as long as it is being used. In this way, no single party controls a blockchain and no single party can modify it or turn it off.

## 1.3 Blockchains Are Not Made Equal

Blockchain technology is a new form of database and each blockchain implementation may have different characteristics that make it unique. While the technology is emerging, there is a risk that a specific blockchain implementation does not live up to the promise of the technology. In the current ecosystem, there are two major classifications of blockchain networks: permission less and permissioned. The biggest difference is the determination of which parties are allowed access to the network. A blockchain may be shared publicly with anyone who has access to the Internet (i.e., permission less or “public” blockchain), or shared with only certain participants (i.e., permissioned or “private” blockchain). Permission less Blockchain A permission less blockchain is open to any potential user. For example, the Bitcoin blockchain is a public or permission less blockchain; anyone can participate as a node in the chain by agreeing to relay and validate transactions on the network thereby offering their computer processor as a node. Joining the blockchain is as simple as downloading the software and bitcoin ledger from the Internet. Because the blockchain maintains a list of every transaction ever performed, it reflects the full transaction history and account balances of all parties. Figure 1 is an example of a transfer of bitcoin (BTC) from one individual to another. When one party sends bitcoin (i.e., buyer sending value) to another party (i.e., seller receiving value), the Bitcoin blockchain is updated by the following process, including a process referred to as “mining”:

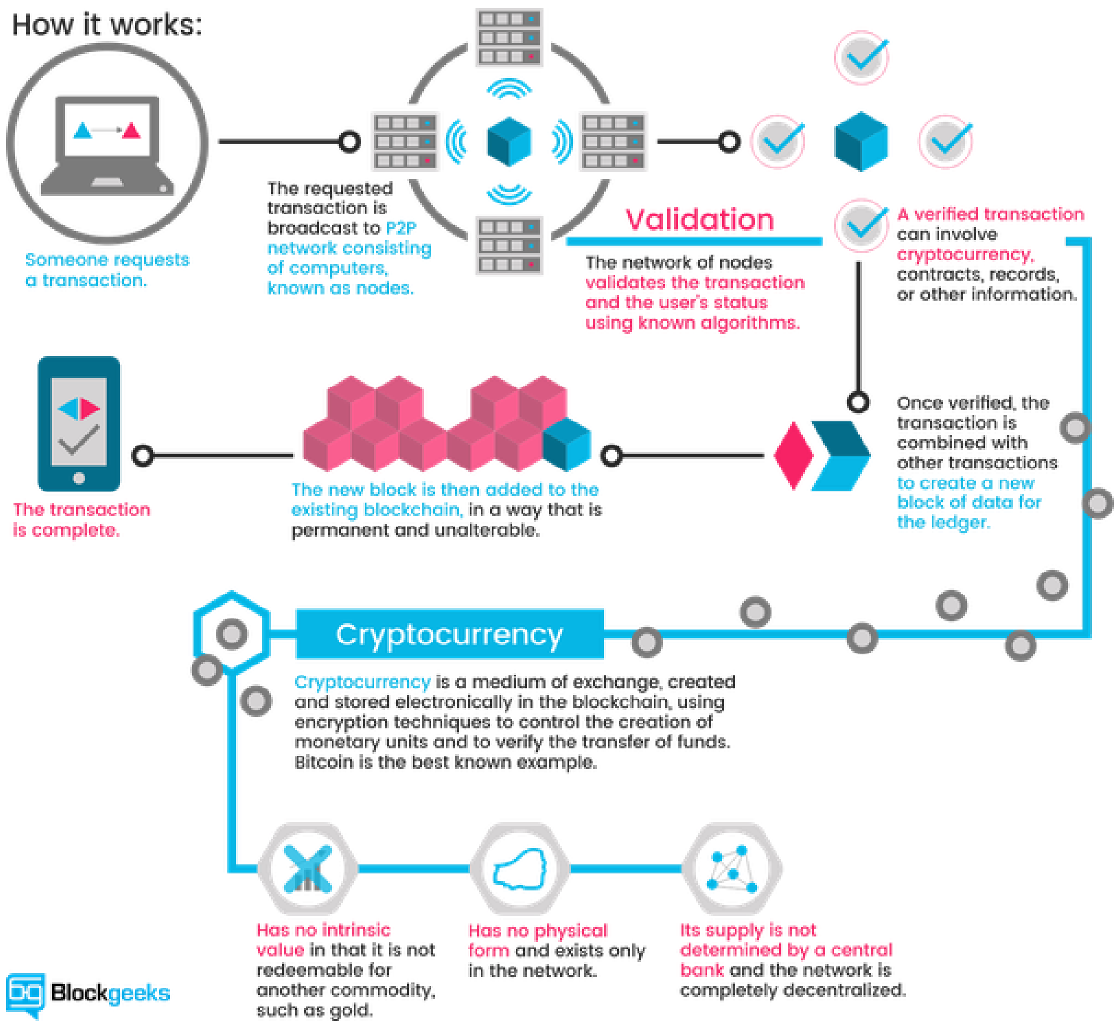
# CHAPTER 2
# SOFTWARE AND HARDWARE REQUIREMENTS

1. A computer that runs Ethereum Blockchain.
2. The project should run just fine on any Windows or Linux version.
3. Mist browser
4. Meta task keylogger
5. Ethereum virtual machine
6. Minimum 35GBs of storage on your hard drive
7. Sound Card:yes
8. Ram:4GB
9. CPU SPEED:AMD/INTEL DUAL CORE-CORE 2.4GHZ

# CHAPTER 3
# REVIEW OF LITERATURE

Blockchain is a decentralized, distributed and public digital ledger that is used to record transactions across many computers so that the record cannot be altered retroactively without the alteration of all subsequent blocks and the collusion of the network. This allows the participants to verify and audit transactions inexpensively. A blockchain database is managed autonomously using a peer-to-peer network and a distributed timestamping server. They are authenticated by mass collaboration powered by collective self-interests.

Blockchain technology was first used within Bitcoin and is a public ledger of all transactions. A blockchain stores these transactions in a block, the block eventually becomes completed as more transactions are carried out. Once complete it is then added in a linear, chronological order to the blockchain. The initial block in a blockchain is known as the 'Genesis block' or 'Block 0'. The genesis block is usually hardcoded into the software; it is special in that it doesn't contain a reference to a previous block. ('Genesis Block', 2015) Once the genesis block has been initialised 'Block 1' is created and when complete is attached to the genesis block. Each block has a transaction data part, copies of each transaction are hashed, and then the hashes are paired and hashed again, this continues until a single hash remains; also known as a merkle root (Figure 1). The block header is where the merkle root is stored. To ensure that a transaction cannot be modified each block also keeps a record of the previous blocks header, this means to change data you would have to Figure 1: Hash table 5 modify the block that records the transaction as well as all following blocks, as seen in Figure 2. (Bitcoin.org, 2009) A blockchain is designed to be accessed across a peer-to-peer network, each node/peer then communicates with other nodes for block and transaction exchange. Once connected to the network, peers start sending messages about other peers on the network, this creates a decentralised method of peer discovery. The purpose of the nodes within the network is to validate unconfirmed transactions and recently mined blocks, before a new node can start to do this it first has to carry out an initial block down



How it works:
Someone requests a transaction.
The requested transaction is broadcast to P2P network consisting of computers, known as nodes.
Validation
The network of nodes validates the transaction and the user's status using known algorithms.
A verified transaction can involve cryptocurrency, contracts, records, or other information.
Once verified, the transaction is combined with other transactions to create a new block of data for the ledger.
The new block is then added to the existing blockchain, in a way that is permanent and unalterable.
The transaction is complete.
Cryptocurrency
Cryptocurrency is a medium of exchange, created and stored electronically in the blockchain, using encryption techniques to control the creation of monetary units and to verify the transfer of funds. Bitcoin is the best known example.
Has no intrinsic value in that it is not redeemable for another commodity, such as gold.
Has no physical form and exists only in the network.
Its supply is not determined by a central bank and the network is completely decentralized.
Blockgeeks

Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality. It supports a modified version of Nakamoto consensus via transaction-based state transitions.

Ether is a cryptocurrency whose blockchain is generated by the Ethereum platform. *Ether* cans be transferred between accounts and used to compensate participant mining nodes for computations performed. Ethereum provides a decentralized Turing-complete virtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes. "Gas", an internal transaction pricing mechanism, is used to mitigate spam and allocate resources on the network.

Mist is an electron application, meaning that its a desktop hybrid app with a web interface as well. This allows for faster development and changes of the Mist interface and helps with the browser part of Mist. Its important that you recognize Mist is in beta, so expect to run into a few issues here and there.

Mist is powerful in that it includes a Geth node that runs in the background upon launch; a connection to the Ethereum blockchain is made right away. But because we want to practice developing a smart contract, we are going to run a node on our private network and will not need Mist to connect to the default Geth node.

Bitcoin is a cryptocurrency made in January 2009. It pursues the thoughts set out in a whitepaper by the secretive and pseudonymous developer Satoshi Nakamoto. Bitcoin offers the guarantee of lower exchange charges than customary online installment components and is worked by a decentralized position, dissimilar to official monetary standards. Bitcoin is one of the primary advanced monetary forms to utilize distributed innovation to encourage moment installments. The autonomous people and organizations who possess the overseeing registering power and take an interest in the Bitcoin arrange, otherwise called "miners" are spurred by remunerations (the arrival of new bitcoin) and exchange expenses paid in bitcoin. These excavators can be thought of as the decentralized authority implementing the validity of the Bitcoin organize. New bitcoin is being discharged to the excavators at a fixed, however intermittently declining rate, with the end goal that the all-out supply of bitcoins approaches 21 million. Right now, there are approximately 3 million bitcoins which presently can't seem to be mined. Along these lines, Bitcoin (and any digital money created through a comparable procedure) works uniquely in contrast to fiat cash; in brought together financial frameworks, cash is discharged at a rate coordinating the development in products trying to keep up value steadiness, while a decentralized framework like Bitcoin sets the discharge rate early and as indicated by a calculation. The white paper which is written about bitcoin is an official document outlining proposals on the issue. It includes following sections:

1. Transactions
2. Timestamp servers
3. Proof of work
4. Network
5. Incentive
6. Reclaiming Disk Space
7. Simplified payment verification
8. Combining and splitting value
9. Privacy

Smart contracts are a method to automate the contracting process and enable monitoring and enforcement of contractual promises with minimal human intervention. Automation can improve efficiency, reduce settlement times and operational errors. Because using smart contract technology requires the translation of all contractual terms into logic, it may also improve contract compliance by reducing ambiguity in certain situations. As smart contracts continue to evolve, inherent risks may emerge that need to be mitigated. For example, when setting up a smart contract, the parties may decide not to address every possible outcome, or they may include some level of flexibility so they do not limit themselves. This could lead to smart contracts with vulnerabilities or errors that could lead to unexpected business outcomes. Parties may find it difficult to renegotiate the terms of a deal or modify terms due to an unforeseen error. Also, incomplete or flexible contracts can lead to settlement problems and disputes. Perhaps most importantly, however, at the date of this publication, smart contracts have not been tested thoroughly in the court system. Nevertheless, smart contracts offer a compelling use case for blockchain adoption.

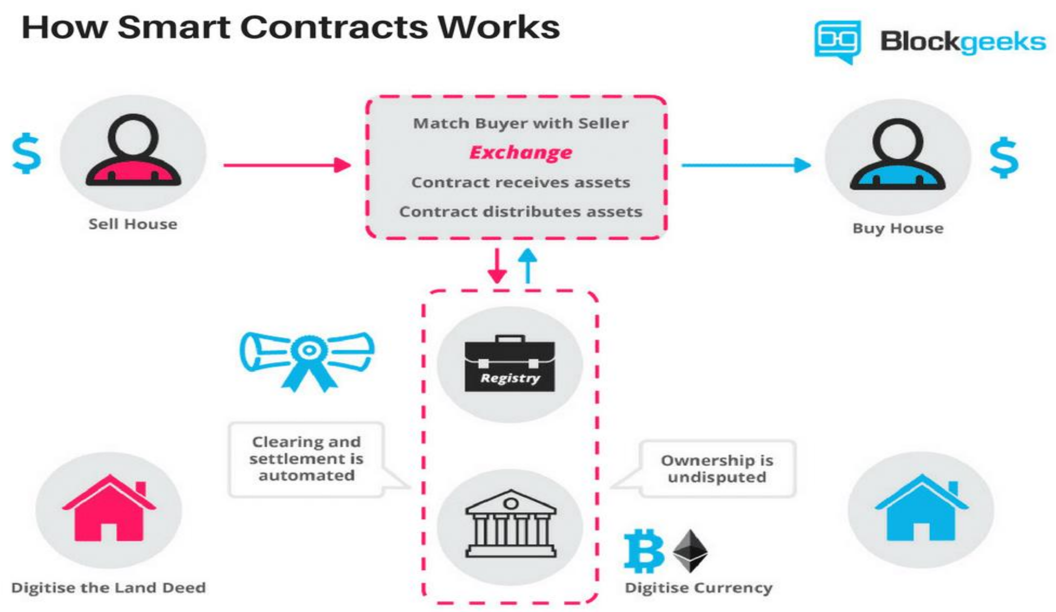
A digital signature on a blockchain can be validated by participants that know the public key, or the ‘address’, of the issuing party. However, the signature does not verify the correctness of the data itself, whether it pertains to a measurement from a humidity sensor, or the content of a signed report. For this reason, moderators may need to by employed and incentivised to resolve disputes. For example, if a project party signs and publishes a document on an agreed deliverable to the blockchain, all parties, including the Project Manager, can verify that the document has been signed and that the timestamp proves the document was submitted in time. However, this does not guarantee that the content of the document meets the agreed terms. If someone disputes the correctness of the document’s content, there are three possible outcomes: • All parties talk to each other and attempt to resolve the problem by giving the issuer time to rework and re-issue the document. • All parties open a case with the moderator, paid a defined percentage of the fee in question, to resolve the issue by checking the agreed terms and the issued document. The moderator can then decide whether or not locked funds should be released to pay the issuing party. • As a last resort, the case can be handed to lawyers to resolve the dispute, as with non blockchain based contracts.

# CHAPTER 4
# SOFTWARE REQUIREMENT ANALYSIS

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible. Smart contracts were first proposed by Nick Szabo who coined the term, in 1994

The best way to describe smart contracts is to compare the technology to a vending machine. Ordinarily, you would go to a lawyer or a notary, pay them, and wait while you get the document. With smart contracts, you simply drop a bitcoin into the vending machine (i.e. ledger), and your escrow, driver's license, or whatever drops into your account. More so, smart contracts not only define the rules and penalties around an agreement in the same way that a traditional contract does, but also automatically enforce those obligations.

## 4.1 How smart contract works?



## 4.2 MIST BROWSER

Mist is an electron application, meaning that its a desktop hybrid app with a web interface as well. This allows for faster development and changes of the Mist interface and helps with the browser part of Mist. Its important that you recognize Mist is in beta, so expect to run into a few issues here and there.

Mist is powerful in that it includes a Geth node that runs in the background upon launch; a connection to the Ethereum blockchain is made right away. But because we want to practice developing a smart contract, we are going to run a node on our private network and will not need Mist to connect to the default Geth node.

Ethereum is an open source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality. It supports a modified version of Nakamoto consensus via transaction-based state transitions. Ether is a cryptocurrency generated by the Ethereum platform and used to compensate mining nodes for computations performed.[3] Each Ethereum account has an ether balance and ether may be transferred from one account to another.

Ethereum provides a decentralized virtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes.[4] The virtual machine's instruction set, in contrast to others like Bitcoin Script, is thought to be Turing-complete. "Gas", an internal transaction pricing mechanism, is used to mitigate spam and allocate resources on the network.

Ethereum was proposed in late 2013 by Vitalik Buterin, a cryptocurrency researcher and programmer. Development was funded by an online crowdsale that took place between July and August 2014.[4] The system then went live on 30 July 2015, with 72 million coins "premined". This accounts for about 68 percent of the total circulating supply in 2019.

In 2016, as a result of the exploitation of a flaw in The DAO project's smart contract software, and subsequent theft of $50 million worth of ether,[6] Ethereum was split into two separate blockchains – the new separate version became Ethereum (ETH) with the theft reversed,[7] and the original continued as Ethereum Classic (ETC).

Ethereum's smart contracts are based on different computer languages, which developers use to program their own functionalities. Smart contracts are high-level programming abstractions that are compiled down to EVM bytecode and deployed to the Ethereum blockchain for execution. They can be written in Solidity (a language library with similarities to C and JavaScript), Serpent (similar to Python, but deprecated), LLL (a low-level Lisp-like language), and Mutan (Go-based, but deprecated). There is also a research-oriented language under development called Vyper (a strongly-typed Python-derived decidable language).

Smart contracts can be public, which opens up the possibility to prove functionality, e.g. self-contained provably fair casinos.

## 4.3 Why we need to reform the present system?

In future, the platform plans to sign agreements with large charitable foundations and develop an API so that the existing cash collection services can receive donations in the cryptocurrency.

To further develop the TrueDonate platform after the ICO, an internal fund will be created. 3% of all proceeds will go to the fund. This is relevant only for proceeds of creative people (bloggers, artists, musicians). Charity is an exception and is not subject to deductions (100% of the funds are transferred to the recipient).

**Problems**

-The current system doesn't have transparency.
-the EVM can easily be accessed and corrupted
-elections can be biased

## 4.4 How it is different from current system?

The idea of decentralization

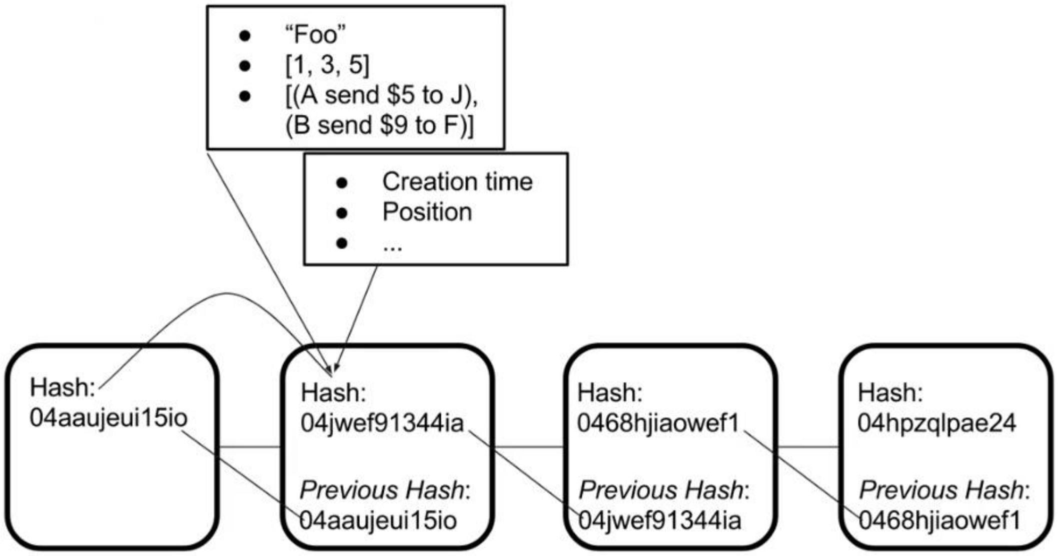
By design, the blockchain is a decentralized technology. That means, no central authority has a control over it. In a blockchain network, the communication or transaction is only between the two nodes of the network and no third party is required here.

**Transparency**

data is embedded within the network as a whole, by definition it is public.

**It cannot be corrupted**

altering any unit of information on the blockchain would mean using a huge amount of computing power to override the entire network.

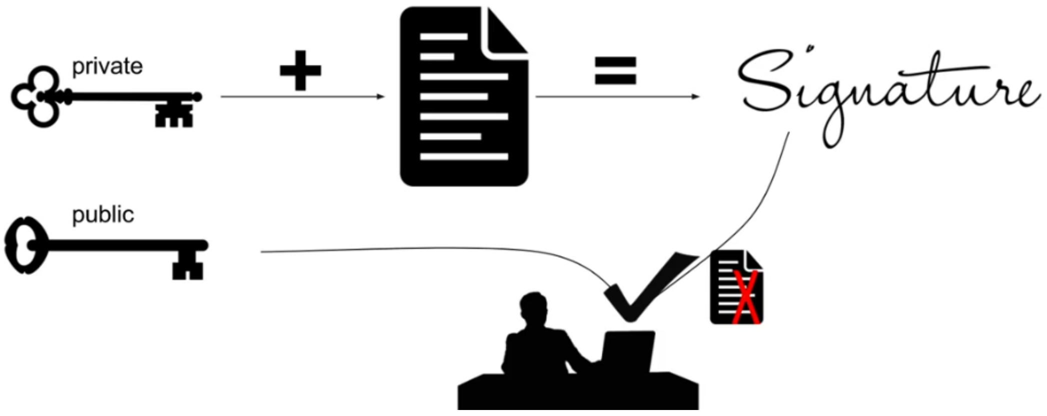

"Foo"
[1, 3, 5]
[(A send $5 to J), (B send $9 to F)]
Creation time
Position
...
Hash: 04aaujeui15io
Hash: 04jwef91344ia
Previous Hash: 04aaujeui15io
Hash: 0468hjiaowef1
Previous Hash: 04jwef91344ia
Hash: 04hpzqlpae24
Previous Hash: 0468hjiaowef1

1

Digital Signature is a way of authentication used to validate the authenticity and integrity of a message or a transaction. There are two keys, public key and private key. Public keys are publicly known and essential for identification. Private keys are secret keys which are used for authentication and encryption. Private key grants ownership of the funds to the user whereas public key can be derived with the help of an algorithm.

Wallets play important role in blockchain as they store both public key and private key of an individual. The public key is address of the wallet. The wallet also helps in signing the transactions. The transactions are temporarily unconfirmed and have to be confirmed by the miners. Mining is the process of sorting through large data sets to identify patterns and solve complex computational problems.

2



3

It includes block of transactions by solving a proof of work algorithm.
Difficult to solve and computationally expensive
Once solved, miners can add the block and other miners will verify.
The miners are rewarded for adding the block to the chain.
The difficulty can adjust according to the new number of blocks coming in.

# CHAPTER 5
# SOFTWARE DESIGN

Blocks hold batches of valid transactions that are hashed and encoded into a Merkle tree. Each block includes the cryptographic hash of the prior block in the blockchain, linking the two. The linked blocks form a chain. This iterative process confirms the integrity of the previous block, all the way back to the original genesis block.

## 6.1 BLOCK TIME

The block time is the average time it takes for the network to generate one extra block in the blockchain. Some blockchains create a new block as frequently as every five seconds. By the time of block completion, the included data becomes verifiable. In cryptocurrency, this is practically when the money transaction takes place, so a shorter block time means faster transactions. The block time for Ethereum is set to between 14 and 15 seconds, while for bitcoin it is 10 minutes.

Picture a spreadsheet that is duplicated thousands of times across a network of computers. Then imagine that this network is designed to regularly update this spreadsheet and you have a basic understanding of the blockchain.

Information held on a blockchain exists as a shared — and continually reconciled — database. This is a way of using the network that has obvious benefits. The blockchain database isn't stored in any single location, meaning the records it keeps are truly public and easily verifiable. No centralized version of this information exists for a hacker to corrupt. Hosted by millions of computers simultaneously, its data is accessible to anyone on the internet.

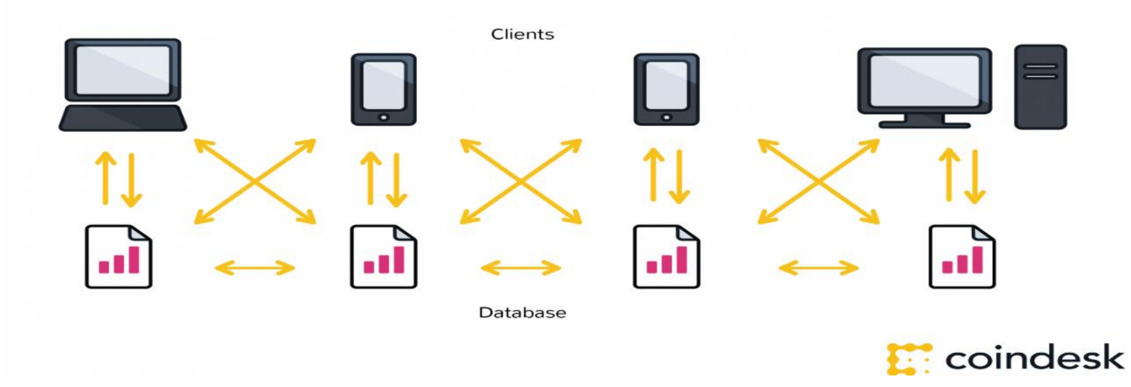
The benefit and need for a distributed network can be understood by the 'if a tree falls in the forest' thought experiment.

If a tree falls in a forest, with cameras to record its fall, we can be pretty certain that the tree fell. We have visual evidence, even if the particulars (why or how) may be unclear.

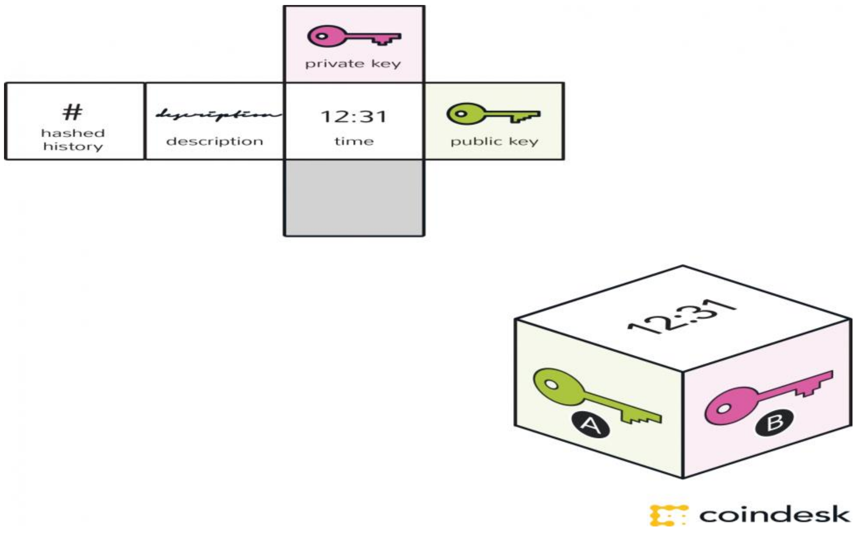
Much of the value of the bitcoin blockchain is that it is a large network where validators, like the cameras in the analogy, reach a consensus that they witnessed the same thing at the same time. Instead of cameras, they use mathematical verification.

In short, the size of the network is important to secure the network.

That is one of the bitcoin blockchain's most attractive qualities -- it is so large and has amassed so much computing power. At time of writing, bitcoin is secured by 3,500,000 TH/s, more than the 10,000 largest banks in the world combined. Ethereum, which is still more immature, is secured by about 12.5 TH/s, more than Google and it is only two years old and still basically in test mode.



## 4 PROTOCOL



Proof of Work (POW) was introduced by bitcoin and is used by other cryptocurrencies also. The proof of work comes in the form of an answer to a mathematical problem. The "proof of work" comes as a response to a numerical issue, one that requires extensive work to land at, however is effectively confirmed to be right once the appropriate response has been come to.

Results of using Smart Contracts are:

- Elimination of middle men from the market.
- More secure way of transactions.
- Block tampering is very difficult leading to more secure network.
- Can be used at various places like storing medical records, creating digital notary and collecting taxes.

Ethereum is a public blockchain platform that can support advanced and customized smart contracts with the help of Turing-complete programming language. Ethereum platform can support withdrawal limits, loops, financial contracts and gambling markets. The code of Ethereum smart contracts is written in a stack-based bytecode language and executed in Ethereum Virtual Machine

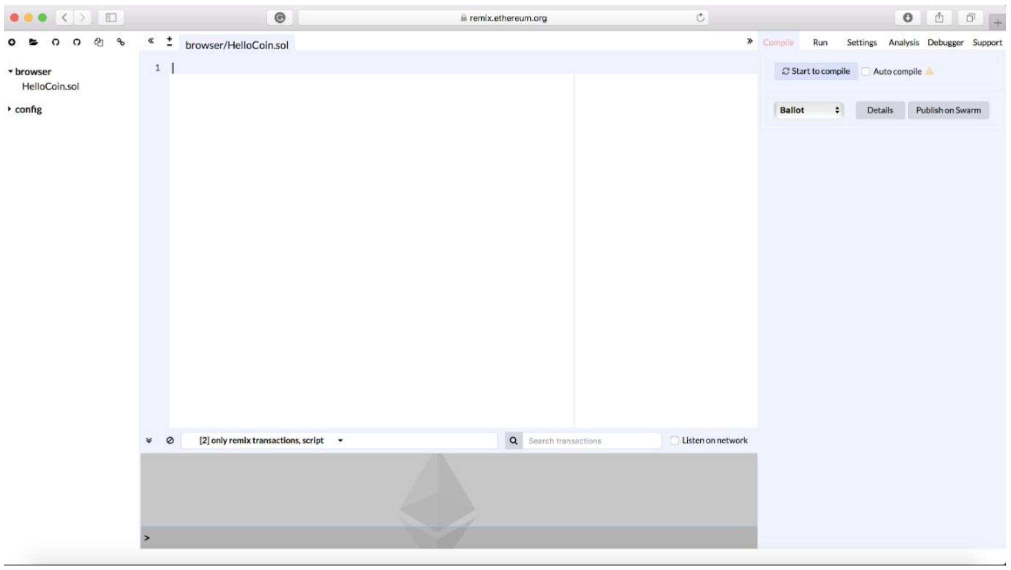
(EVM). Several high-level languages (e.g., Solidity, Serpent and LLL) can be used to write Ethereum smart contracts. The code of those languages can then be compiled into EVM bytecodes to be run. Ethereum currently is the most common platform for developing smart contracts.

Smart contracts can be developed and deployed in different blockchain platforms (e.g., Ethereum, Bitcoin and NXT). Different platforms offer distinctive features for developing smart contracts. Some platforms support high-level programming languages to develop smart contracts. We will use Ethereum platform for building our smart contract.

A blockchain is a digital ledger created to capture transactions conducted among various parties in a network. It is a peer-to-peer, Internet-based distributed ledger which includes all transactions since its creation. All participants (i.e., individuals or businesses) using the shared database are “nodes” connected to the blockchain,5 each maintaining an identical copy of the ledger. Every entry into a blockchain is a transaction that represents an exchange of value between participants (i.e., a digital asset that represents rights, obligations or ownership). In practice, many different types of blockchains are being developed and tested. However, most blockchains follow this general framework and approach. When one participant wants to send value to another, all the other nodes in the network communicate with each other using a pre-determined mechanism to check that the new transaction is valid. This mechanism is referred to as a consensus algorithm.6 Once a transaction has been accepted by the network, all copies of the ledger are updated with the new information. Multiple transactions are usually combined into a “block” that is added to the ledger. Each block contains information that refers back to previous blocks and thus all blocks in the chain link together in the distributed identical copies. Participating nodes can add new, time-stamped transactions, but participants cannot delete or alter the entries once they have been validated and accepted by the network. If a node modified a previous block, it would not sync with the rest of the network and would be excluded from the blockchain. A properly functioning blockchain is thus immutable despite lacking a central administrator.

As a Solidity new learner, a light-weight and easy development environment would be a good starting point. In this tutorial, we use Remix Solidity IDE to create a new token.

Open the Remix Solidity IDE link: https://remix.ethereum.org/ , you will see the layout of Remix IDE, as shown below.



Remix Solidity IDE interface

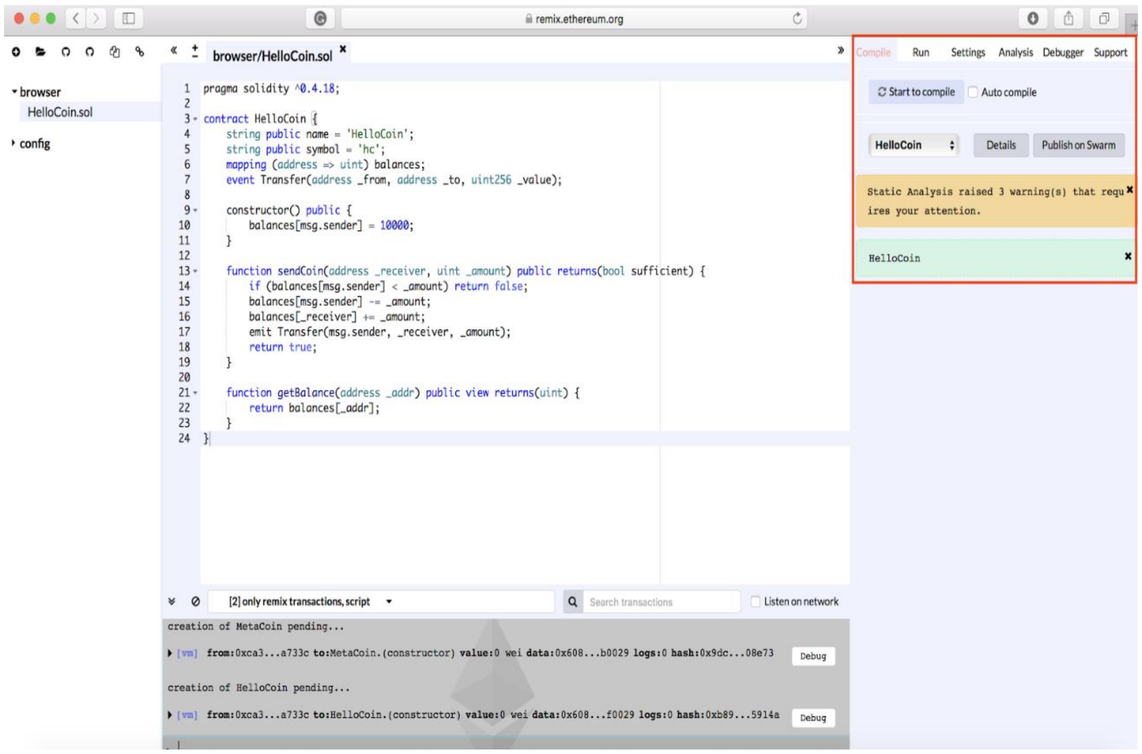
The left side shows a list of Solidity files. The middle is the code writing and editing section, which also provides syntax highlighting mapped to Solidity keywords. Compilation warnings and errors will be displayed in the gutter. On the right side, there are some buttons for compiling and running Solidity codes. You can also choose which VM you are about to use.

Now, let’s create a new Solidity file, called “HelloCoin”. The code we wrote to create a new currency is shown as below.

pragma solidity ^0.4.18;

```
contract HelloCoin {string public name = 'HelloCoin';
//currency name. Please feel free to change itstring public symbol = 'hc';
//choose a currency symbol. Please feel free to change itmapping (address => uint)
balances;
//a key-value pair to store addresses and their account balancesevent Transfer(address
_from, address _to, uint256 _value);
// declaration of an event. Event will not do anything but add a record to the
logconstructor() public {
//when the contract is created, the constructor will be called
automaticallybalances[msg.sender] = 10000;
//set the balances of creator account to be 10000. Please feel free to change it to any
number you want.}function sendCoin(address _receiver, uint _amount) public
returns(bool sufficient) {if (balances[msg.sender] < _amount) return false;
// validate transferbalances[msg.sender] -= _amount;balances[_receiver] +=
_amount;emit Transfer(msg.sender, _receiver, _amount);
// complete coin transfer and call event to record the logreturn true;}function
getBalance(address _addr) public view returns(uint) {
//balance checkreturn balances[_addr];}}
```

Above is a simple example of a cryptocurrency contract. The code is very straightforward. We create some variables to record our currency’s name and symbol — “HelloCoin” and “hc”. We use mapping to keep track of the balances of all addresses. In the constructor, it gives the initial creator 10,000 value of tokens. It’s consistent with the basic definition of currency as I mentioned in the beginning — a database with one operation, that is subtracting X units from A and adding X units to B. However, the above example is not standard compatible and cannot be expected to talk to other coin/token contracts, unless your implement your coins with standards, for instance ERC20.



```
pragma solidity ^0.4.18;

contract HelloCoin {
    string public name = 'HelloCoin';
    string public symbol = 'hc';
    mapping (address => uint) balances;
    event Transfer(address _from, address _to, uint256 _value);

    constructor() public {
        balances[msg.sender] = 10000;
    }

    function sendCoin(address _receiver, uint _amount) public returns(bool sufficient) {
        if (balances[msg.sender] < _amount) return false;
        balances[msg.sender] -= _amount;
        balances[_receiver] += _amount;
        emit Transfer(msg.sender, _receiver, _amount);
        return true;
    }

    function getBalance(address _addr) public view returns(uint) {
        return balances[_addr];
    }
}
```

```
Static Analysis raised 3 warning(s) that requires your attention.

HelloCoin
```

```
creation of MetaCoin pending...

[vm] from:0xca3...a733c to:MetaCoin.(constructor) value:0 wei data:0x608...b0029 logs:0 hash:0x9dc...08e73

creation of HelloCoin pending...

[vm] from:0xca3...a733c to:HelloCoin.(constructor) value:0 wei data:0x608...f0029 logs:0 hash:0xb89...5914a
```

# CHAPTER 6

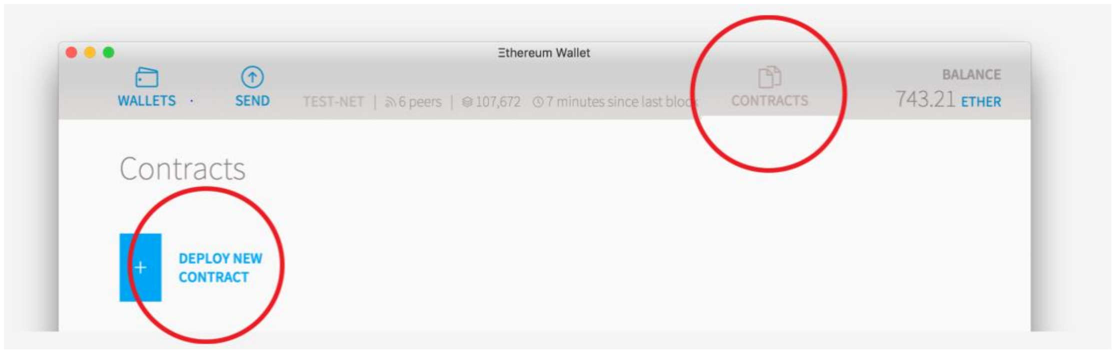
# THE CODE

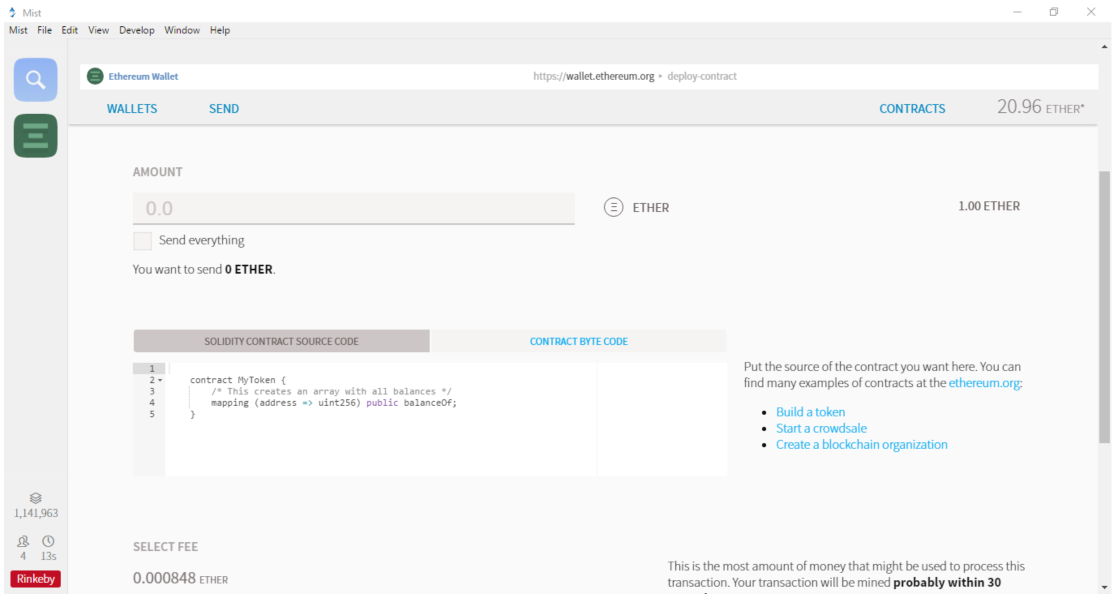
```
pragma solidity ^0.4.16;

interface tokenRecipient { function receiveApproval(address _from, uint256
_value, address _token, bytes _extraData) public; }

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;

    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;

    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);

    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);

    /**
     * Constrctor function
     *
     * Initializes contract with initial supply tokens to the creator of the contract
     */
    function TokenERC20(
        uint256 initialSupply,
        string tokenName,
        string tokenSymbol
    ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals);  // Update total supply
with the decimal amount
        balanceOf[msg.sender] = totalSupply;                // Give the creator all initial
tokens
        name = tokenName;                                   // Set the name for display purposes
        symbol = tokenSymbol;                               // Set the symbol for display
purposes
    }

    /**
     * Internal transfer, only can be called by this contract
     */
    function _transfer(address _from, address _to, uint _value) internal {
        // Prevent transfer to 0x0 address. Use burn() instead
        require(_to != 0x0);
        // Check if the sender has enough
        require(balanceOf[_from] >= _value);
        // Check for overflows
        require(balanceOf[_to] + _value > balanceOf[_to]);
        // Save this for an assertion in the future
        uint previousBalances = balanceOf[_from] + balanceOf[_to];
        // Subtract from the sender
        balanceOf[_from] -= _value;
        // Add the same to the recipient
        balanceOf[_to] += _value;
        Transfer(_from, _to, _value);
        // Asserts are used to use static analysis to find bugs in your code. They
should never fail
        assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
    }

    /**
     * Transfer tokens
     *
     * Send `_value` tokens to `_to` from your account
     *
     * @param _to The address of the recipient
     * @param _value the amount to send
     */
    function transfer(address _to, uint256 _value) public {
        _transfer(msg.sender, _to, _value);
    }
    /**
     * Transfer tokens from other address
     *
     * Send `_value` tokens to `_to` in behalf of `_from`
     *
     * @param _from The address of the sender
     * @param _to The address of the recipient
     * @param _value the amount to send
     */
    function transferFrom(address _from, address _to, uint256 _value) public
returns (bool success) {
        require(_value <= allowance[_from][msg.sender]);     // Check allowance
        allowance[_from][msg.sender] -= _value;
        _transfer(_from, _to, _value);
        return true;
    }

    /**
     * Set allowance for other address
     *
     * Allows `_spender` to spend no more than `_value` tokens in your behalf
     *
     * @param _spender The address authorized to spend
     * @param _value the max amount they can spend
     */
    function approve(address _spender, uint256 _value) public
```

```
        returns (bool success) {
        allowance[msg.sender][_spender] = _value;
        return true;
    }

    /**
     * Set allowance for other address and notify
     *
     * Allows `_spender` to spend no more than `_value` tokens in your behalf, and
then ping the contract about it
     *
     * @param _spender The address authorized to spend
     * @param _value the max amount they can spend
     * @param _extraData some extra information to send to the approved contract
     */
    function approveAndCall(address _spender, uint256 _value, bytes _extraData)
        public
        returns (bool success) {
        tokenRecipient spender = tokenRecipient(_spender);
        if (approve(_spender, _value)) {
            spender.receiveApproval(msg.sender, _value, this, _extraData);
            return true;
        }
    }

    /**
     * Destroy tokens
     *
     * Remove `_value` tokens from the system irreversibly
     *
     * @param _value the amount of money to burn
     */
    function burn(uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);   // Check if the sender has
enough
        balanceOf[msg.sender] -= _value;            // Subtract from the sender
        totalSupply -= _value;                      // Updates totalSupply
        Burn(msg.sender, _value);
        return true;
    }

    /**
     * Destroy tokens from other account
     *
     * Remove `_value` tokens from the system irreversibly on behalf of `_from`.
     *
     * @param _from the address of the sender
     * @param _value the amount of money to burn
     */
    function burnFrom(address _from, uint256 _value) public returns (bool
success) {
        require(balanceOf[_from] >= _value);                // Check if the targeted
balance is enough
        require(_value <= allowance[_from][msg.sender]);    // Check allowance
        balanceOf[_from] -= _value;                          // Subtract from the targeted
balance
        allowance[_from][msg.sender] -= _value;              // Subtract from the sender's
allowance
        totalSupply -= _value;                               // Update totalSupply
        Burn(_from, _value);
        return true;
    }
}
```

# CHAPTER 7

# TESTING





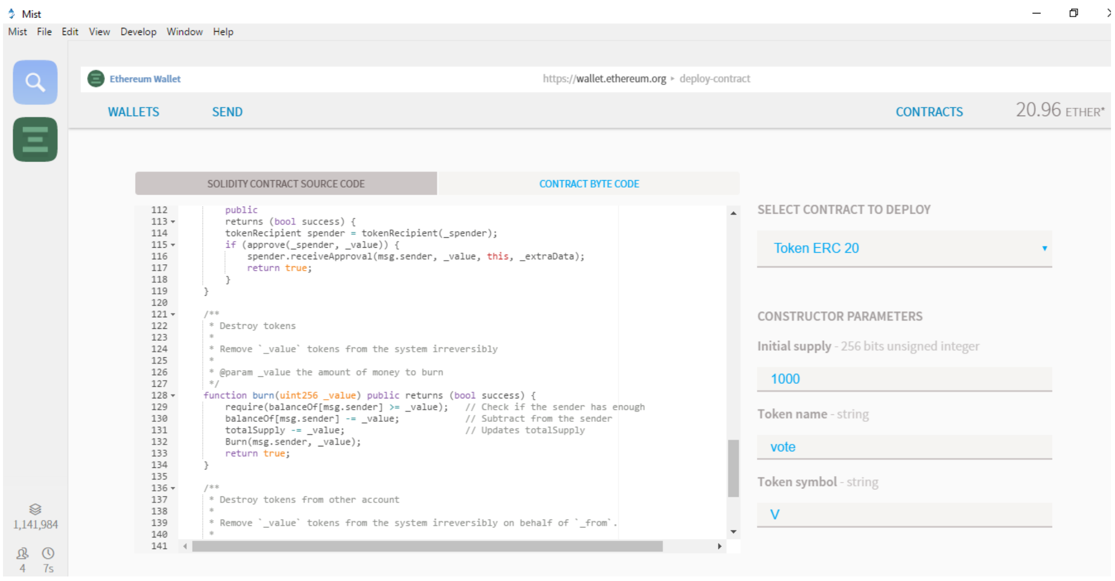
You want to send **0 ETHER**.

```
contract MyToken {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;
}
```

Put the source of the contract you want here. You can find many examples of contracts at the ethereum.org:

- Build a token
- Start a crowdsale
- Create a blockchain organization

This is the most amount of money that might be used to process this transaction. Your transaction will be mined **probably within 30**


Mist
Mist File Edit View Develop Window Help
Ethereum Wallet
https://wallet.ethereum.org ▸ deploy-contract
WALLETS
SEND
CONTRACTS
20.96 ETHER*
SOLIDITY CONTRACT SOURCE CODE
CONTRACT BYTE CODE
1,141,984
4
7s

```
        public
        returns (bool success) {
        tokenRecipient spender = tokenRecipient(_spender);
        if (approve(_spender, _value)) {
            spender.receiveApproval(msg.sender, _value, this, _extraData);
            return true;
        }
    }

    /**
     * Destroy tokens
     *
     * Remove `_value` tokens from the system irreversibly
     *
     * @param _value the amount of money to burn
     */
    function burn(uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);   // Check if the sender has enough
        balanceOf[msg.sender] -= _value;            // Subtract from the sender
        totalSupply -= _value;                      // Updates totalSupply
        Burn(msg.sender, _value);
        return true;
    }

    /**
     * Destroy tokens from other account
     *
     * Remove `_value` tokens from the system irreversibly on behalf of `_from`.
     *
```

SELECT CONTRACT TO DEPLOY

Token ERC 20

CONSTRUCTOR PARAMETERS

Initial supply - 256 bits unsigned integer

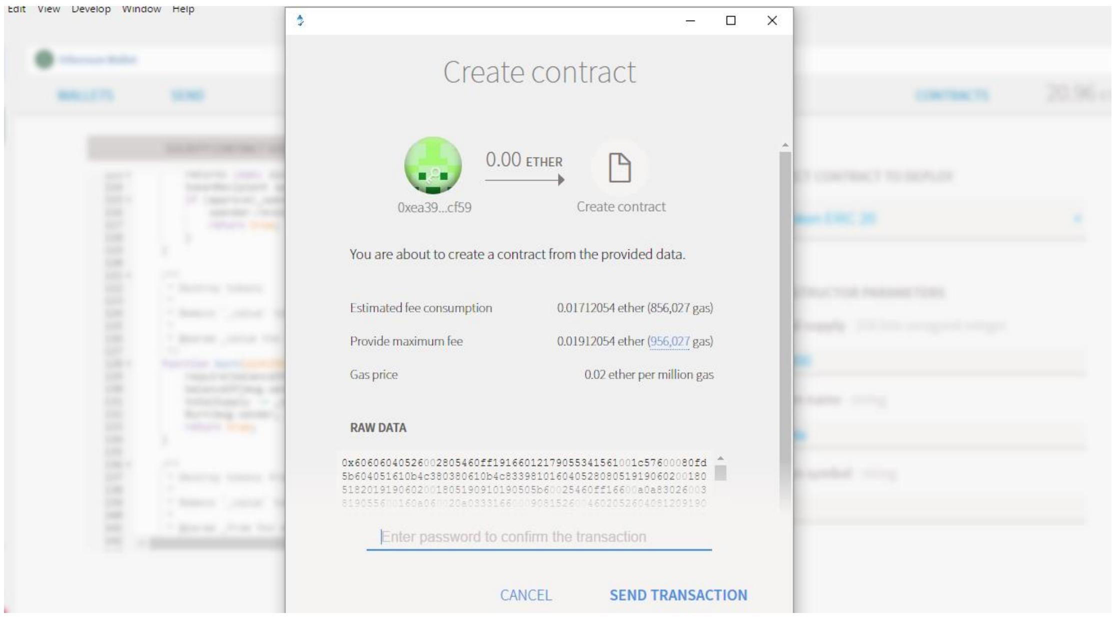
1000

Token name - string

vote

Token symbol - string

V



Create contract

0.00 ETHER

0xea39...cf59

Create contract

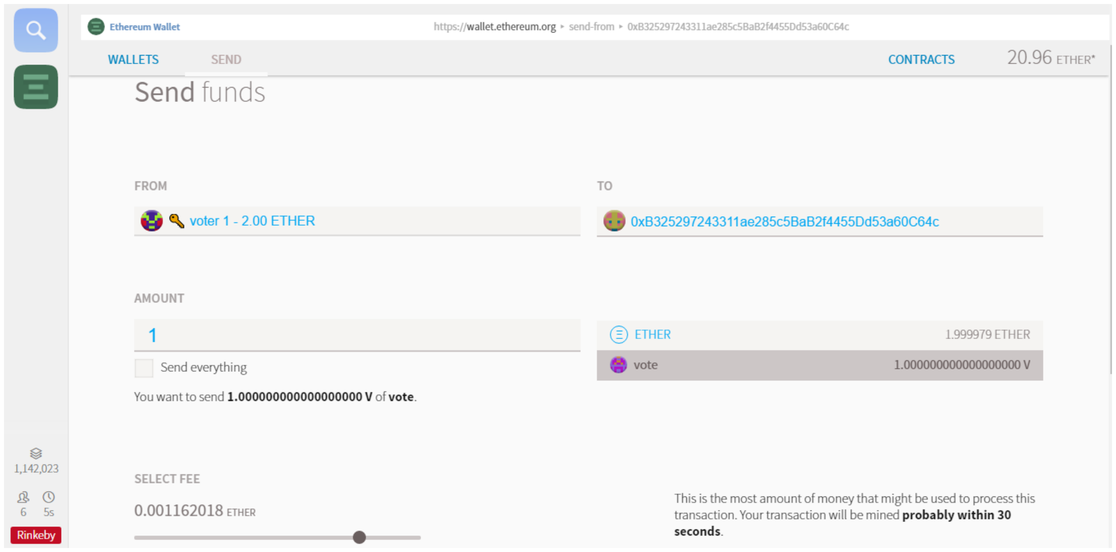
You are about to create a contract from the provided data.

| | |
|---|---|
| Estimated fee consumption | 0.01712054 ether (856,027 gas) |
| Provide maximum fee | 0.01912054 ether (956,027 gas) |
| Gas price | 0.02 ether per million gas |

RAW DATA

0x6060604052600280546 0ff191660121790553415610 01c57600080fd
5b604051610b4c380380610b4c833981016040528080519190602001 80
5182019190602001805190910190505b600254 60ff166 00a0a83026003
8190556 00160a06 0020a033316600090815260 046020526040812 09190

Enter password to confirm the transaction



Ethereum Wallet https://wallet.ethereum.org ▸ send-from ▸ 0xB325297243311ae285c5BaB2f4455Dd53a60C64c

WALLETS SEND CONTRACTS 20.96 ETHER*

Send funds

FROM: voter 1 - 2.00 ETHER

TO: 0xB325297243311ae285c5BaB2f4455Dd53a60C64c

AMOUNT: 1

| | |
|---|---|
| ETHER | 1.999979 ETHER |
| vote | 1.000000000000000000 V |

Send everything

You want to send **1.000000000000000000 V** of **vote**.

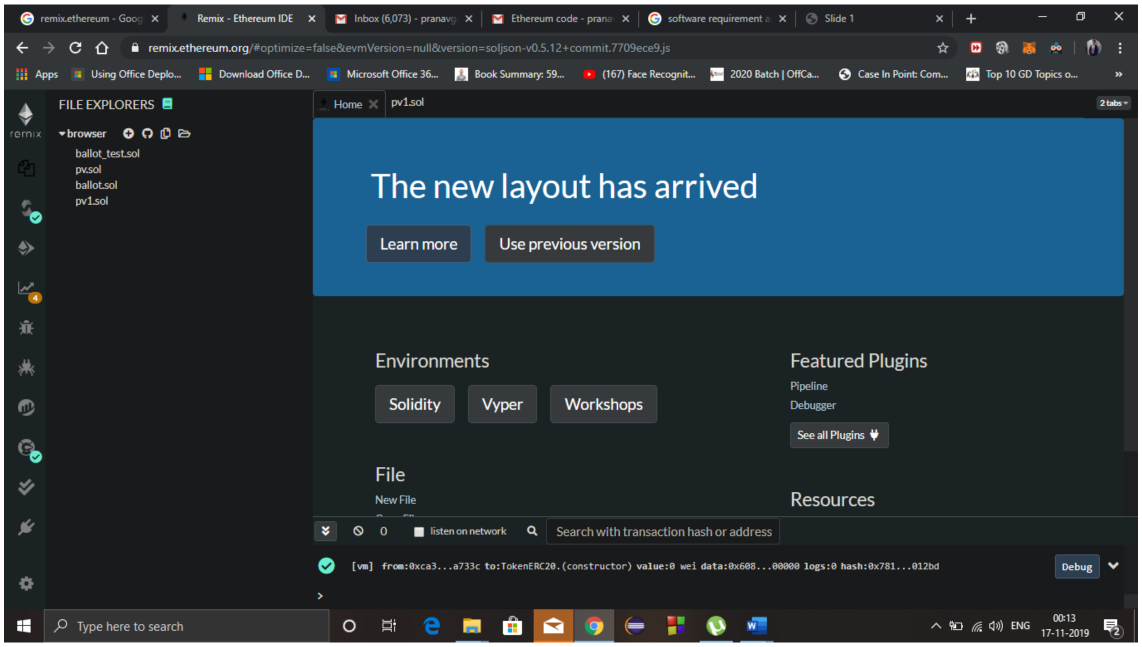
SELECT FEE: 0.001162018 ETHER

This is the most amount of money that might be used to process this transaction. Your transaction will be mined **probably within 30 seconds**.

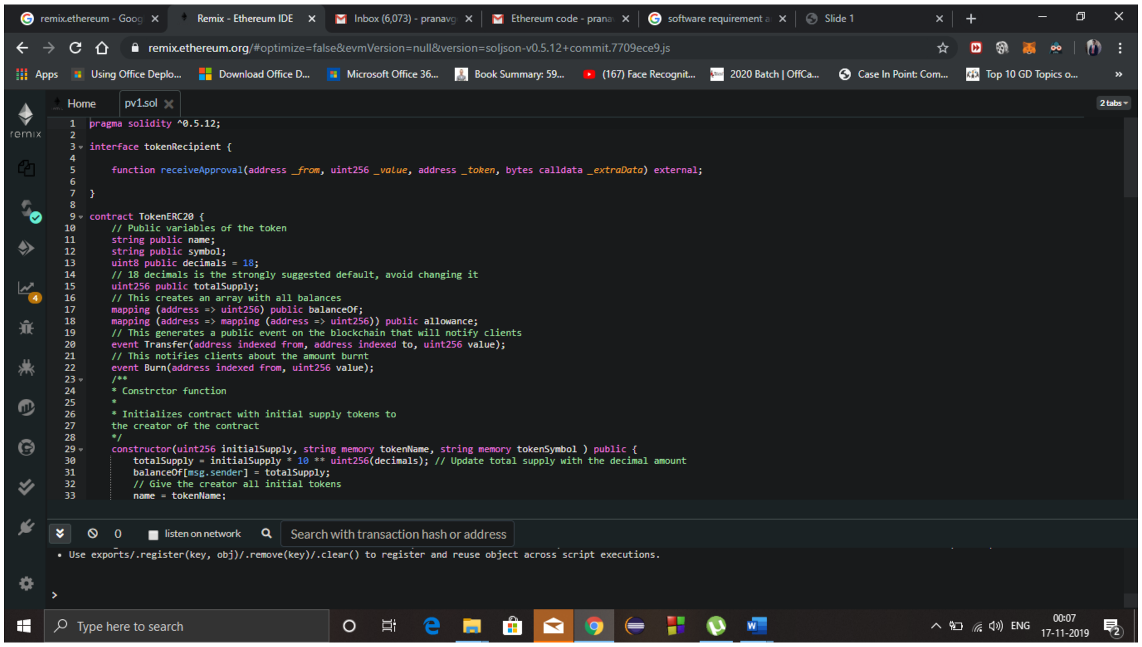
1,142,023 6 5s Rinkeby

Blockchain technology offers the potential to impact a wide range of industries. The most promising applications exist where transferring value or assets between parties is currently cumbersome, expensive and requires one or more centralized organization. A specific activity attracting significant interest is securities settlement, which today can involve multi-day clearing and settlement processes between multiple financial intermediaries. Certain financial services experts believe the financial services industry is on the verge of being disrupted: advances in innovative technologies such as blockchain are expected to transform the industry and its workforce by automating many of the activities currently performed by humans.

# CHAPTER 8
# OUTPUT SCREENS

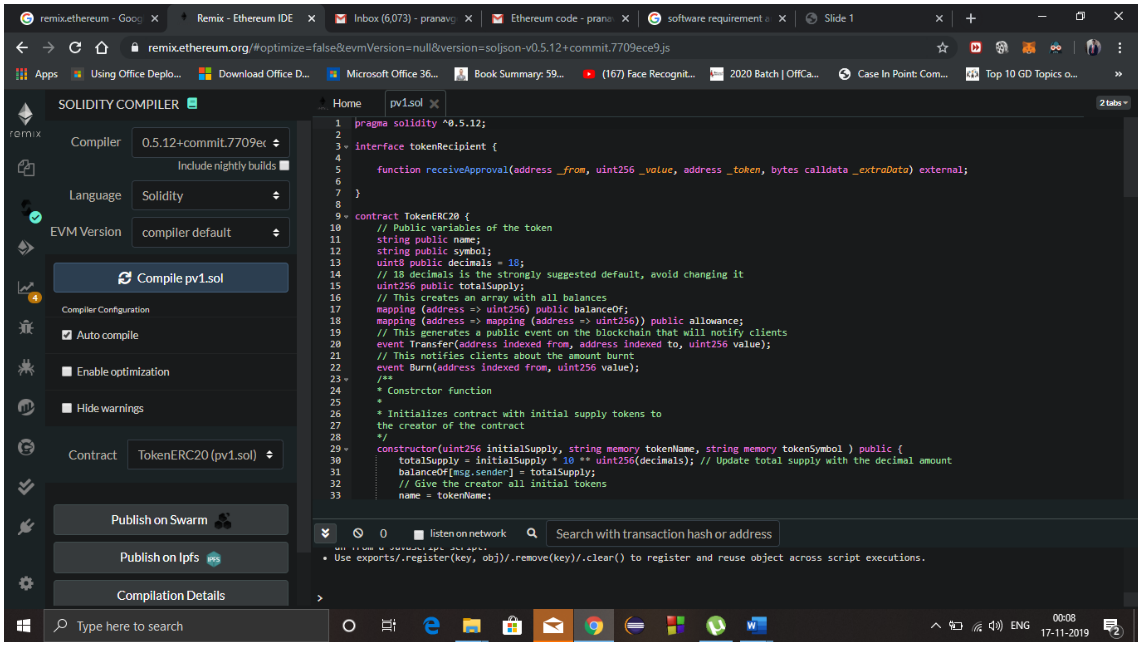


```
[vm] from:0xca3...a733c to:TokenERC20.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0x781...012bd
```



```
pragma solidity ^0.5.12;

interface tokenRecipient {

    function receiveApproval(address _from, uint256 _value, address _token, bytes calldata _extraData) external;

}

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;
    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;
    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);
    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);
    /**
    * Constrctor function
    *
    * Initializes contract with initial supply tokens to
    the creator of the contract
    */
    constructor(uint256 initialSupply, string memory tokenName, string memory tokenSymbol ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply with the decimal amount
        balanceOf[msg.sender] = totalSupply;
        // Give the creator all initial tokens
        name = tokenName;
```

```
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
```



SOLIDITY COMPILER

Compiler: 0.5.12+commit.7709ec

Include nightly builds

Language: Solidity

EVM Version: compiler default

Compile pv1.sol

Compiler Configuration

- [✓] Auto compile
- [ ] Enable optimization
- [ ] Hide warnings

Contract: TokenERC20 (pv1.sol)

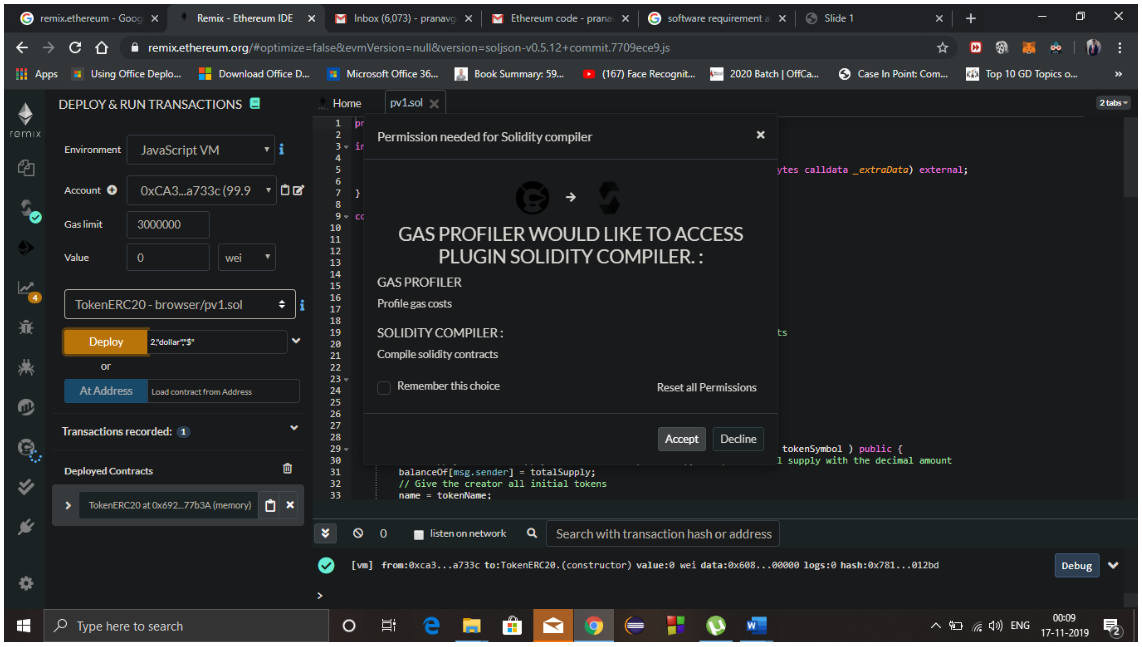
Publish on Swarm

Publish on Ipfs

Compilation Details

```
pragma solidity ^0.5.12;

interface tokenRecipient {

    function receiveApproval(address _from, uint256 _value, address _token, bytes calldata _extraData) external;

}

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;
    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;
    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);
    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);
    /**
    * Constrctor function
    *
    * Initializes contract with initial supply tokens to
    the creator of the contract
    */
    constructor(uint256 initialSupply, string memory tokenName, string memory tokenSymbol ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply with the decimal amount
        balanceOf[msg.sender] = totalSupply;
        // Give the creator all initial tokens
        name = tokenName;
```

```
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
```



DEPLOY & RUN TRANSACTIONS

Environment: JavaScript VM

Account: 0xCA3...a733c (99.9

Gas limit: 3000000

Value: 0 wei

TokenERC20 - browser/pv1.sol

Deploy: 2,"dollar","$"

or

At Address: Load contract from Address

Transactions recorded: 1

Deployed Contracts

TokenERC20 at 0x692...77b3A (memory)

Permission needed for Solidity compiler

GAS PROFILER WOULD LIKE TO ACCESS PLUGIN SOLIDITY COMPILER. :

GAS PROFILER

Profile gas costs

SOLIDITY COMPILER :

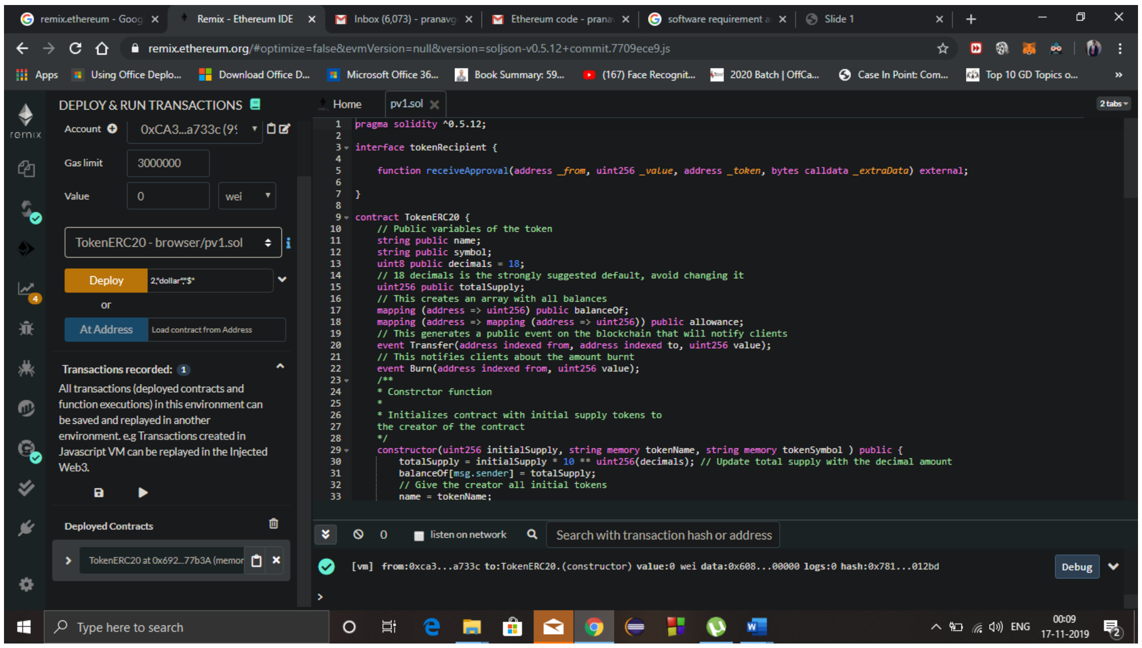
Compile solidity contracts

- [ ] Remember this choice

Reset all Permissions

Accept | Decline

```
[vm] from:0xca3...a733c to:TokenERC20.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0x781...012bd
```


DEPLOY & RUN TRANSACTIONS

Account: 0xCA3...a733c (9!

Gas limit: 3000000

Value: 0 wei

TokenERC20 - browser/pv1.sol

Deploy: 2,"dollar","$"

or

At Address: Load contract from Address

Transactions recorded: 1

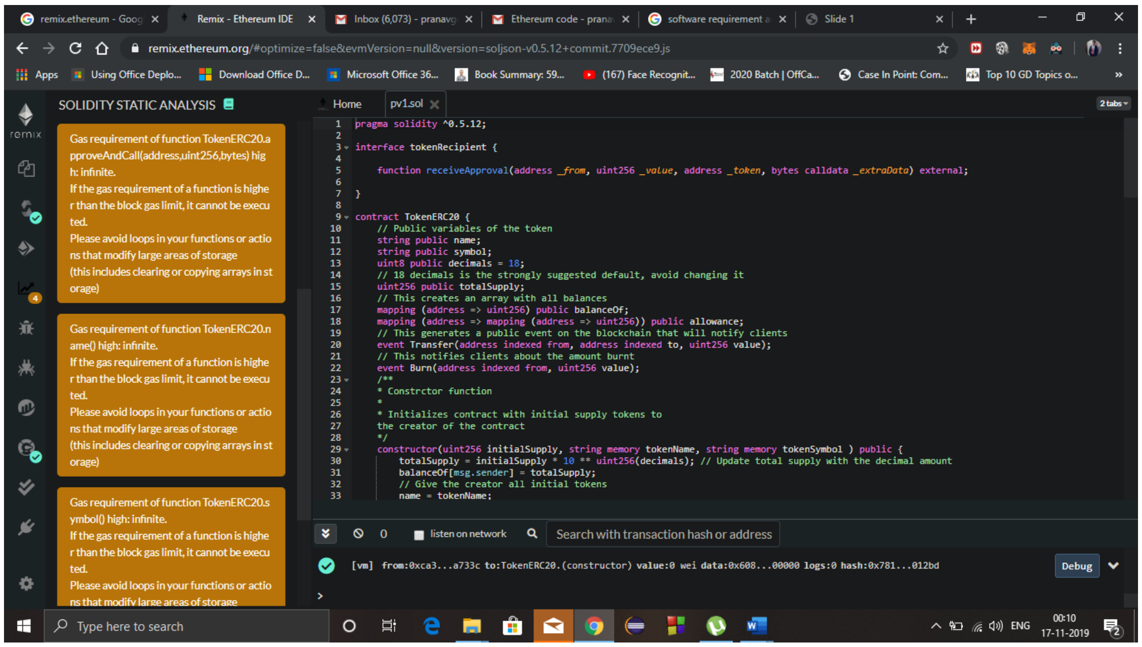
All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

Deployed Contracts

TokenERC20 at 0x692...77b3A (memor

```
pragma solidity ^0.5.12;

interface tokenRecipient {

    function receiveApproval(address _from, uint256 _value, address _token, bytes calldata _extraData) external;

}

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;
    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;
    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);
    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);
    /**
    * Constrctor function
    *
    * Initializes contract with initial supply tokens to
    the creator of the contract
    */
    constructor(uint256 initialSupply, string memory tokenName, string memory tokenSymbol ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply with the decimal amount
        balanceOf[msg.sender] = totalSupply;
        // Give the creator all initial tokens
        name = tokenName;
```

```
[vm] from:0xca3...a733c to:TokenERC20.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0x781...012bd
```


SOLIDITY STATIC ANALYSIS

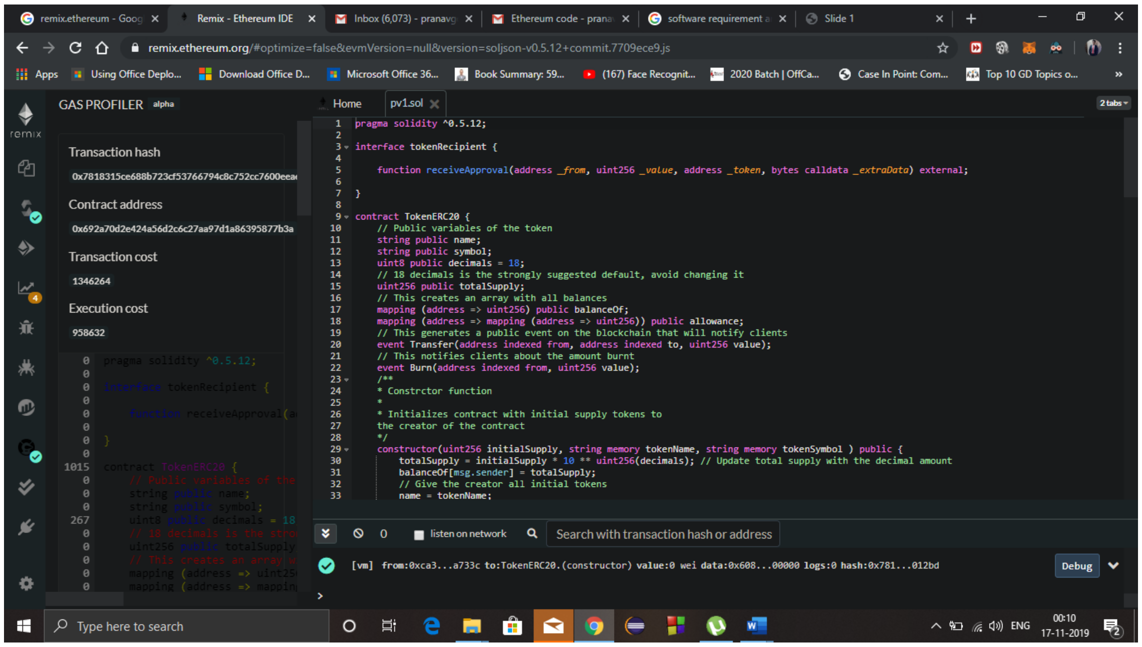
Gas requirement of function TokenERC20.approveAndCall(address,uint256,bytes) high: infinite.
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)

Gas requirement of function TokenERC20.name() high: infinite.
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)

Gas requirement of function TokenERC20.symbol() high: infinite.
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage

```
pragma solidity ^0.5.12;

interface tokenRecipient {

    function receiveApproval(address _from, uint256 _value, address _token, bytes calldata _extraData) external;

}

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;
    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;
    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);
    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);
    /**
    * Constrctor function
    *
    * Initializes contract with initial supply tokens to
    the creator of the contract
    */
    constructor(uint256 initialSupply, string memory tokenName, string memory tokenSymbol ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply with the decimal amount
        balanceOf[msg.sender] = totalSupply;
        // Give the creator all initial tokens
        name = tokenName;
```

```
[vm] from:0xca3...a733c to:TokenERC20.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0x781...012bd
```



GAS PROFILER alpha

Transaction hash
0x7818315ce688b723cf53766794c8c752cc7600eea

Contract address
0x692a70d2e424a56d2c6c27aa97d1a86395877b3a

Transaction cost
1346264

Execution cost
958632

```
   0  pragma solidity ^0.5.12;
   0
   0  interface tokenRecipient {
   0
   0      function receiveApproval(a
   0
   0  }
   0
1015  contract TokenERC20 {
   0      // Public variables of the
   0      string public name;
   0      string public symbol;
 267      uint8 public decimals = 18
   0      // 18 decimals is the stro
   0      uint256 public totalSupply
   0      // This creates an array w
   0      mapping (address => uint25
   0      mapping (address => mappin
```

pv1.sol

```solidity
pragma solidity ^0.5.12;

interface tokenRecipient {

    function receiveApproval(address _from, uint256 _value, address _token, bytes calldata _extraData) external;

}

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;
    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;
    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);
    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);
    /**
    * Constrctor function
    *
    * Initializes contract with initial supply tokens to
    the creator of the contract
    */
    constructor(uint256 initialSupply, string memory tokenName, string memory tokenSymbol ) public {
        totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply with the decimal amount
        balanceOf[msg.sender] = totalSupply;
        // Give the creator all initial tokens
        name = tokenName;
```

```
[vm] from:0xca3...a733c to:TokenERC20.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0x781...012bd
```

# CHAPTER 9

# FUTURE SCOPE

The blockchain protocol is a means of logging and verifying records that is transparent and distributed among users. Usually, donate are recorded, managed, counted and checked by a central authority. Blockchain-enabled donation system would empower to do these tasks themselves, by allowing them to hold a copy of the voting record. The historic record could then not be changed because other donation would see that the record differs from theirs. Illegitimate donation could not be added, because other donation would be able to scrutinise whether votes were compatible with the rules (perhaps because they have already been counted, or are not associated with a valid voter record).

The non-profit sector is painfully slow in terms of innovation: the costs are enormous and are borne by those most in need. In the US alone, $370 billion donations flow every year into over 1.5 million charities. Online donations, however, account only for 5% of total transactions, while the use of cryptocurrencies is almost non-existent. Yet, by 2025 10% of GDP is expected to be generated on blockchain and cryptocurrencies already reached a total market capitalization of $200 billion.

The issues plaguing charities worldwide are innumerable and cannot be countered by relying on traditional methods. Nor can we afford to maintain the status quo or harbour doubts about the impact new technologies can actually bring about: wars, climate change, natural disasters, immigration and the erosion of the welfare state render the resources and the services provided by non-profits ever more in demand by billions of people and animals worldwide. The solution is, for once, simple and close at hand: there needs to be widespread adoption of blockchain technology by charities and donors alike, encouraged by clearer and friendly cross-border regulation and, eventually, a blockchain ledger to for all transactions—for all currencies.

# CHAPTER 10
# REFERENCES

1. http://solidity.readthedocs.io
2. https://www.coindesk.com
3. https://geth.ethereum.org
4. https://codeburst.io
5. file:///C:/Users/gautam/Downloads/Blockchain_Based_Smart_Contracts_A_Systematic_Mapp.pdf
6. https://www.researchgate.net/publication/326752872_Smart_Contracts_A_Primer
7. https://www.iuk.fraunhofer.de/content/dam/iuk/en/docs/Fraunhofer-Paper_Blockchain-and-Smart-Contracts_EN.pdf