CROWDFUNDING USING BLOCKCHAIN

MINOR PROJECT REPORT

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING
by

Anuj Arora	**Rhythm Jayee**	**Jatin Malhotra**
Enrollment No:	**Enrollment No:**	**Enrollment No:**
44814802718	**75114802718**	**41914802718**

Guided by

Ms. Mini Agarwal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY
(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)

(Session 2021-2022)

ABSTRACT

Crowd funding is an online money-raising approach that began as a way for people to donate a little amount of money to help inventive people fund their ventures. People may invest in pioneering by using crowdfunding. Businesses can use an intermediate medium or platform to connect with each other. The problem with the present method of crowd financing is through a third-party channel. Don't guarantee the money given by the investor. The project and the investor have no control over the money contributed. This paper offers a blockchain-based crowdfunding infrastructure that can provide a private, secure, and decentralized crowdfunding path. The major goal of this paper is to allow investors to successfully contribute to any project by establishing smart contracts that allow contributors to have control over their money and project creators and investors to effectively make and reserve funds for the project.

Investors can contribute to any project they are interested in, and if the initiative is successful, they can profit. Many crowdfunding sites exist these days, and they accept large sums of money from investors and contributors and then leave them with unfulfilled promises. The traditional manner of dealing with company finance is being disrupted by blockchain-based crowdfunding. In general, when people require money to start a firm, they must first create a strategy, statistical surveys, and models, and then offer their ideas to people or organizations.

Banks, angel capitalists, and venture capital firms were among the subsidized sources. The current crowdfunding concept is based on three types of on-screen characters: the task initiator who offers the idea or venture to be financed, individuals or investors who invest in the idea, and a platform that connects these two characters to make the venture a success. It is used to fund a wide range of start-ups and unique concepts, such as new activities, medical breakthroughs, travel, and social entrepreneurship initiatives

ACKNOWLEDGEMENT

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our respected guide Ms. Mini Agarwal (Assistant Professor, CSE) MAIT Delhi, for her valuable guidance, encouragement and help for completing this work. Her useful suggestions for this project and cooperative behavior are sincerely acknowledged.

We also wish to express our indebtedness to our parents as well as our family members whose blessings and support always helped us to face the challenges ahead.

Place: Delhi

Students :
Anuj Arora(44814802718),
Rhythm Jayee(75114802718),
Jatin Malhotra(41914802718)

Date: 10 December 2021

TABLE OF CONTENTS

1. Abstract
2. Acknowledgement
3. Introduction
4. Technology Used
5. Methodology
6. Results & Discussions
7. Architecture
8. Implementation
9. Code Snippets
10. Outputs
11. Conclusion
12. References
13. Certificates

INTRODUCTION

A blockchain is a growing list of records, called blocks that are linked together using cryptography. It's also described as a "trustless and fully decentralized peer-to-peer immutable data storage" that is spread over a network of participants often referred to as nodes. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree). The timestamp proves that the transaction data existed when the block was published in order to get into its hash. As blocks each contain information about the block previous to it, they form a chain, with each additional block reinforcing the ones before it. Therefore, blockchains are resistant to modification of their data because once recorded, the data in any given block cannot be altered retroactively without altering all subsequent blocks.

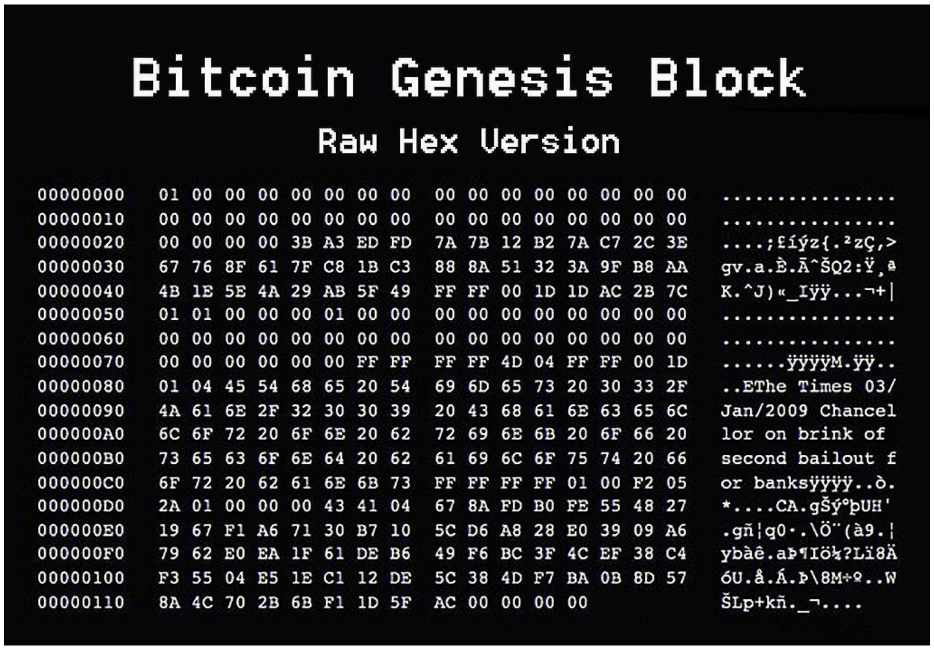
Blockchains are typically managed by a peer-to-peer network for use as a publicly distributed ledger, where nodes collectively adhere to a protocol to communicate and validate new blocks. Although blockchain records are not unalterable as forks are possible, blockchains may be considered secure by design and exemplify a distributed computing system with high Byzantine fault tolerance.

History

Cryptographer David Chaum first proposed a blockchain-like protocol in his 1982 dissertation "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups." Further work on a cryptographically secured chain of blocks was described in 1991 by Stuart Haber and W. Scott Stornetta. They wanted to implement a system wherein document timestamps could not be tampered with. In 1992, Haber, Stornetta, and Dave Bayer incorporated Merkle trees to the design, which improved its efficiency by allowing several document certificates to be collected into one block.

The first blockchain was conceptualized by a person (or group of people) known as Satoshi Nakamoto in 2008. Nakamoto improved the design in an important way using a Hashcash-like method to timestamp blocks without requiring them to be signed by a trusted party and introducing a difficulty parameter to stabilize rate with which blocks are added to the chain. The design was implemented the following year by Nakamoto as a core component of the cryptocurrency bitcoin, where it serves as the public ledger for all transactions on the network.

In August 2014, the bitcoin blockchain file size, containing records of all transactions that have occurred on the network, reached 20 GB (gigabytes). In January 2015, the size had grown to almost 30 GB, and from January 2016 to January 2017, the bitcoin blockchain grew from 50 GB to 100 GB in size. The ledger size had exceeded 200 GB by early 2020.



Bitcoin Genesis Block

Raw Hex Version

```
00000000   01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000010   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000020   00 00 00 00 3B A3 ED FD  7A 7B 12 B2 7A C7 2C 3E   ....;£íýz{.²zÇ,>
00000030   67 76 8F 61 7F C8 1B C3  88 8A 51 32 3A 9F B8 AA   gv.a.È.Ã^ŠQ2:Ÿ¸ª
00000040   4B 1E 5E 4A 29 AB 5F 49  FF FF 00 1D 1D AC 2B 7C   K.^J)«_Iÿÿ...¬+|
00000050   01 01 00 00 00 01 00 00  00 00 00 00 00 00 00 00   ................
00000060   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000070   00 00 00 00 00 00 FF FF  FF FF 4D 04 FF FF 00 1D   ......ÿÿÿÿM.ÿÿ..
00000080   01 04 45 54 68 65 20 54  69 6D 65 73 20 30 33 2F   ..EThe Times 03/
00000090   4A 61 6E 2F 32 30 30 39  20 43 68 61 6E 63 65 6C   Jan/2009 Chancel
000000A0   6C 6F 72 20 6F 6E 20 62  72 69 6E 6B 20 6F 66 20   lor on brink of
000000B0   73 65 63 6F 6E 64 20 62  61 69 6C 6F 75 74 20 66   second bailout f
000000C0   6F 72 20 62 61 6E 6B 73  FF FF FF FF 01 00 F2 05   or banksÿÿÿÿ..ò.
000000D0   2A 01 00 00 00 43 41 04  67 8A FD B0 FE 55 48 27   *....CA.gŠý°þUH'
000000E0   19 67 F1 A6 71 30 B7 10  5C D6 A8 28 E0 39 09 A6   .gñ¦q0·.\Ö¨(à9.¦
000000F0   79 62 E0 EA 1F 61 DE B6  49 F6 BC 3F 4C EF 38 C4   ybàê.aÞ¶Iö¼?Lï8Ä
00000100   F3 55 04 E5 1E C1 12 DE  5C 38 4D F7 BA 0B 8D 57   óU.å.Á.Þ\8M÷º..W
00000110   8A 4C 70 2B 6B F1 1D 5F  AC 00 00 00 00            ŠLp+kñ._¬....
```

Satoshi Nakamoto message embedded in the coinbase of the first block

The Myth of Satoshi Nakamoto

Satoshi Nakamoto is the name used by the presumed pseudonymous person or persons who developed bitcoin, authored the bitcoin white paper, and created and deployed bitcoin's original reference implementation. As part of the implementation, Nakamoto also devised the first blockchain database. Nakamoto was active in the development of bitcoin up until December 2010. Many people have claimed, or have been claimed, to be Nakamoto.

Nakamoto stated that work on the writing of the code for bitcoin began in 2007. On 18 August 2008, he or a colleague registered the domain name bitcoin.org, and created a web site at that address. On 31 October, Nakamoto published a white paper on the cryptography mailing list at metzdowd.com describing a digital cryptocurrency, titled "Bitcoin: A Peer-to-Peer Electronic Cash System".

On 9 January 2009, Nakamoto released version 0.1 of the bitcoin software on SourceForge, and launched the network by defining the genesis block of bitcoin (block number 0), which had a reward of 50 bitcoins. Embedded in the coinbase transaction of this block is the text: “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks”, citing a headline in the UK newspaper The Times published on that date. This note has been interpreted as both a timestamp and a derisive comment on the instability caused by fractional-reserve banking.

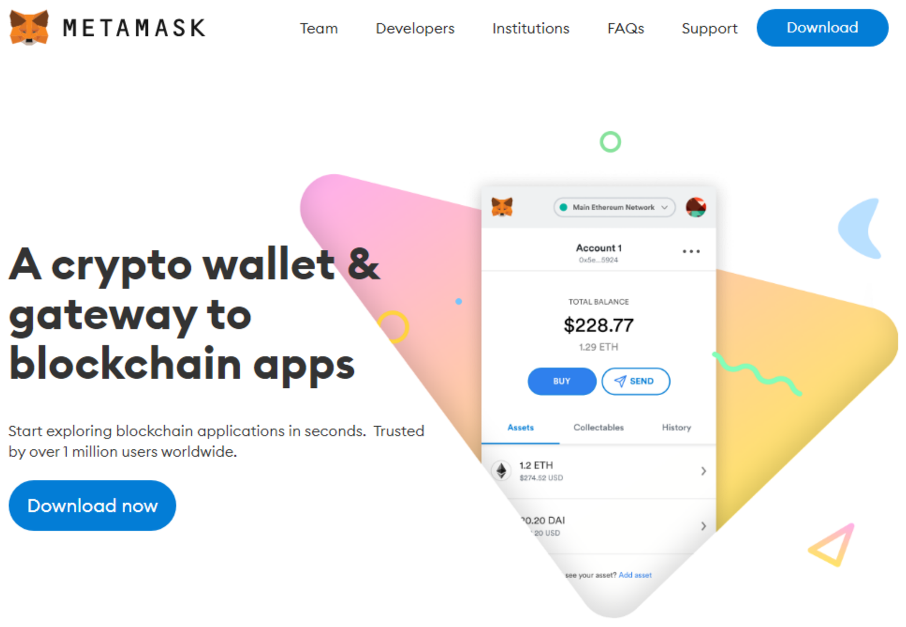
Nakamoto continued to collaborate with other developers on the bitcoin software until mid-2010, making all modifications to the source code himself. He then gave control of the source code repository and network alert key to Gavin Andresen, transferred several related domains to various prominent members of the bitcoin community, and stopped his recognized involvement in the project.

In 2021 a monument was announced in honor of Nakamoto in Budapest, Hungary for his work on Bitcoin and cryptocurrencies.

TOOLS & TECHNOLOGY USED

Installing MetaMask

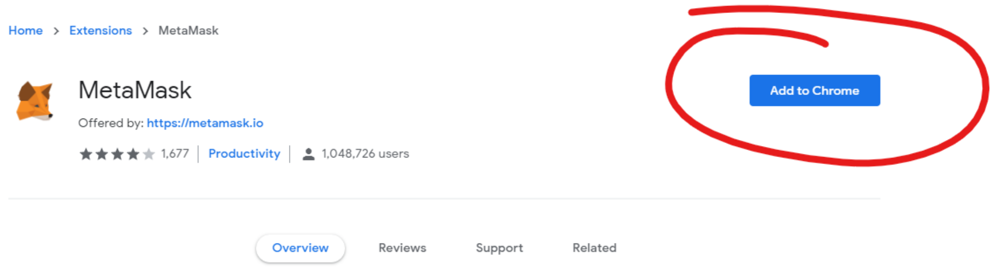
MetaMask is a browser plugin which can securely store private keys and connect to different blockchains.



A crypto wallet & gateway to blockchain apps

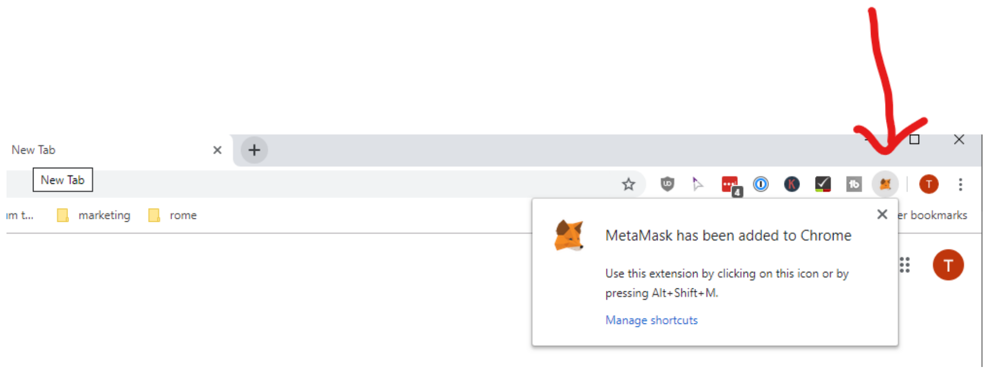
Start exploring blockchain applications in seconds. Trusted by over 1 million users worldwide.

Open https://metamask.io and download the plugin for your browser



Offered by: https://metamask.io

1,677 | Productivity | 1,048,726 users


MetaMask has been added to Chrome

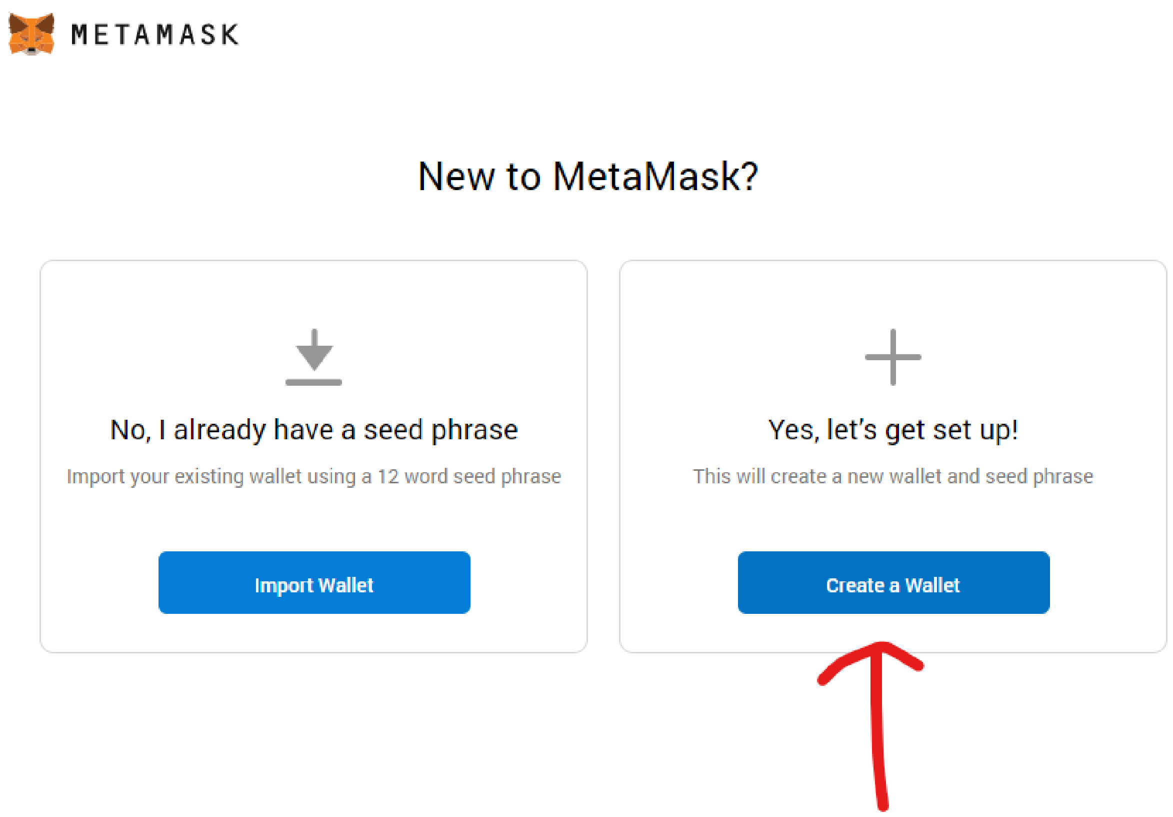
Use this extension by clicking on this icon or by pressing Alt+Shift+M.

Manage shortcuts

Setup MetaMask

When you install MetaMask, then it will automatically open up a “setup” page.

Hit “Begin” and walk through the setup-wizard. Let’s create a new Wallet!


METAMASK

New to MetaMask?

No, I already have a seed phrase

Import your existing wallet using a 12 word seed phrase

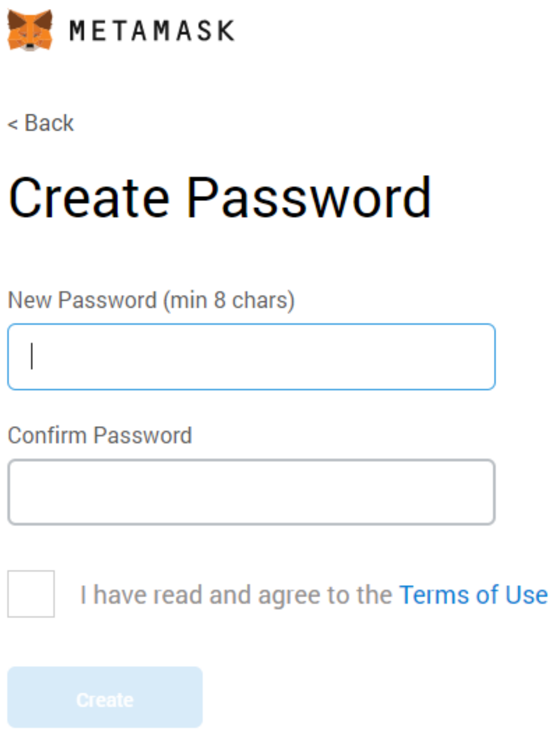
Import Wallet

Yes, let’s get set up!

This will create a new wallet and seed phrase

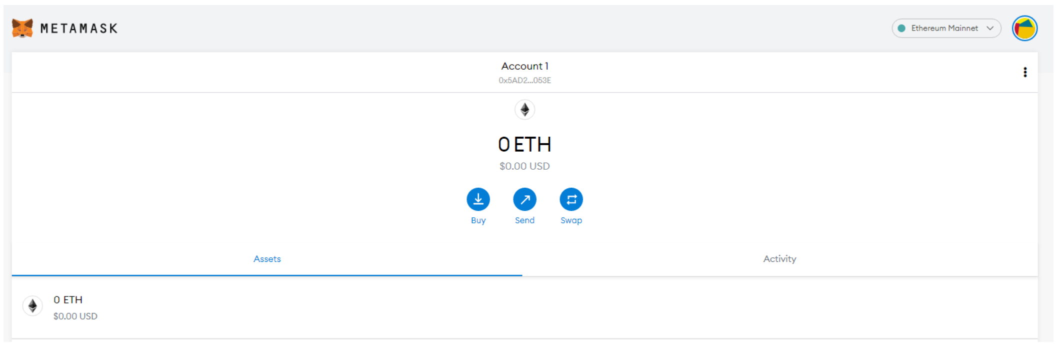
Create a Wallet

Create a new strong password. This password is used to encrypt your private keys. What private keys are exactly is discussed in a later section of the course, suffice to say though, they give access to all your Ether. So, better have a strong password here:


METAMASK
< Back
Create Password
New Password (min 8 chars)
Confirm Password
I have read and agree to the Terms of Use
Create

Backup Phrase: It would be better to safely store the secret phrase

Final Screen

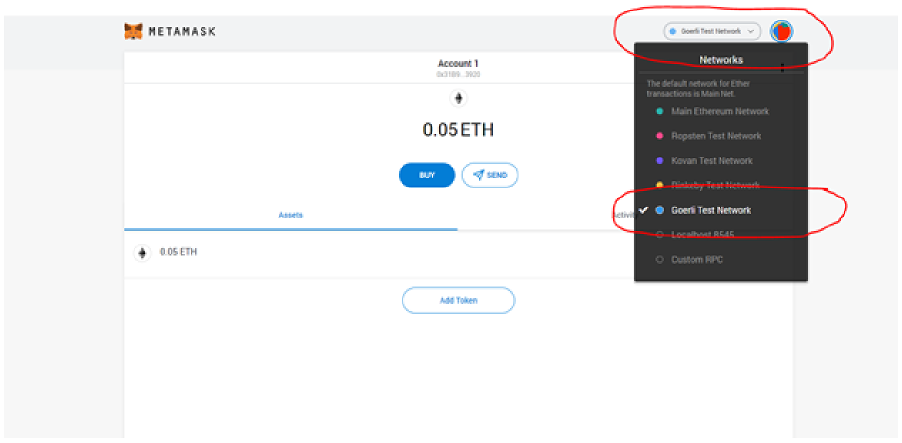


Get Free Ether

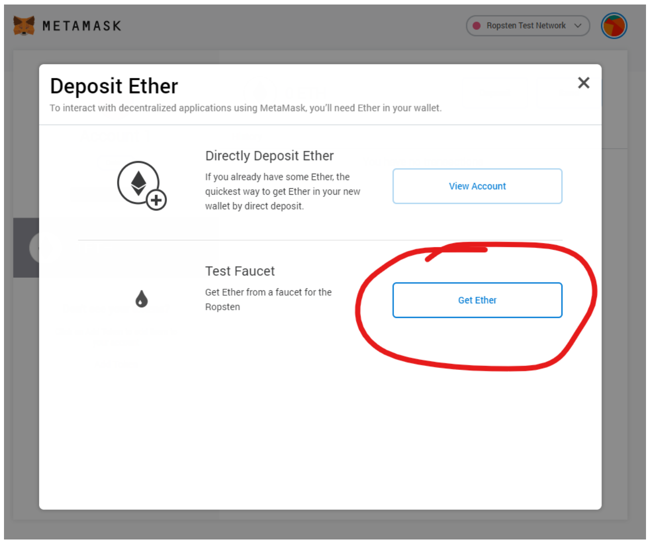
If you have never worked with blockchains before, then the first confusion that you will encounter is: There is not one blockchain, but many different blockchains. I am talking about Ethereum Blockchains. It’s like having different databases. But only one is considered the “Main” Database, or “Mainnet”. There are also other blockchains, for testing different aspects. Each of those usually have a name and a specific network and chain id. There is no central list of them, because everyone can open their own blockchain.

In this, we will use either Ropsten or Görli to get Test-Ether and start a transaction.

Get Görli Test-Ether. Switch the network to Goerli.



Hit “BUY”. Click on “Get Ether”



Deposit Ether

To interact with decentralized applications using MetaMask, you'll need Ether in your wallet.

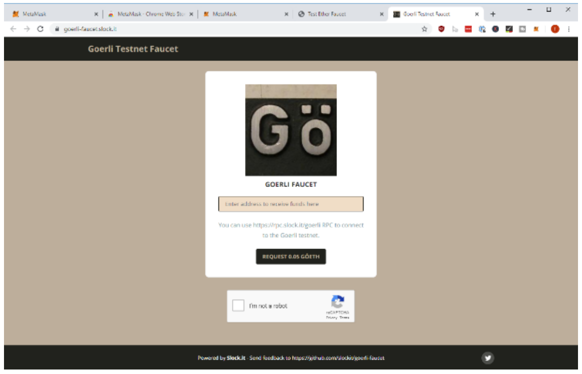
Directly Deposit Ether

If you already have some Ether, the quickest way to get Ether in your new wallet by direct deposit.

Test Faucet

Get Ether from a faucet for the Ropsten

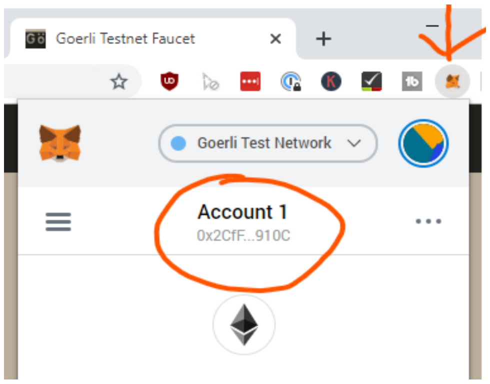
A new website should open up. That's the faucet to get Ether. A Faucet is like a "get free Ether" – site. The Ethers are having no value, they are running under a "test" Blockchain, but they are great for getting your feet wet with transactions and how Wallets work.



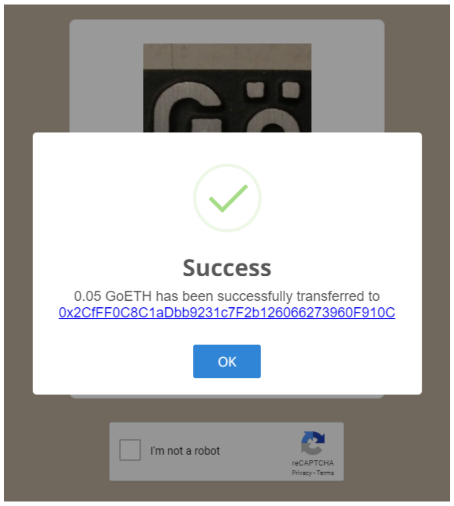
You can use https://rpc.slock.it/goerli RPC to connect to the Goerli testnet.

Powered by Slock.it - Send feedback to https://github.com/slockit/goerli-faucet

Copy your Address from MetaMask by clicking directly on the address:



Paste the address and click on get ether, wait for a while and this window should pop up



Success

0.05 GoETH has been successfully transferred to 0x2CfFF0C8C1aDbb9231c7F2b126066273960F910C

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). Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

When deploying contracts, you should use the latest released version of Solidity. This is because breaking changes as well as new features and bug fixes are introduced regularly. We currently use a 0.x version number to indicate this fast pace of change.

Data Types

Solidity value types include boolean, integers, fixed point numbers, addresses, contract types, fixed-size byte arrays, rational and integer literals, and enums. Reference types such as arrays and structs can be stored in these options: memory, storage , and calldata .

Booleans of the Solidity value types can be either true or false. The boolean is defined with the bool keyword.

Integers: There are two main Solidity types of integers of differing sizes:

- int – signed integers.
- uint – unsigned integers.

Speaking of size, to specify it, you have keywords such as uint8 up to uint256, that is, of 8 to 256 bits. The simple uint and int are similar to uint256 and int256, respectively.

Fixed Point Numbers: There are two types of fixed point numbers:

- fixed – signed fixed point number.
- ufixed – unsigned fixed point number.

This value type also can be declared with keywords such as ufixedMxN and fixedMxN. The M represents the amount of bits that the type takes, with N representing the number of decimal points that are available. M has to be divisible by 8, and a number from 8 to 256. N has to be a value between 0 and 80, also being inclusive.

Addresses: The address Solidity value type has two similar kinds:

- Address holds a 20-byte value (size of an Ethereum address).
- Address payable is the same as address, but has transfer and send members.

Implicit conversions of addresses:

- From address payable to address: allowed.
- From address to address payable: not allowed. This type of conversion is only possible with intermediate conversion to uint160.
- Address literals can be converted to address payable.

Explicit conversions to and from address are permitted for integers, integer literals, contact types and bytes20. However, Solidity prevents the conversions of address payable(x). The address(x) can be converted to address payable in cases when x is of integer, fixed bytes type, or a literal or a contract that has a payable fallback function. When x is a contract without the payable fallback function, the address(x) is of type address.

Members of Address

The balance property queries the balance of an address, while Ether can be sent to addresses with the transfer function.mThe transfer function queries the balance of an address by applying the property balance and sending Ether (in units of wei) to a payable address:

When the balance of current contracts is not sufficient enough or when the receiver rejects the transfer, the transfer function fails. It reverts on failure.

Gain more direct control over encoding or interface with contracts (not adhere to the ABI) with call, delegatecall and staticcall functions.

They accept one bytes memory parameter, deliver the success condition as a oolean and the returned data. For encoding of structured data, use abi.encode, abi.encodePacked, abi.encodeWithSelector and abi.encodeWithSignature:

Solidity Operators

Arithmetic Operator

Operator	Meaning	Example
+	Add two operands or unary plus	x+y+2

-	Subtract right operand from the left or unary minus	x-y-2
*	Multiply two operands	x*y
/	Divide left operand by the right one (always result into float	x/y
%	Modulus- remainder of the division of left operand by the right	x%y (remainder of x/y)
//	Floor division- division that results into whole number adjusted to the left in the number line	x//y
**	Exponent- left operand raised to the power of right	x**y(x to the power y)

Table 2.1

Comparison Operator

>	Greater that – True if left operand is greater than the right	x > y
<	Less that – True if left operand is less than the right	x < y
==	Equal to – True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to – True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to – True if left operand is less than or equal to the right	x <= y

Table 2.2

Logical Operator

Operator	Meaning	Example
And	True if both the operands are True	X and Y
Or	True if either of the operands are True	X or Y
Not	True if operand is False (complements the operand)	Not x

Table 2.3

Contract Types

All contracts define their type. It is possible to implicitly convert contracts to the contracts they inherit from. Contracts can be changed from and to address with explicit conversion.

The explicit conversion from and to address payable is permitted only when the contract type contains the payable fallback function.

Fixed-size Byte Arrays

The value types bytes1, bytes2, bytes3, …, bytes32 contain a sequence of bytes (from 1 to 32).

Dynamically-Sized Byte Array

- bytes are dynamically-sized byte array and is not one of the Solidity value types.
- string is a dynamically-sized UTF-8-encoded string and is not a value type.

Enums

Enums generate user-defined Solidity types. The explicit conversion is possible to and from all integer types, but the implicit conversion is not.

Since enum types are not part of the ABI, the signature of getChoice will automatically be modified to getChoice() returns (uint8) for all matters external to Solidity.

The integer type used is just large enough to hold all enum values, i.e., if you have more than 256 values, uint16 will be used and so on.

Function Types

Solidity function types represent types of functions. Function type variables can be assigned from functions. To designate functions to and return functions from function calls, use function parameters.

Members that the public and external functions can have:

- .selector: retrieves the ABI function selector.
- .gas(uint): retrieves a callable function object. After that function is called, it sends the indicated amount of gas to the target function.
- value(uint): retrieves a callable function object. After that function is called, it sends the specified amount of wei to the target function.

Call internal functions only on the current contract since they cannot be completed outside the context of the current contract. External functions contain signatures of functions and addresses: such functions can be passed and returned from external function calls.

Function types are internal by default. Therefore, it is possible to remove the internal keyword. Remember that this is only applicable for function types.

Reference Types

Solidity reference types have to be handled with more caution than value types. It is crucial to clearly indicate the data area where the reference type is stored: memory, storage or calldata. Reference type values are manipulated via more than one different name.

Data Location and Assignment Behaviour

Every reference type contains information on where it is stored. There are three possible options: memory, storage, and calldata. The set location is important for semantics of assignments, not only for the persistence of data:

- Assignments between storage and memory (or from calldata) always generate an independent copy.
- Assignments from memory to memory only create references. This means that changes to one memory variable are also visible in all other memory variables that refer to the same data.

- Assignments from storage to a local storage variable also only assign a reference.
- All other assignments to storage always copy. Examples for this case are assignments to state variables or to members of local variables of storage struct type, even if the local variable itself is just a reference.

Arrays

Arrays can have fixed or dynamic sizes for the compiling process. The array with fixed size k and element type T is combined as T[k]. The dynamically-sized array is T[]. An array consisting of 6 dynamic arrays of uint looks like this: uint[] [6]. By default in Solidity, x[3] array consists of three elements of type despite the fact that it can be an array.

Allocating Memory Arrays :The keyword creates arrays with a runtime-dependent length in memory.

Array Literals

A list of one or multiple expressions, separated by commas and placed in square brackets ([...]) is an array literal. It is a statically-sized memory array. The following example shows that the type of [1, 2, 3] is uint8[3] memory. Since every constant is of uint8 type, the first element must be converted to uint before it can be uint[3] memory.

Array Members

Arrays have these members:

- length – indicates the number of elements. For memory arrays, the length is fixed after they are generated. However, it can be dynamic and can rely on runtime parameters. The length is set to dynamically-sized arrays to change their size.
- push – attaches an element at the end of the dynamic storage arrays and bytes (not string). The newly-added element is zero-initialized.
- pop removes an element at the end of the dynamic storage arrays and bytes (not string).

Structs

Solidity allows us to define new types in the form of structs.

Mapping Types

Syntax of (_KeyType =>_ValueType) defines the mapping types.

- The _KeyType can be any elementary type (plus bytes and string). However, contract types, enums, mappings, structs, and any array type apart are not permitted.
- _ValueType accepts any type, mappings as well.
- It is possible to set state variables of mapping Solidity type as public and the language generates a getter. Then, the _KeyType is a parameter for the getter.
- In cases when _ValueType is a value type or a struct, the getter returns _ValueType. When it is an array or mapping, the getter has one parameter for every _KeyType, recursively.

Operators Involving Lvalues

The a is an Lvalue (for instance, a variable or something that can be assigned to). It has operators as shorthands:

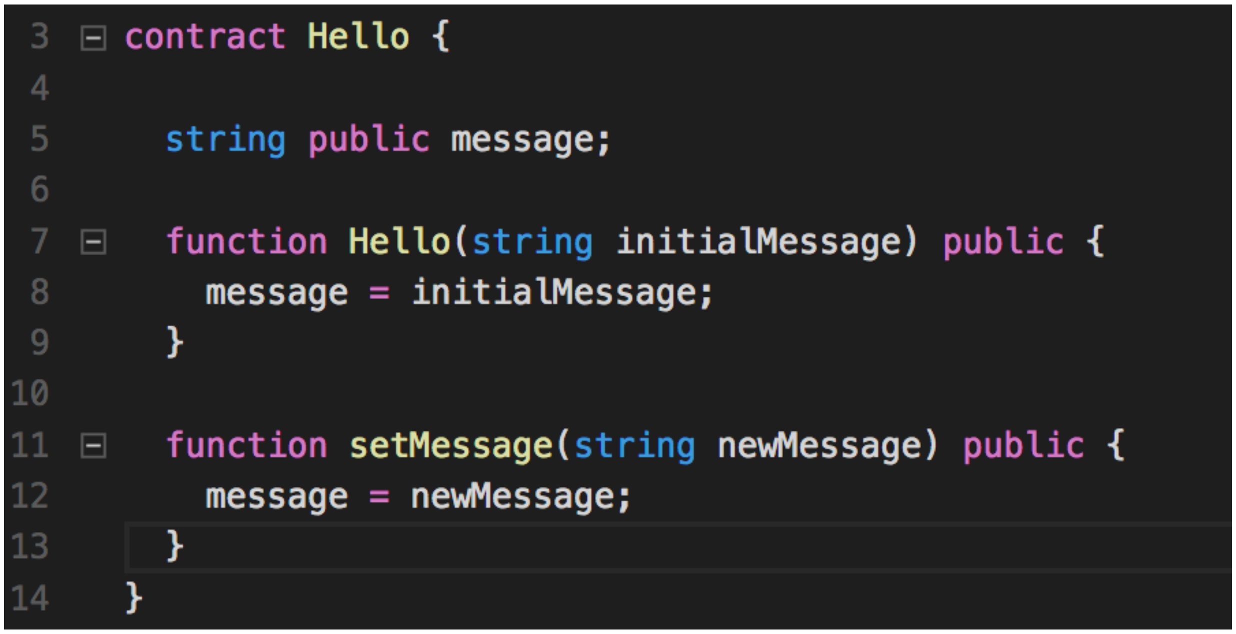
- a += e is the same as a= a + e. The operators -=, *=, /=, %=, |=, &= and ^= are defined accordingly.
- a++ and a–is the same as a += 1 / a -= 1 but the expression still has the previous a value.
- --a and ++a work the same on a but return the value after the change.

Delete

- The delete a sets the initial value for the type to a. I.e. for integers it is equivalent to a = 0.
- It is possible to apply it on arrays. It will set a dynamic array of length zero or static array of the same length with all elements assigned their initial value.
- delete a[x] will remove an element at the specified array index (won't change other items and length). This results in a gap in an array.

● It sets a struct with all members reset. The a value after delete a is the same as delete does not influence mappings. By deleting a struct, you reset all members that are not related to mappings.

Code in Solidity



```
contract Hello {

  string public message;

  function Hello(string initialMessage) public {
    message = initialMessage;
  }

  function setMessage(string newMessage) public {
    message = newMessage;
  }
}
```

Fig 2.6

TOOLS

Remix- Ethereum IDE

Introduction

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.

Remix IDE is part of the Remix Project which is a platform for development tools that use a plugin architecture. It encompasses sub-projects including Remix Plugin Engine, Remix Libs, and of course Remix-IDE. Remix IDE is a powerful open source tool that helps you write Solidity contracts straight from the browser.It is written in JavaScript and supports both usage in the browser, in the browser but runs locally and in a desktop version.

Remix IDE has modules for testing, debugging and deploying of smart contracts and much more.

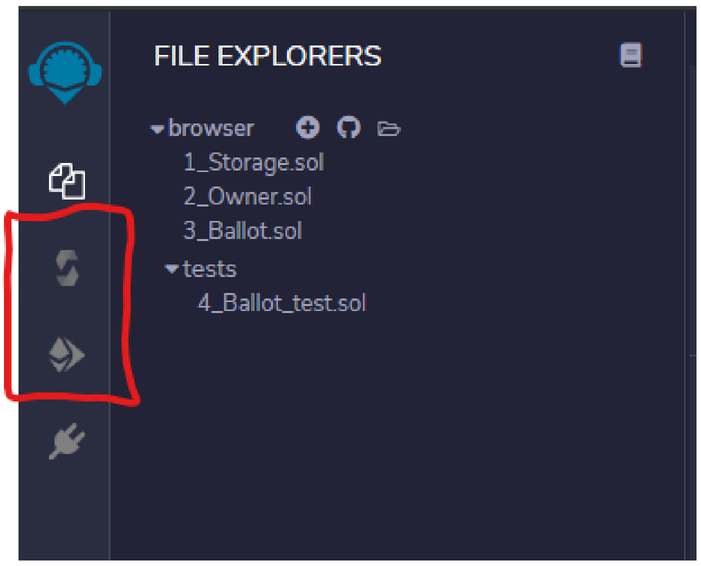
Remix-IDE is available at remix.ethereum.org and more information can be found in these docs. Our IDE tool is available at our GitHub repository.

PLUGINS

Remix is built with a pluggable architecture. All functions are done via plugins. The Compiler is a plugin, the blockchain connection is a plugin, the debugging functionality is a plugin and there are a lot of other plugins that might be useful.

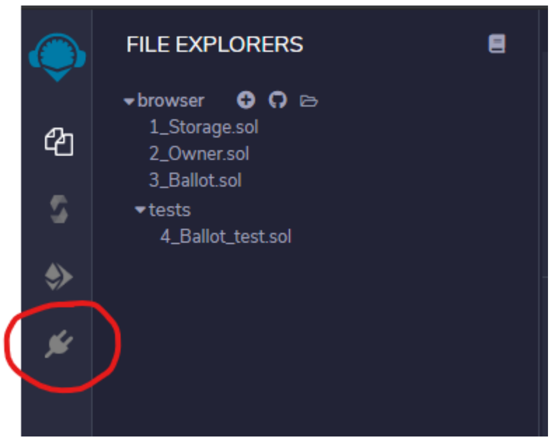
What you need in the next few chapters are

- The Solidity compiler
- The “Deploy & Run Transactions” Plugin’

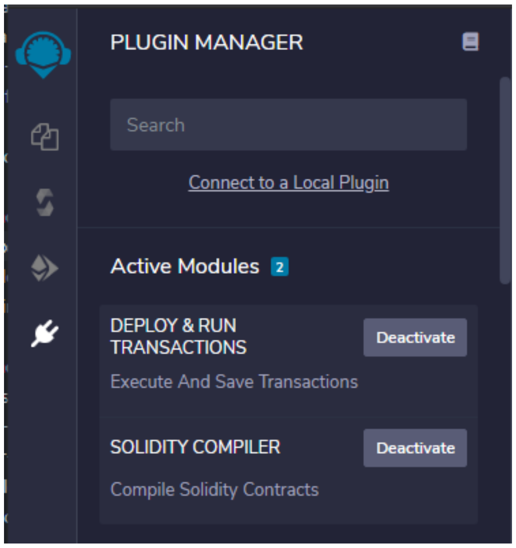


Enable Plugin

If the plugins are not showing up yet, then click on the plugin symbol and enable them:



And find the two plugins and activate them:


PLUGIN MANAGER

Search

Connect to a Local Plugin

Active Modules 2

DEPLOY & RUN TRANSACTIONS — Deactivate

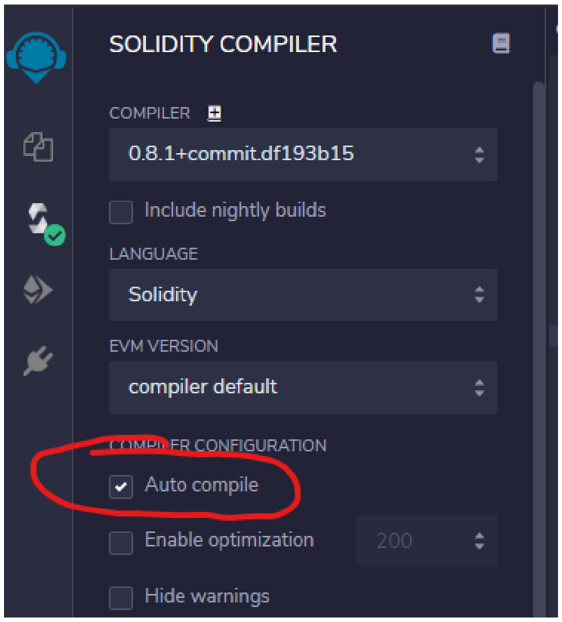
Execute And Save Transactions

SOLIDITY COMPILER — Deactivate

Compile Solidity Contracts

Configure the Compiler

The compiler will normally switch automatically based on the pragma line in your solidity files. But you can set a specific version, if necessary.


SOLIDITY COMPILER

COMPILER: 0.8.1+commit.df193b15

☐ Include nightly builds

LANGUAGE: Solidity

EVM VERSION: compiler default

COMPILER CONFIGURATION

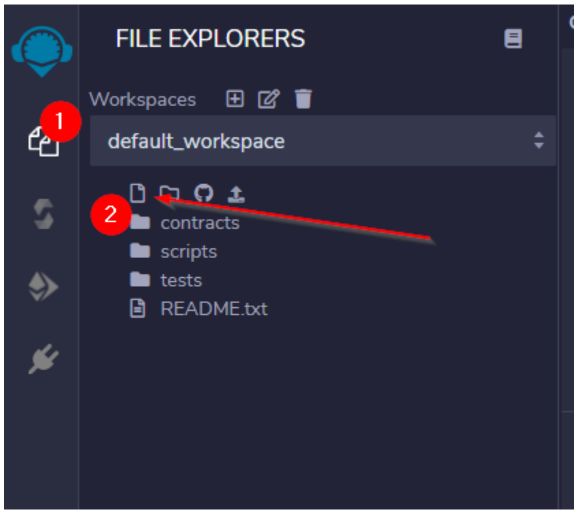
☑ Auto compile

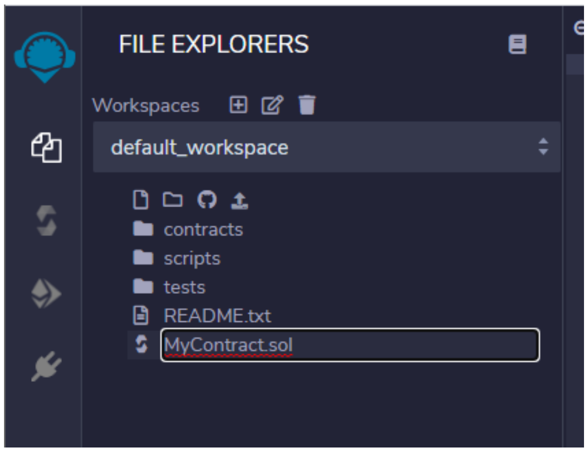
☐ Enable optimization 200

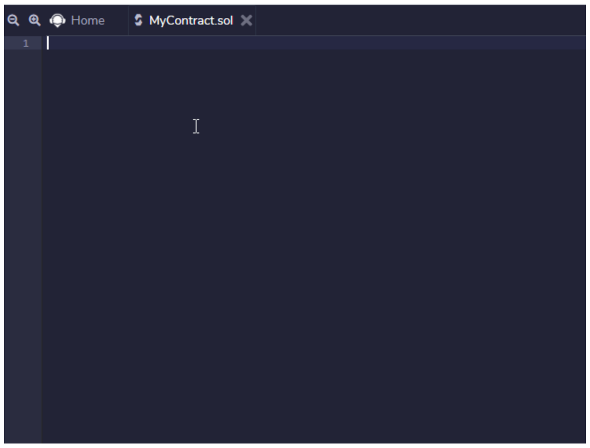
☐ Hide warnings

Create a New File

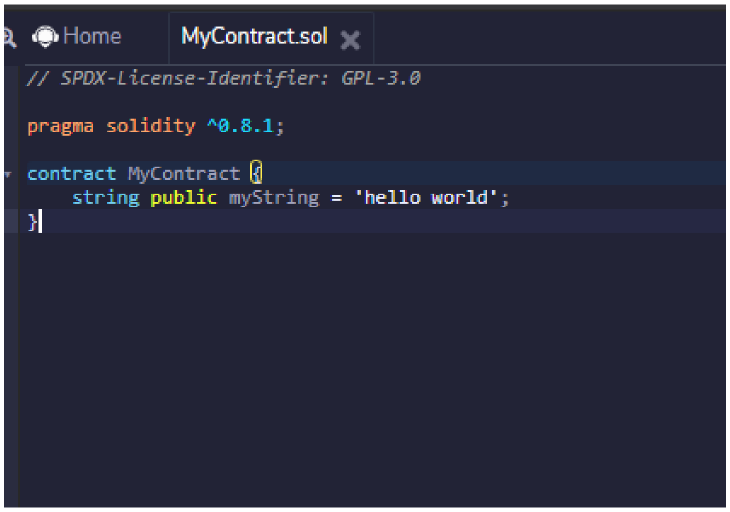
Click on the plus icon in the code-editor and create a new file, name it “MyContract.sol”. The sol-extension stands for Solidity.


FILE EXPLORERS
Workspaces
default_workspace
1
2
contracts
scripts
tests
README.txt


FILE EXPLORERS
Workspaces
default_workspace
contracts
scripts
tests
README.txt
MyContract.sol


Home
MyContract.sol

Add Code to the File



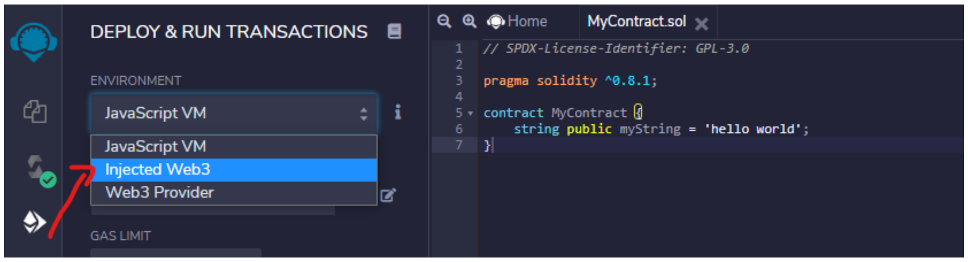
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

contract MyContract {
    string public myString = 'hello world';
}
```

Deploy Contract

Now it's time to deploy our Smart Contract. We will do this to a real blockchain.

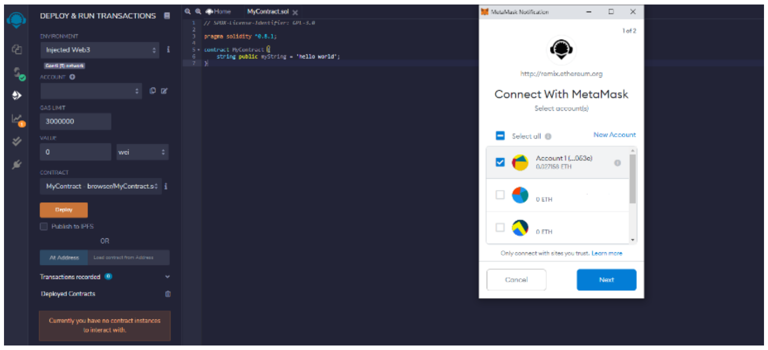
Connect MetaMask to Remix

Switch over to the "Deploy & Run Transactions" Plugin. We need to configure it, so it uses our MetaMask Wallet to access the Blockchain.



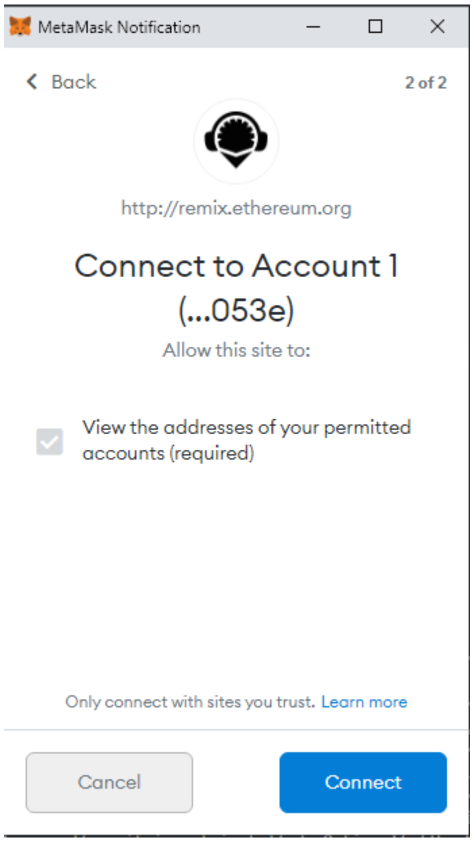
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

contract MyContract {
    string public myString = 'hello world';
}
```

As soon as you do this, MetaMask should pop up and ask you to connect your account to Remix.



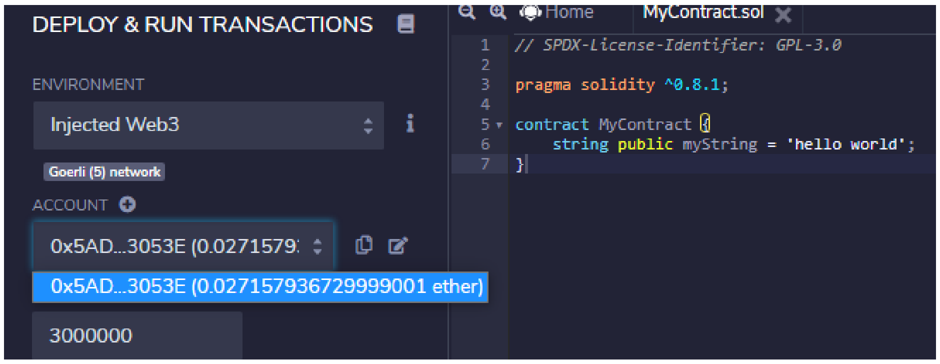
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

contract MyContract {
    string public myString = 'hello world';
}
```

Click "Next" and the "Connect":



MetaMask Notification
Back
2 of 2
http://remix.ethereum.org
Connect to Account 1 (...053e)
Allow this site to:
View the addresses of your permitted accounts (required)
Only connect with sites you trust. Learn more
Cancel
Connect

Now your account should pop-up in the dropdown under the Environment Selection:


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Injected Web3

Goerli (5) network

ACCOUNT

0x5AD...3053E (0.0271579:

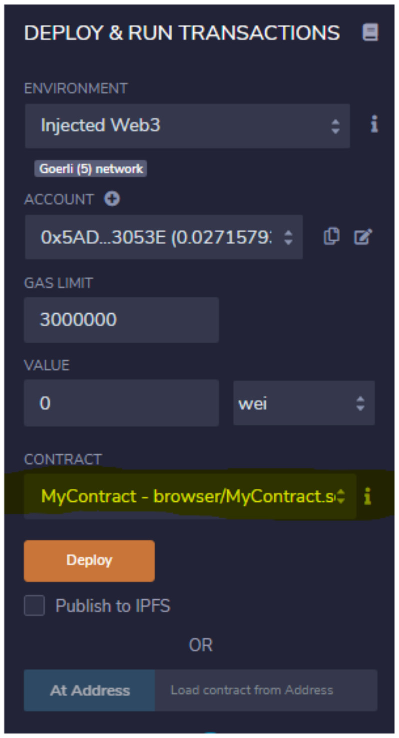
0x5AD...3053E (0.027157936729999001 ether)

3000000

Home MyContract.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

contract MyContract {
    string public myString = 'hello world';
}
```

Deploy the Smart Contract. Let’s deploy the Smart Contract now. First, make sure the correct Smart Contract is selected in the Dropdown:


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Web3

Goerli (5) network

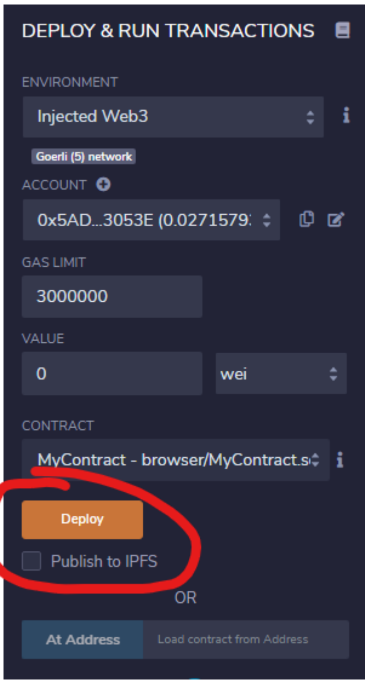
ACCOUNT: 0x5AD...3053E (0.0271579:

GAS LIMIT: 3000000

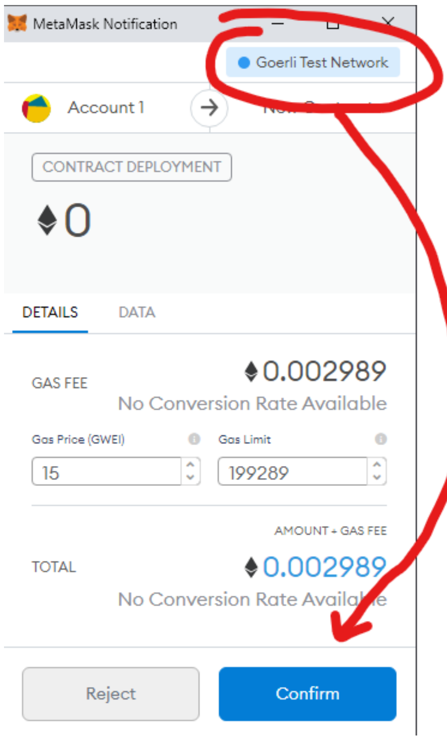
VALUE: 0 wei

CONTRACT: MyContract - browser/MyContract.s

Then simply hit “Deploy”:



This should trigger MetaMask to ask you if you really want to send this transaction. Make sure the Görli Test-Network is selected and then simply hit “Confirm”. If you selected the wrong network, then cancel the transaction, switch the network in MetaMask and hit “Deploy” again.



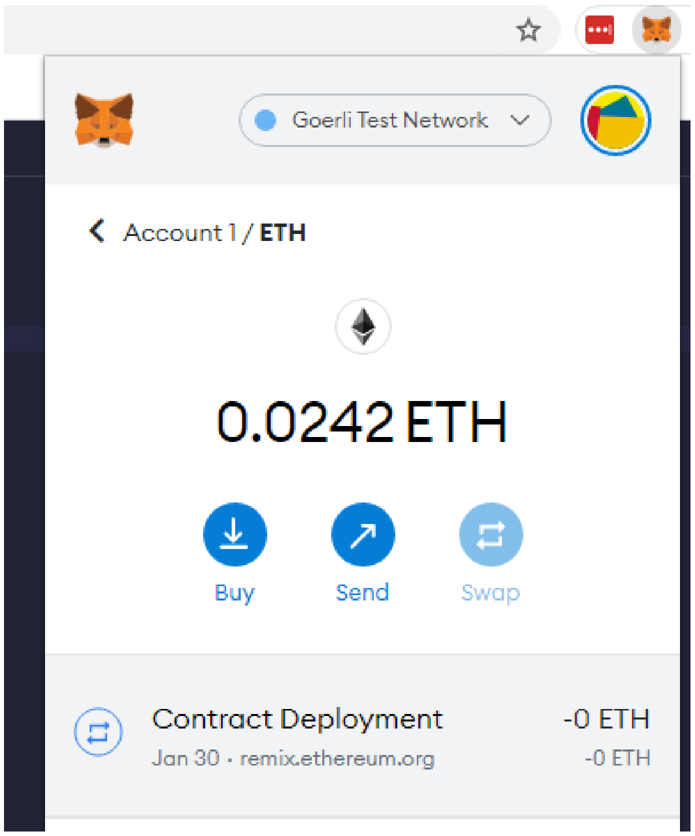
Interact with the Smart Contract

Now let's see if we can interact with the smart contract. Of course, at this point you have no idea what interaction actually means, so, let's just follow along.

Wait For Contract Readiness

First, we need to wait until the transaction is mined. We sent a transaction to the network, but before it's mined the contract won't be ready for any interaction. This can sometimes take a while, and sometimes it's really fast.

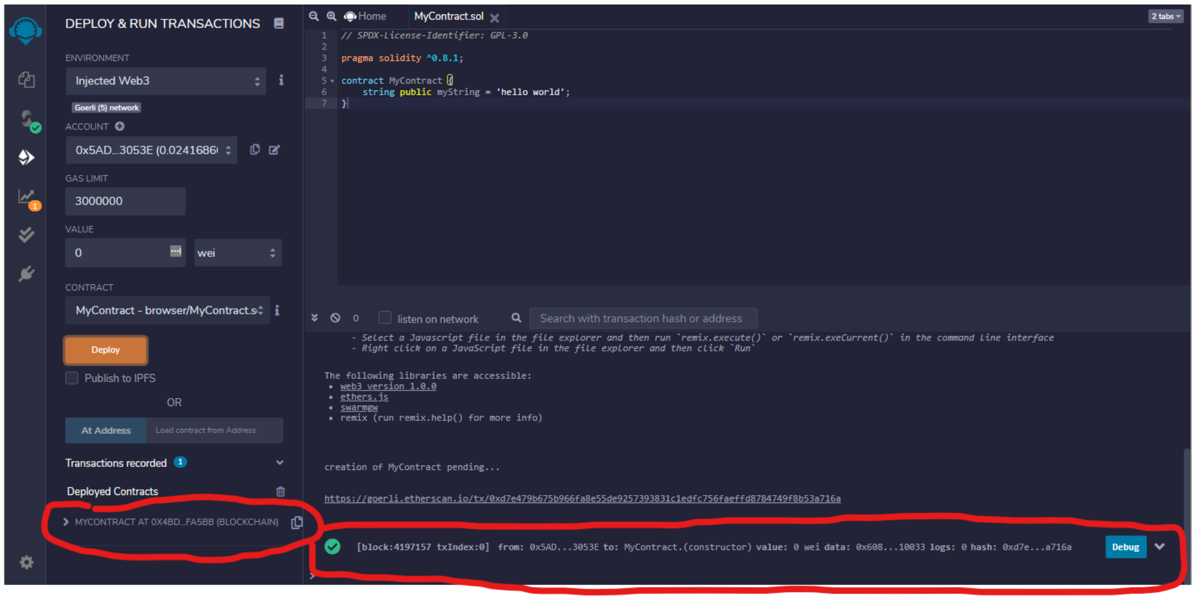
Wait until MetaMask says the Contract Deployment is complete. Open the MetaMask plugin in the top-right corner of Chrome, then check if the Smart Contract was already deployed:



Contract Deployment -0 ETH
Jan 30 · remix.ethereum.org -0 ETH

Wait until it says “Contract Deployment” without a pending flag.

You will also see in Remix that the Contract is now ready:



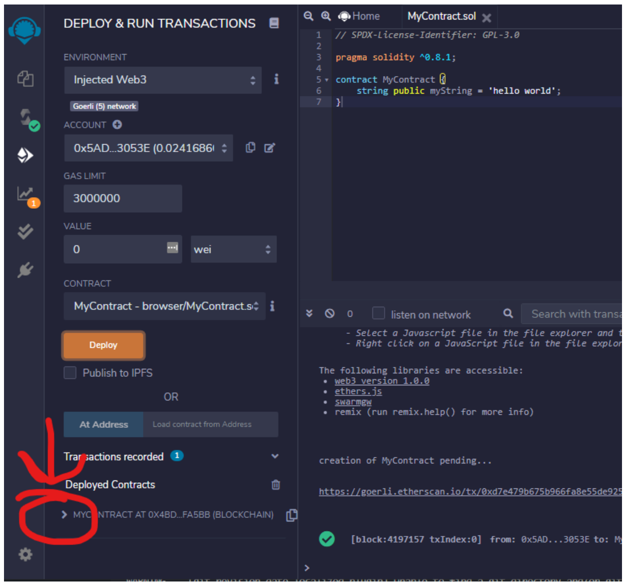
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

contract MyContract {
    string public myString = 'hello world';
}
```

```
    - Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()` in the command line interface
    - Right click on a JavaScript file in the file explorer and then click `Run`

The following libraries are accessible:
 • web3 version 1.0.0
 • ethers.js
 • swarmgw
 • remix (run remix.help() for more info)


creation of MyContract pending...

https://goerli.etherscan.io/tx/0xd7e479b675b966fa8e55de9257393831c1edfc756faeffd8784749f8b53a716a

[block:4197157 txIndex:0] from: 0x5AD...3053E to: MyContract.(constructor) value: 0 wei data: 0x608...10033 logs: 0 hash: 0xd7e...a716a
```

Interact With the Smart Contract

Now it’s time to start our first interaction. In Remix we don’t have to do any low-level things, it conveniently shows us buttons and input fields. You will later see how that works under the hood. We are covering it in the videos about the ABI Array.

Open the Dropdown on the left side:


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Web3 (Goerli (5) network)

ACCOUNT: 0x5AD...3053E (0.0241686

GAS LIMIT: 3000000

VALUE: 0 wei

CONTRACT: MyContract - browser/MyContract.s

Transactions recorded 1

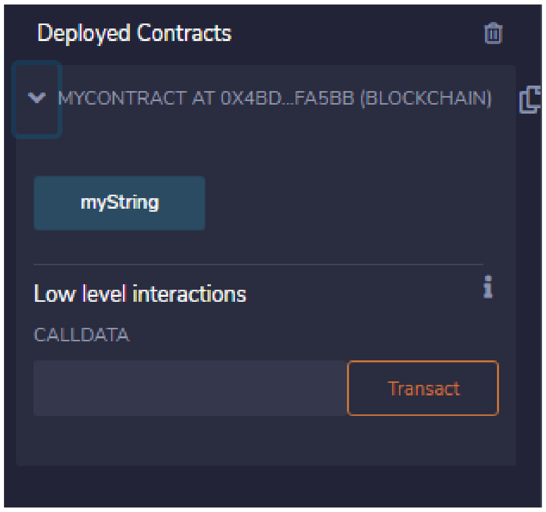
Deployed Contracts

MYCONTRACT AT 0X4BD...FA5BB (BLOCKCHAIN)

```solidity
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.1;

contract MyContract {
    string public myString = 'hello world';
}
```

```
    - Select a Javascript file in the file explorer and t
    - Right click on a JavaScript file in the file explor

The following libraries are accessible:
 • web3 version 1.0.0
 • ethers.js
 • swarmgw
 • remix (run remix.help() for more info)


creation of MyContract pending...

https://goerli.etherscan.io/tx/0xd7e479b675b966fa8e55de925

[block:4197157 txIndex:0] from: 0x5AD...3053E to: My
```

So that you can interact with the newly deployed Smart Contract:

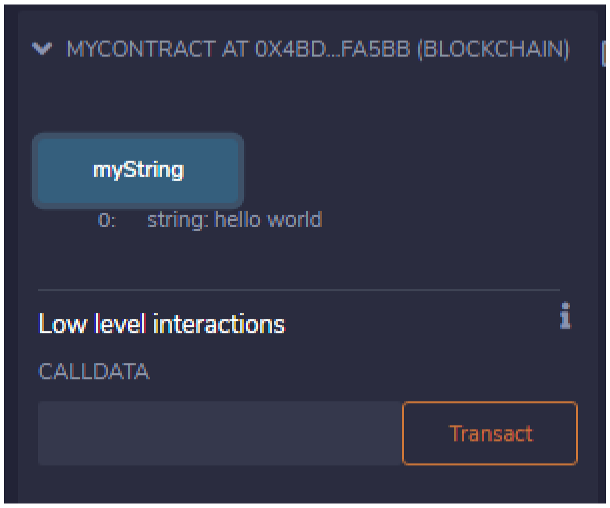

Deployed Contracts

MYCONTRACT AT 0X4BD...FA5BB (BLOCKCHAIN)

Low level interactions

CALLDATA

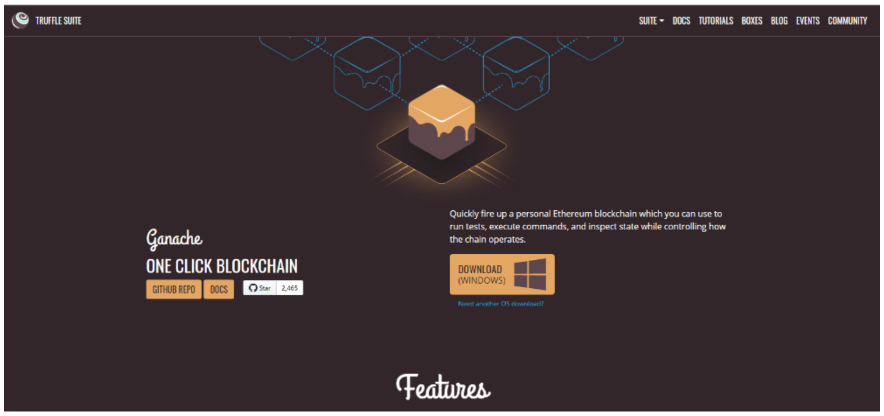
Hit the “myString” Button and you will hopefully see that it returns “hello world” correctly.



Ganache

Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps in a safe and deterministic environment.

Ganache comes in two flavors: a UI and CLI. Ganache UI is a desktop application supporting both Ethereum and Corda technology. The command-line tool, ganache-cli (formerly known as the TestRPC), is available for Ethereum development


TRUFFLE SUITE
SUITE DOCS TUTORIALS BOXES BLOG EVENTS COMMUNITY
Ganache
ONE CLICK BLOCKCHAIN
GITHUB REPO
DOCS
Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.
DOWNLOAD (WINDOWS)
Features

Why use Ganache?

Connecting using a Web3 Provider actually establishes a connection to a software outside of the browser. Like you'd connect to a database. Or to any other API. It's either a RESTful interface or a WebSocket interface. And behind that interface sits a blockchain. This can be Go-Ethereum, Hyperledger Besu, Nethereum – or – Ganache for Development.

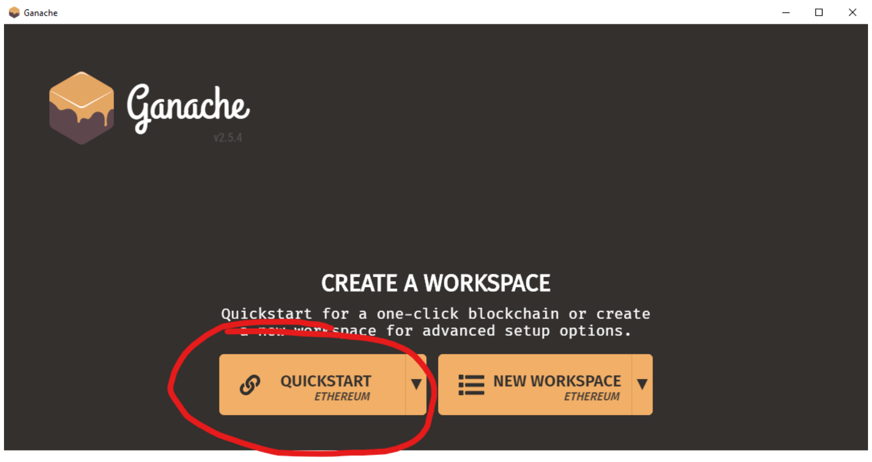
How to Download Ganache?

Go to https://nodejs.org/en/ and download and install the latest version.

Next, we need git. Download and install git from https://gitforwindows.org and install it.

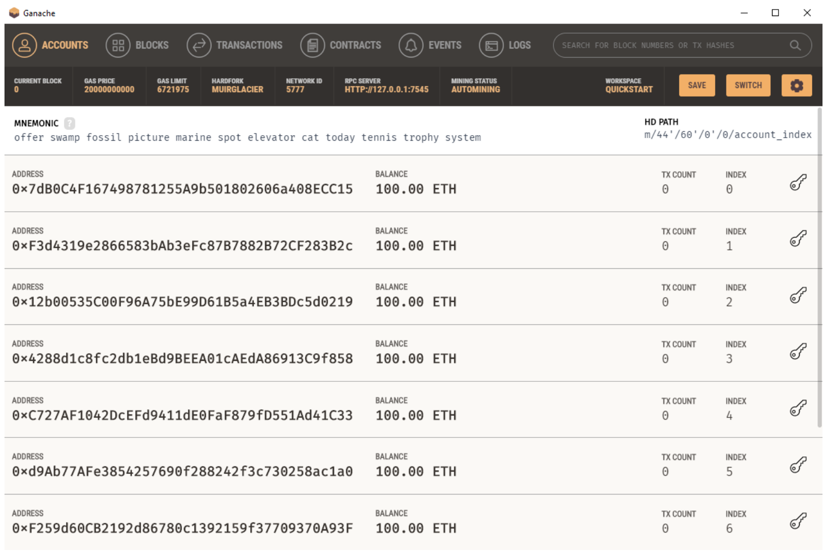
Start Ganache

If you start ganache, it will ask you if you want to do a quick-start or actually start with a workspace.



Quickstart for a one-click blockchain or create a new workspace for advanced setup options.

This will create 10 accounts and assign each 100 ether. The accounts are unlocked. All is ready!



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
0	20000000000	6721975	MUIRGLACIER	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

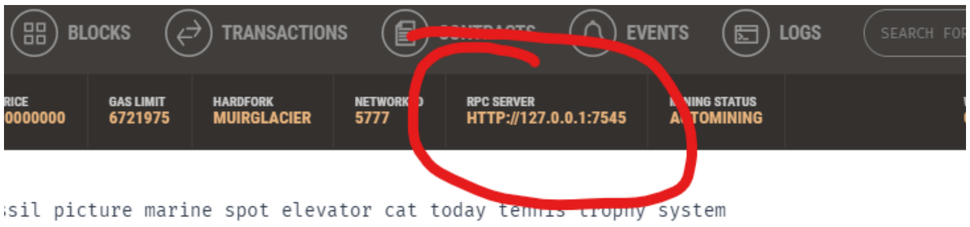
MNEMONIC
offer swamp fossil picture marine spot elevator cat today tennis trophy system

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×7dB0C4F167498781255A9b501802606a408ECC15	100.00 ETH	0	0
0×F3d4319e2866583bAb3eFc87B7882B72CF283B2c	100.00 ETH	0	1
0×12b00535C00F96A75bE99D61B5a4EB3BDc5d0219	100.00 ETH	0	2
0×4288d1c8fc2db1eBd9BEEA01cAEdA86913C9f858	100.00 ETH	0	3
0×C727AF1042DcEFd9411dE0FaF879fD551Ad41C33	100.00 ETH	0	4
0×d9Ab77AFe3854257690f288242f3c730258ac1a0	100.00 ETH	0	5
0×F259d60CB2192d86780c1392159f37709370A93F	100.00 ETH	0	6

Ganache is now a Blockchain and a Wallet. Two in one. Anyone can connect to it using a Web3 Provider Method either via http RPC or WebSockets.

Pay attention to the RPC Endpoint:

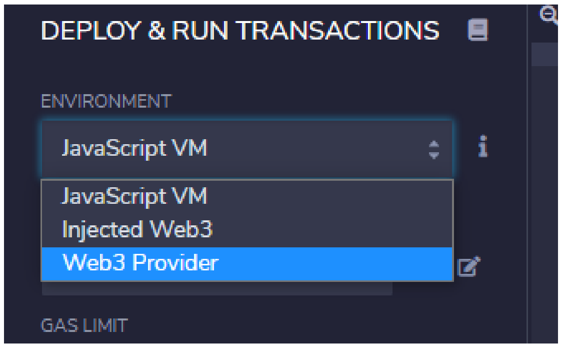


RICE 0000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING

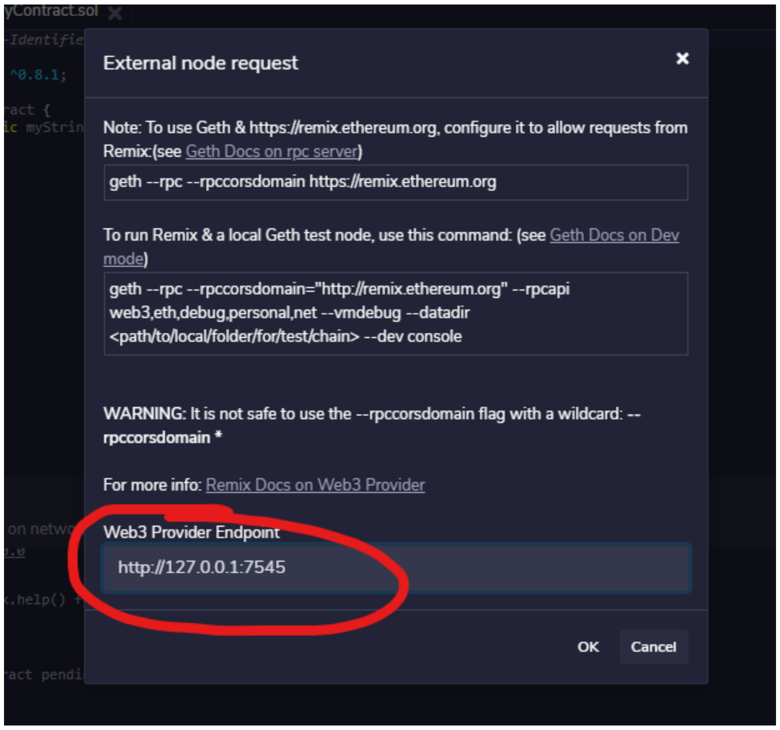
sil picture marine spot elevator cat today tennis trophy system

Connect Remix

From the Environment Dropdown select now “Web3 Provider”:



A new Popup will appear, and enter the RPC Server URL from Ganache. Pay attention to the Port number:


External node request

Note: To use Geth & https://remix.ethereum.org, configure it to allow requests from Remix:(see Geth Docs on rpc server)

```
geth --rpc --rpccorsdomain https://remix.ethereum.org
```

To run Remix & a local Geth test node, use this command: (see Geth Docs on Dev mode)

```
geth --rpc --rpccorsdomain="http://remix.ethereum.org" --rpcapi
web3,eth,debug,personal,net --vmdebug --datadir
<path/to/local/folder/for/test/chain> --dev console
```

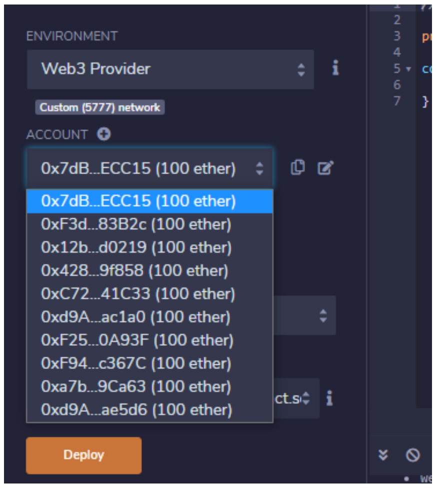
WARNING: It is not safe to use the --rpccorsdomain flag with a wildcard: --rpccorsdomain *

For more info: Remix Docs on Web3 Provider

Web3 Provider Endpoint

```
http://127.0.0.1:7545
```

Your Accounts from Ganache should popup in the Accounts-Dropdown:


ENVIRONMENT: Web3 Provider

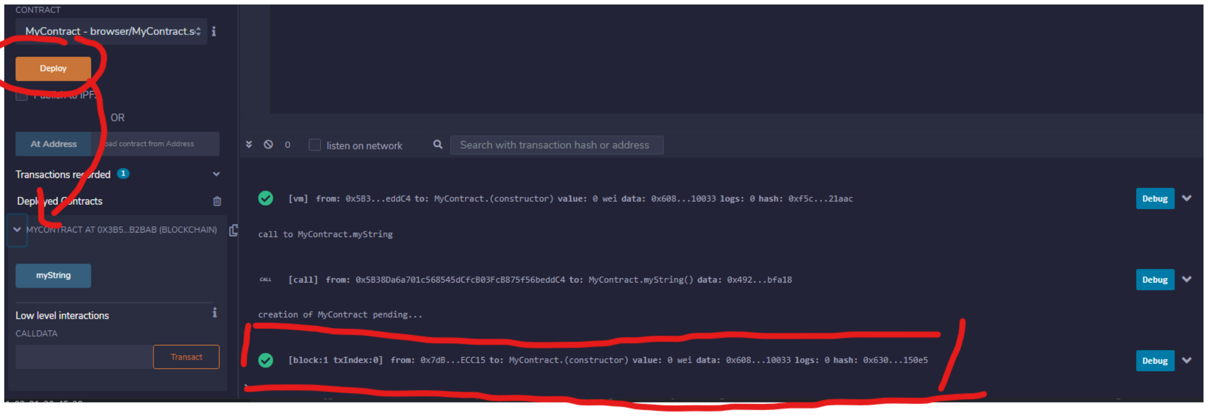
Custom (5777) network

ACCOUNT: 0x7dB...ECC15 (100 ether)

```
0x7dB...ECC15 (100 ether)
0xF3d...83B2c (100 ether)
0x12b...d0219 (100 ether)
0x428...9f858 (100 ether)
0xC72...41C33 (100 ether)
0xd9A...ac1a0 (100 ether)
0xF25...0A93F (100 ether)
0xF94...c367C (100 ether)
0xa7b...9Ca63 (100 ether)
0xd9A...ae5d6 (100 ether)
```

Deploy to Ganache

Now let's see what happens if we hit the Deploy Button:



```
[vm]  from: 0x5B3...eddC4 to: MyContract.(constructor) value: 0 wei data: 0x608...10033 logs: 0 hash: 0xf5c...21aac

call to MyContract.myString

CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MyContract.myString() data: 0x492...bfa18

creation of MyContract pending...

[block:1 txIndex:0]  from: 0x7dB...ECC15 to: MyContract.(constructor) value: 0 wei data: 0x608...10033 logs: 0 hash: 0x630...150e5
```

OpenZeppelin

OpenZeppelin Contracts helps you minimize risk by using battle-tested libraries of smart contracts for Ethereum and other blockchains. It includes the most used implementations of ERC standards.

How does OpenZeppelin work?

OpenZeppelin is a “battle-tested” open source framework comprised of reusable Ethereum smart contracts. The framework helps smart contract developers reduce the risk of vulnerabilities in their distributed applications (dapps) by using standard, tested, community-reviewed code.

ZEPPELIN

Library used to import OpenZeppelin in a smart contract:

import “https://github.com/OpenZeppelin/openzeppelincontracts/blob/master/contracts/access/Ownable.sol”;

Truffle

Truffle is a world-class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier. Truffle is widely considered the most popular tool for blockchain application development with over 1.5 million lifetime downloads.

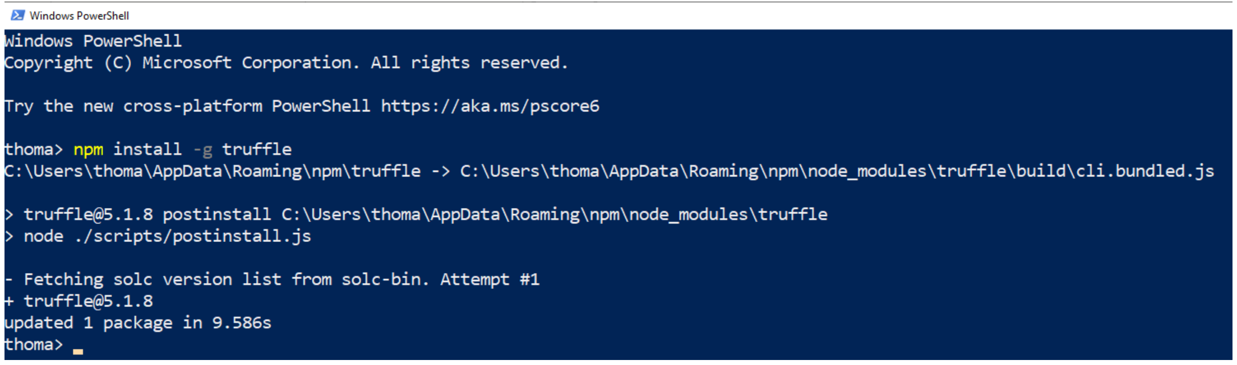
In simple words, Truffle is a development environment, testing framework, and asset pipeline all rolled into one. It is based on Ethereum Blockchain and is designed to facilitate the smooth and seamless development of Dapps (Distributed Applications).

Install Truffle

To install truffle open a terminal (Mac/Linux) or a PowerShell (Windows 10)

Type in:

npm install –g truffle


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

thoma> npm install -g truffle
C:\Users\thoma\AppData\Roaming\npm\truffle -> C:\Users\thoma\AppData\Roaming\npm\node_modules\truffle\build\cli.bundled.js

> truffle@5.1.8 postinstall C:\Users\thoma\AppData\Roaming\npm\node_modules\truffle
> node ./scripts/postinstall.js

- Fetching solc version list from solc-bin. Attempt #1
+ truffle@5.1.8
updated 1 package in 9.586s
thoma>
```

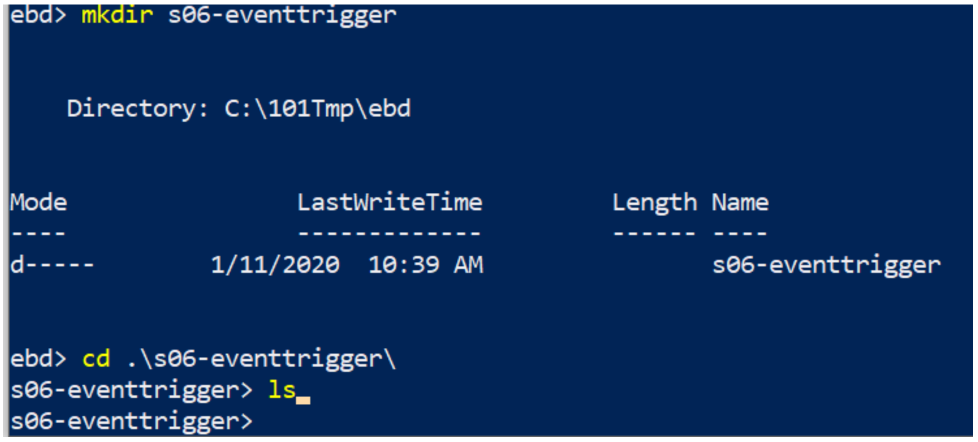
Hint: I am working here with version 5.1.8 of Truffle. If you want to follow the exact same version then type in npm install –g truffle@5.1.8

Then create an empty folder, in this case I am creating "s06-eventtrigger"

mkdir s06-eventtrigger

cd s06-eventtrigger

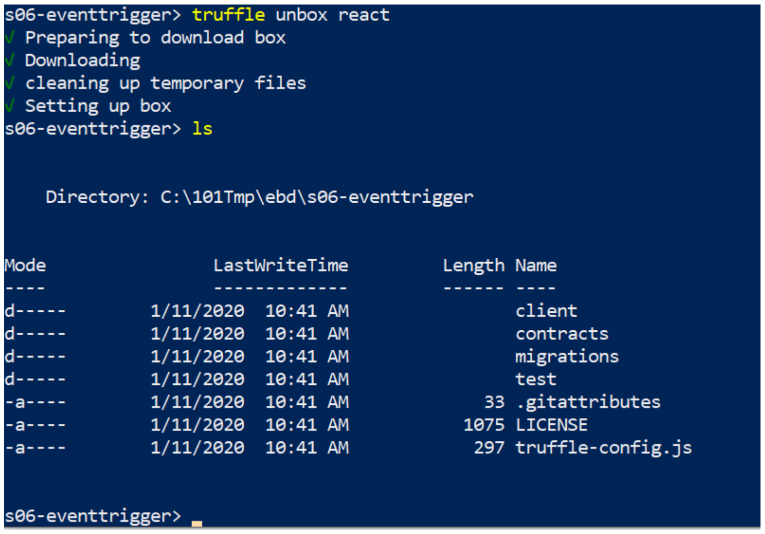
ls


```
ebd> mkdir s06-eventtrigger


    Directory: C:\101Tmp\ebd


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        1/11/2020  10:39 AM                s06-eventtrigger


ebd> cd .\s06-eventtrigger\
s06-eventtrigger> ls
s06-eventtrigger>
```

And unbox the react box:

truffle unbox react

this should download a repository and install all dependencies in the current folder:



```
s06-eventtrigger> truffle unbox react
√ Preparing to download box
√ Downloading
√ cleaning up temporary files
√ Setting up box
s06-eventtrigger> ls


    Directory: C:\101Tmp\ebd\s06-eventtrigger


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/11/2020  10:41 AM                client
d-----         1/11/2020  10:41 AM                contracts
d-----         1/11/2020  10:41 AM                migrations
d-----         1/11/2020  10:41 AM                test
-a----         1/11/2020  10:41 AM             33 .gitattributes
-a----         1/11/2020  10:41 AM           1075 LICENSE
-a----         1/11/2020  10:41 AM            297 truffle-config.js


s06-eventtrigger> _
```

METHODOLOGY

Blockchain technology is one solution that can be used to reduce the problems that occur in crowdfunding. The contract is written in a way that all money will be added to the pool. When the request meets the specified condition then all the money will be transferred to the recipient. Ethereum is an open-source, public, blockchain based distributed platform operating to feature smart contract functionality. It is the modified version of Bitcoin via transaction-based state transitions. Ether is a cryptocurrency which is generated and used by the Ethereum platform. Ethereum provides a decentralized operating, the Ethereum Virtual Machine (EVM), which can execute an application on the public nodes using:

Peer to Peer System*:* The very important part of how blockchain works is based on the Peer to Peer (P2P) system. The whole blockchain is connected to all the nodes in the network.

Consensus Protocol: Consensus protocol is the most important one in blockchain technology. The Blockchain consensus protocol is what keeps the blocks on all the nodes synchronized with each other.

Proof of work (abbreviated to PoW) is a consensus protocol used widely by many cryptocurrencies. This process is known as mining and the nodes on the network are called miners. The Proof of Work is a mathematical problem one that requires considerable work to achieve the solution.

RESULTS & DISCUSSIONS

Blockchain Introduction: The training covers almost every basic aspect of Blockchain. Blockchain is a widely used decentralized, general-purpose, interpreted, ledger technology. Its design philosophy emphasizes Immutability, and it allows safety and security to digital currency, making the world a safer place and less prone to scams.

Founder of Blockchain: Satoshi Nakamoto (late 2008s)

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).

Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

When deploying contracts, you should use the latest released version of Solidity. This is because breaking changes as well as new features and bug fixes are introduced regularly. We currently use a 0.x version number to indicate this fast pace of change.

Who Uses Blockchain Today?

Blockchain isn't exactly a new technology anymore, considering it is over a decade old. However, aside from the financially speculative hype surrounding Bitcoin, blockchain seems to be still trying to establish itself as a mainstream revolution.

- According to Statista, 33% of global organizations say that their companies are working on creating a digital currency using the technology.

- Global enterprise companies including IBM, Walmart and Shell currently have tangible use cases for blockchain technology and continue to research and develop new applications. The

IBM blockchain is presently being used to provide transparency and data integrity within healthcare systems.

● Shell is working with blockchain technology and finance partners to create a platform for the trade and settlement of crude oil. Also, Walmart uses blockchain to deploy a food traceability system.

● Pharmaceutical companies can use blockchain technology to improve supply chains and track things such as vaccine distributions or tracing units of food. Information such as this could be valuable for both vendors and consumers.

Why Do People Use Blockchain?

Blockchain is certainly exciting and has the potential to transform how many businesses operate, but that doesn't mean it's the right solution for every scenario. Here's why you might choose blockchain over, say, a standard database:

● When you want to manage and secure digital relationships or keep a decentralised, shared system of record. 'Smart contracts', in particular, are great for facilitating digital relationships and transactions. With a smart contract, automated payments can be released when parties in a transaction agree that their conditions have been met.

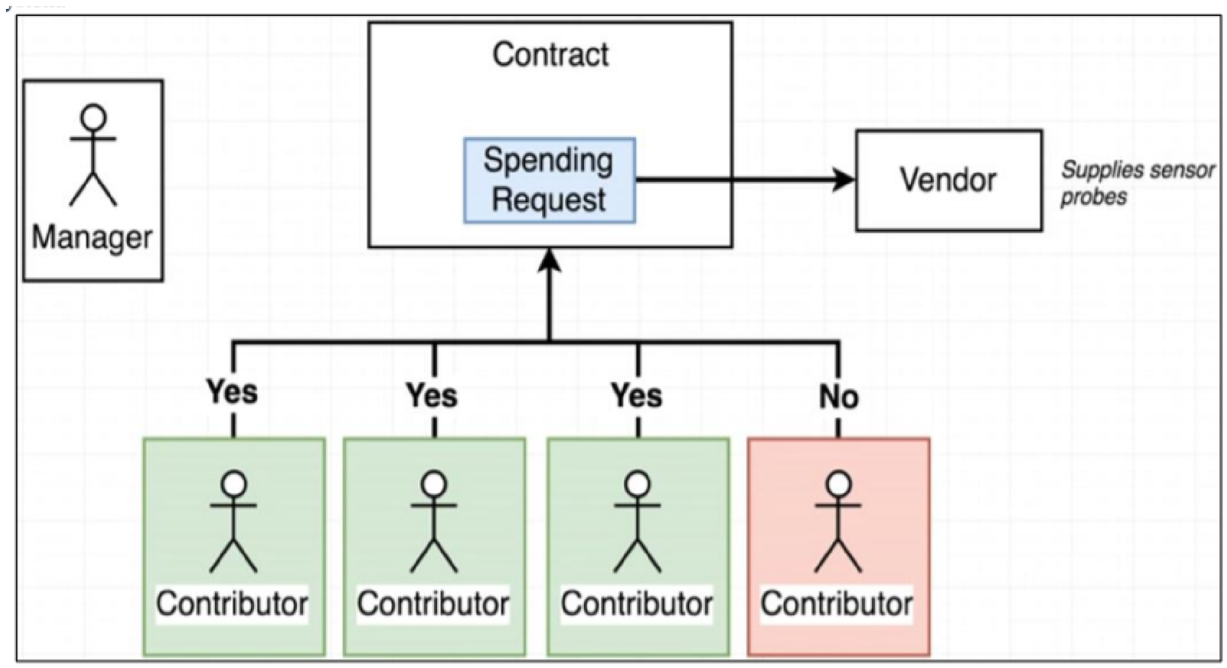
● Anywhere a middleman or gatekeeper functions is expensive or time-consuming.

● When you want to record secure transactions, especially between multiple partners.

● Where the data is in constant flux, but you want to keep a record of past actions. Blockchain is a better, safer way to record activity and keep data fresh, while maintaining a record of its history. The data can't be corrupted by anyone or accidentally deleted, and you benefit from both a historical trail of data, plus an instantly up-to-date record.

ARCHITECTURE

All the contract code is written in solidity which is used to deploy contracts in the blockchain platform. The Campaign Factory is built which contains all the source code to deploy new campaigns in the network. With the use of a campaign factory, the new campaigns can be created. Whenever a campaign factory is deployed a one-time gas fee is needed and it is a very small amount. Initially, the new Campaign is created by giving the Idea of the project, Minimum Contribution to the project and detailed description of the project. When a new campaign is created a block will be created and added to the blockchain. Fig.1 shows the architecture of the system.



The proposed system is implemented using the solidity programming language. Solc is the solidity compiler used to compile the Campaign Factory and Campaign file into bytecode and abi. The Bytecode will be deployed in the blockchain where the abi is in JSON format and is used to interact with the front-end. The front-end is designed using the React Js, Next Js, and Semantic-UI. The user interactive form will be used to contribute easily. The creator or manager of the project is to request money for buying some accessories. He will create a request using the request form. This will be recorded and stored in the blockchain. All the investors need to approve the request if it is necessary. If not they can reject the request. Once all the investors all voted then the request will be finalized, there should be a minimum of 1/2th of the investor should approve the request. If it meets the requirement then the money will be transferred to the vendor.

A. Project creation

A project manager starts a new project in the first step by specifying the project's name, description, and minimum commitment to that project. Contributors may then go through all of the available projects on the crowdfunding site and choose which one they wish to give to. To be recognised as contributors, they must make the minimal contribution to the project that the project manager specified when the project was founded. This money is then deposited in the wallet, which the project managers can use, as shown in Fig. 1.

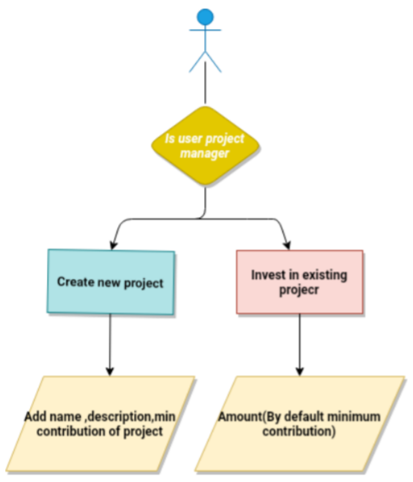

Is user project manager
Create new project
Invest in existing projecr
Add name ,description,min contribution of project
Amount(By default minimum contribution)

Fig. 1. Creating or contributing to project

B. Spending request

If a project manager wants to spend the money contributed by investors at this stage, they must create a spending request that includes a description of where they intend to spend the money, the total amount they intend to spend, and the address of the vendor who will supply the items required by the project manager

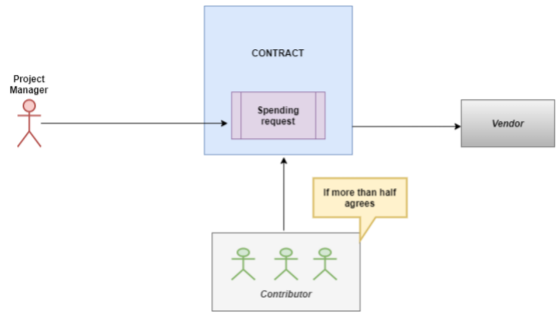


Fig. 2. Voting system ensures money spent is in control of contributors

C. Voting system

The voting system is designed so that only contributors who have invested in that specific project can accept or reject the project managers' expenditure proposals. The voting method also assures that once a contributor votes in favour of a spending request, he or she cannot vote in favour of it again. So, if more than half of the project's contributors agree to the spending request, the funds are delivered to the vendor so that the user can provide the utilities requested by the project manager, and there will be a mechanism in place to protect the project manager's interests if contributors are inactive, the spending request will be processed based on the number of votes cast by all active contributors.

IMPLEMENTATION

A smart contract written in Solidity language is required to construct the crowdfunding platform. The code is then built and distributed on the Ethereum network using the solidity compiler. To complete all transactions, Metamask, a Chrome browser plugin, is utilized. The following is the procedure for creating a crowd fundraising platform:

Step 1: Make a smart contract.

Step 2: Compile the smart contract in order to receive the bytecode and application binary interface (ABI).

Step 3: Upload bytecode to the Ethereum blockchain.

A. Smart contract creation

It is a software created in the Solidity programming language that handles all transactions automatically.

As illustrated in Fig. 3, the project manager must first construct the project by mentioning its name, description, and the minimal contribution. The user can then create a spending request to spend the money provided by the investors. For this project, authors must include a description of where they intend to spend money, the amount of money they intend to spend, and the address of the vendor who will give some service. If more than half of the investors agree to the expenditure request, the project manager has the authority to transmit the funds to the vendor's address. The vendor will then offer the service required by the project manager.

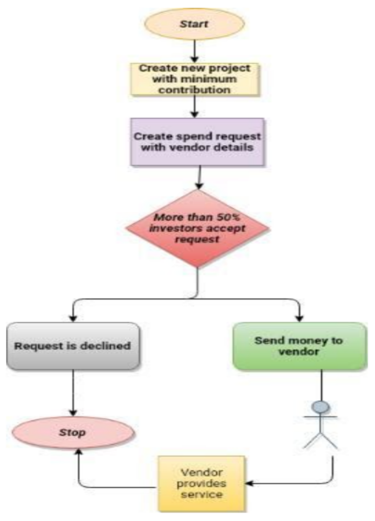

Start
Create new project with minimum contribution
Create spend request with vendor details
More than 50% investors accept request
Request is declined
Send money to vendor
Stop
Vendor provides service

Fig. 3. Flow chart of project manager

If the investor is interested, it is displayed in Fig. 4.

If a project is mentioned on a crowdfunding site, They may participate in the initiative by making a little contribution which the project manager has set while creating the project. Then this money is added to the wallet assigned for the specific project. After that the contributor can either accept the spending request sent by the project manager or decline.

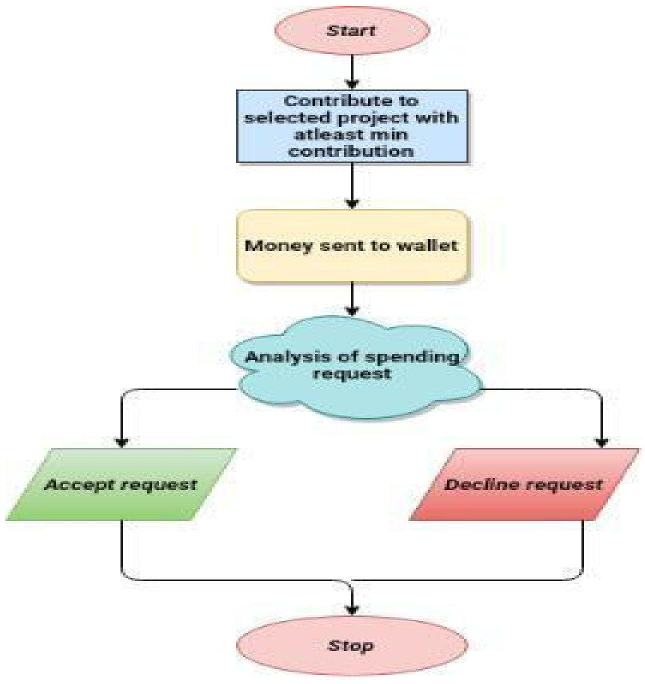

Start
Contribute to selected project with atleast min contribution
Money sent to wallet
Analysis of spending request
Accept request
Decline request
Stop

Fig. 4. Flow chart of investor

B. Compilation and Deployment

The smart contract is compiled using the solidity compiler. This gives bytecode and application binary interface as out-put. Bytecode is then deployed to ethereum blockchain and application binary interface is used to interact with smart con-tract. Bytecode is a hexadecimal representation of the compiled contract which can only be understood by Ethereum Virtual Machine(EVM).

The bytecode obtained from the compilation can be deployed to either rinkeby test network, robsten test network or ethereum live network. After deploying they return the address where the smart contract is deployed using which the user can make the transactions.

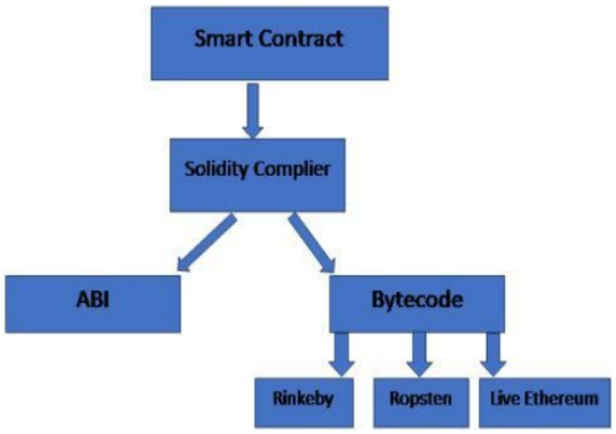
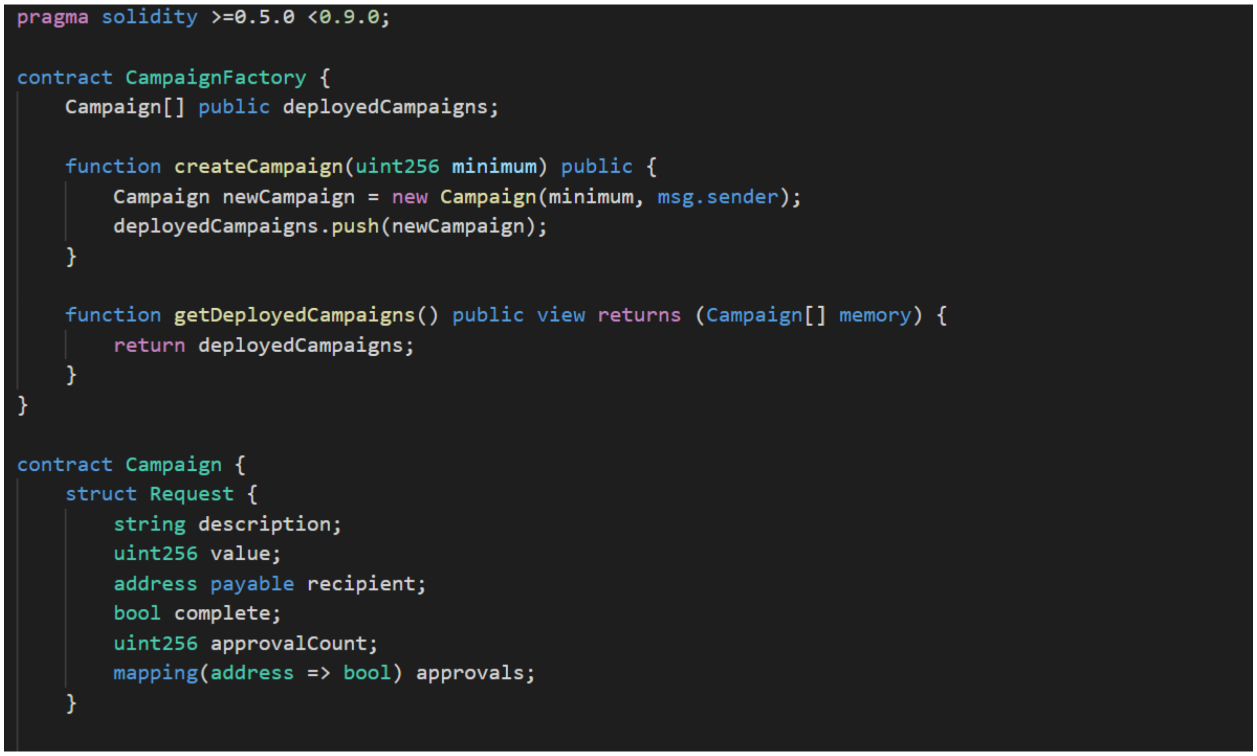
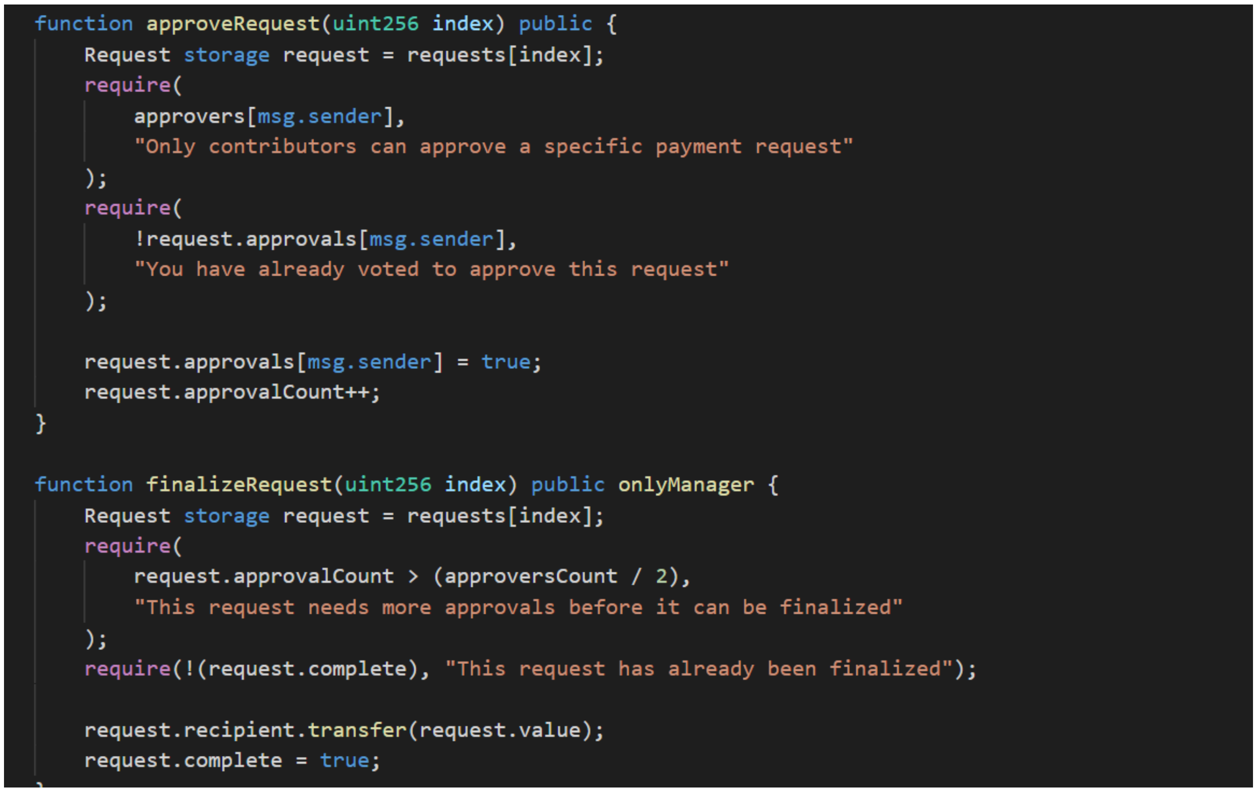


Fig. 5. Compilation and deployment of smart contract

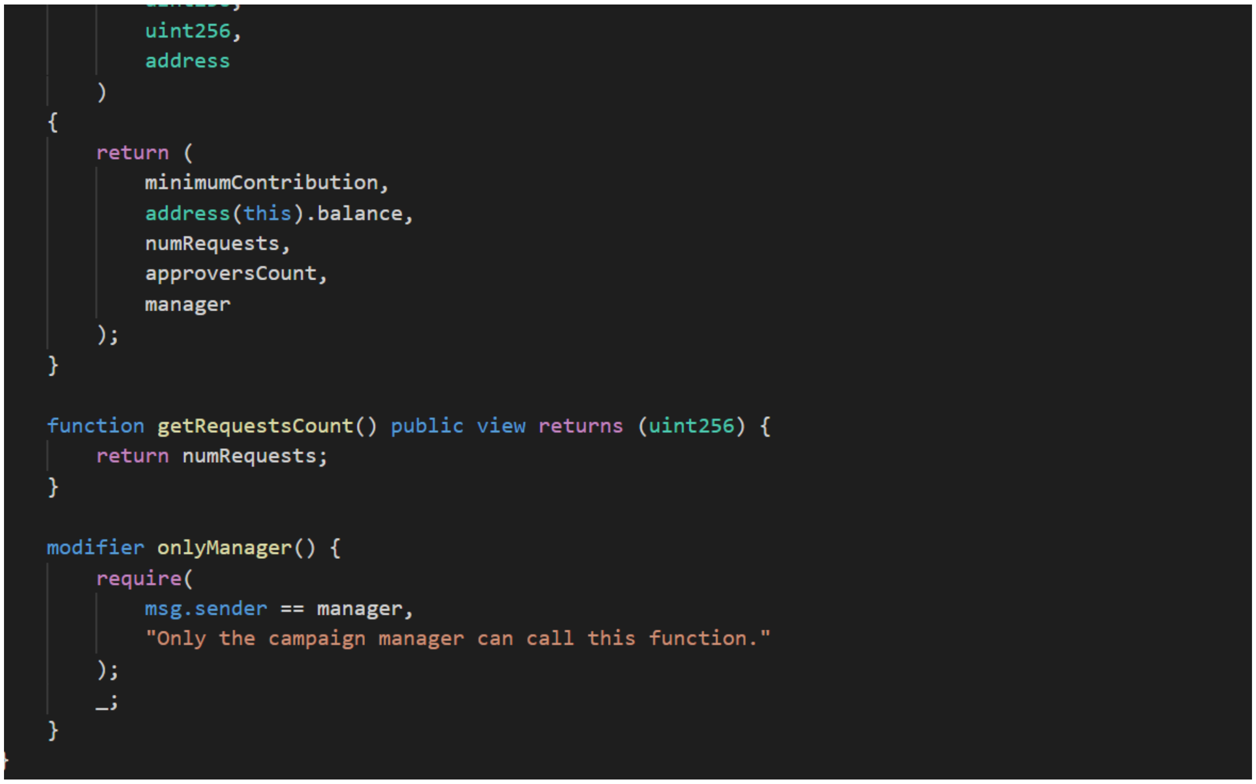
CODE SNIPPETS



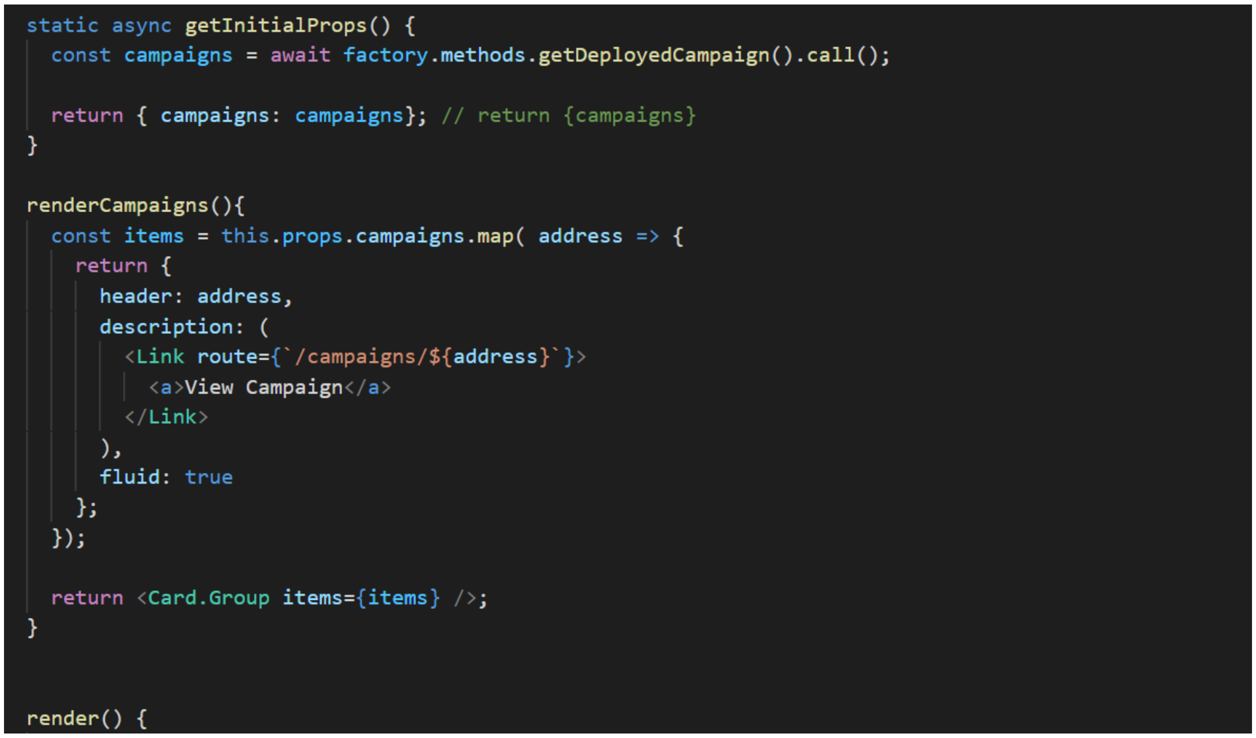
```
pragma solidity >=0.5.0 <0.9.0;

contract CampaignFactory {
    Campaign[] public deployedCampaigns;

    function createCampaign(uint256 minimum) public {
        Campaign newCampaign = new Campaign(minimum, msg.sender);
        deployedCampaigns.push(newCampaign);
    }

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

contract Campaign {
    struct Request {
        string description;
        uint256 value;
        address payable recipient;
        bool complete;
        uint256 approvalCount;
        mapping(address => bool) approvals;
    }
```



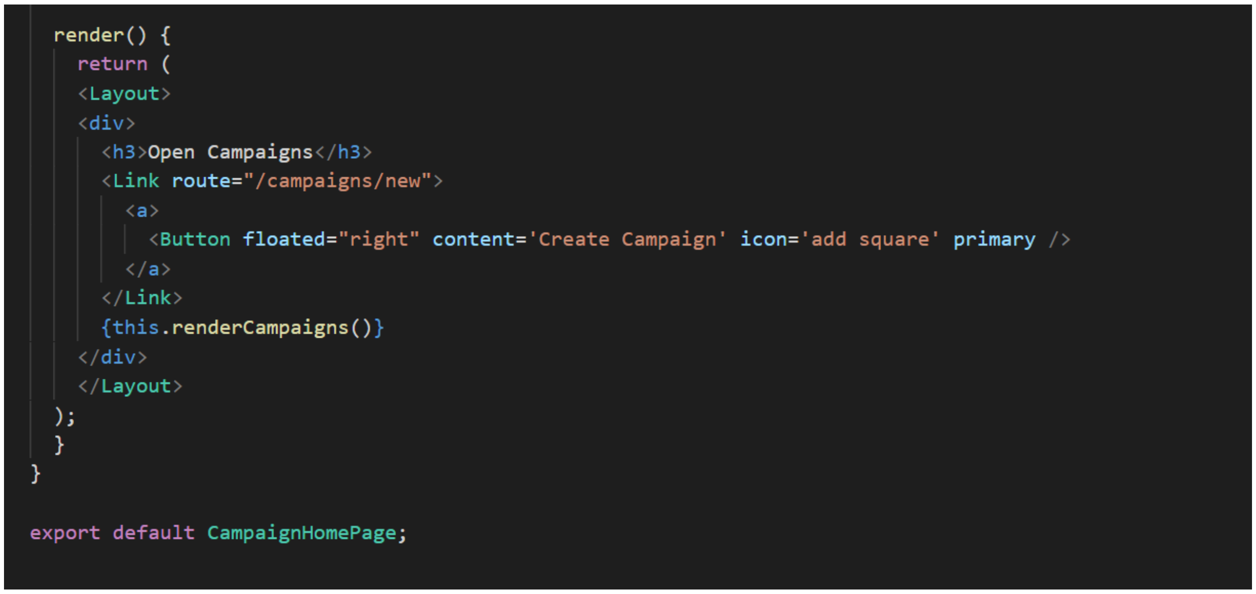
```
function approveRequest(uint256 index) public {
    Request storage request = requests[index];
    require(
        approvers[msg.sender],
        "Only contributors can approve a specific payment request"
    );
    require(
        !request.approvals[msg.sender],
        "You have already voted to approve this request"
    );

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

function finalizeRequest(uint256 index) public onlyManager {
    Request storage request = requests[index];
    require(
        request.approvalCount > (approversCount / 2),
        "This request needs more approvals before it can be finalized"
    );
    require(!(request.complete), "This request has already been finalized");

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



```
            uint256,
            address
        )
    {
        return (
            minimumContribution,
            address(this).balance,
            numRequests,
            approversCount,
            manager
        );
    }

    function getRequestsCount() public view returns (uint256) {
        return numRequests;
    }

    modifier onlyManager() {
        require(
            msg.sender == manager,
            "Only the campaign manager can call this function."
        );
        _;
    }
}
```

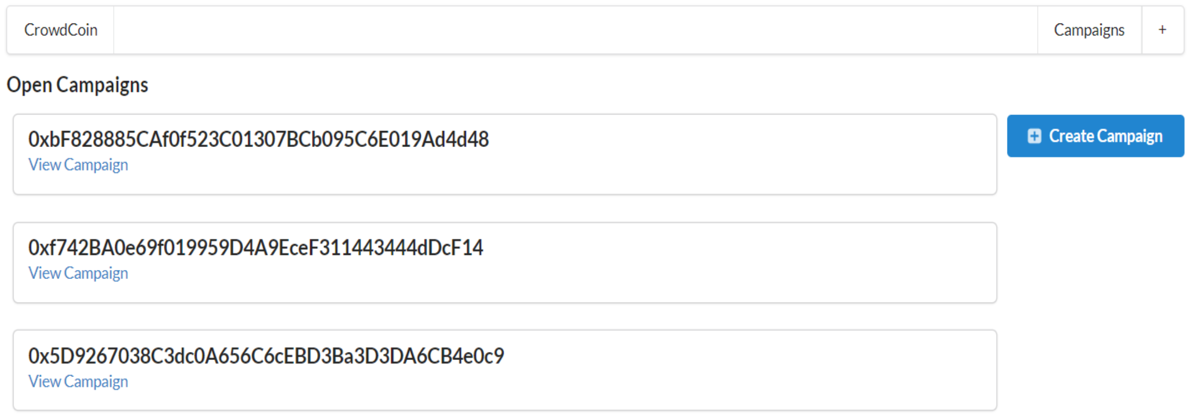


```
static async getInitialProps() {
  const campaigns = await factory.methods.getDeployedCampaign().call();

  return { campaigns: campaigns}; // return {campaigns}
}

renderCampaigns(){
  const items = this.props.campaigns.map( address => {
    return {
      header: address,
      description: (
        <Link route={`/campaigns/${address}`}>
          <a>View Campaign</a>
        </Link>
      ),
      fluid: true
    };
  });

  return <Card.Group items={items} />;
}


render() {
```



```
  render() {
    return (
    <Layout>
    <div>
      <h3>Open Campaigns</h3>
      <Link route="/campaigns/new">
        <a>
          <Button floated="right" content='Create Campaign' icon='add square' primary />
        </a>
      </Link>
      {this.renderCampaigns()}
    </div>
    </Layout>
  );
  }
}

export default CampaignHomePage;
```

OUTPUTS



CrowdCoin   Campaigns   +

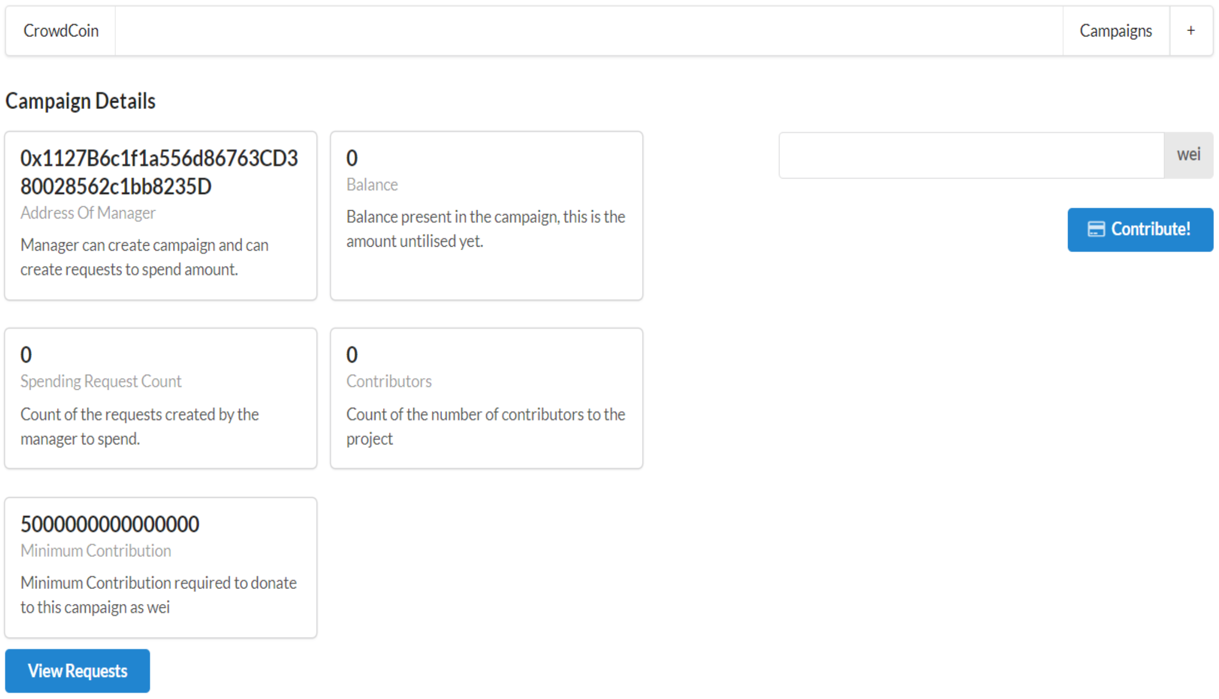
Open Campaigns

0xbF828885CAf0f523C01307BCb095C6E019Ad4d48
View Campaign

0xf742BA0e69f019959D4A9EceF311443444dDcF14
View Campaign

0x5D9267038C3dc0A656C6cEBD3Ba3D3DA6CB4e0c9
View Campaign

Create Campaign



CrowdCoin   Campaigns   +

Campaign Details

0x1127B6c1f1a556d86763CD380028562c1bb8235D
Address Of Manager
Manager can create campaign and can create requests to spend amount.

0
Balance
Balance present in the campaign, this is the amount untilised yet.

0
Spending Request Count
Count of the requests created by the manager to spend.

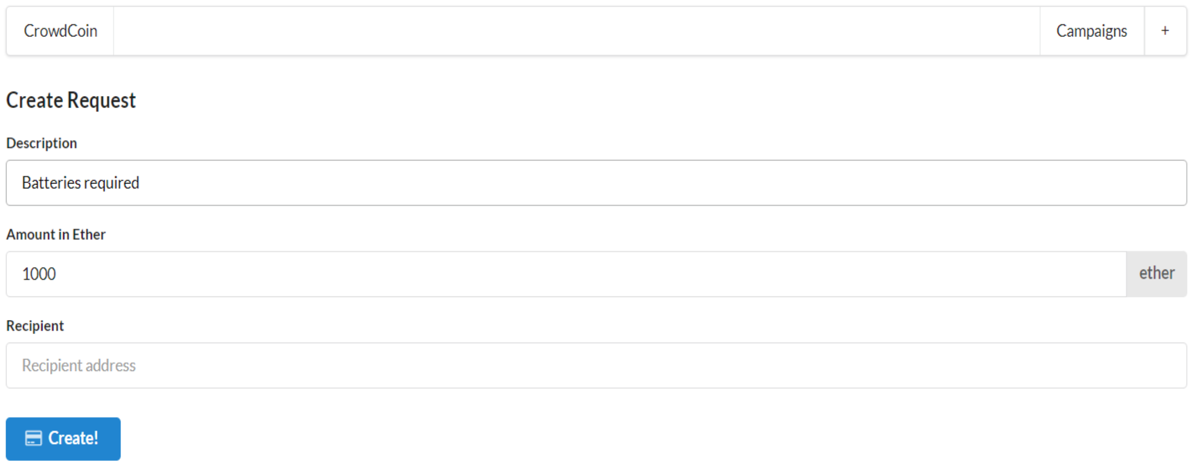
0
Contributors
Count of the number of contributors to the project

5000000000000000
Minimum Contribution
Minimum Contribution required to donate to this campaign as wei

View Requests

wei

Contribute!


CrowdCoin Campaigns +

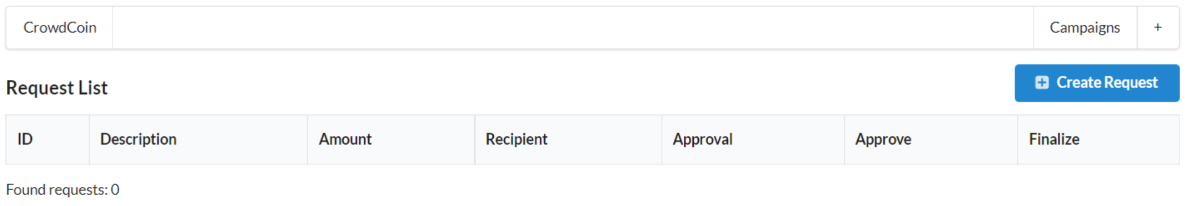
Create Request

Description: Batteries required

Amount in Ether: 1000 ether

Recipient: Recipient address

Create!


CrowdCoin Campaigns +

Request List

Create Request

ID	Description	Amount	Recipient	Approval	Approve	Finalize

Found requests: 0

Conclusion

Blockchain in crowdfunding is a relatively new concept to the community. We have taken that into consideration and designed this app so that even a common man can use it with ease. But this is not the end. With the evolution of Blockchain and introduction of ICOs, our application has a bright future and a large scope for improvement and evolution. The world is still adjusting to Blockchain and Cryptocurrencies and it'll take a couple of years more for Ethereum based Dapps to become popular and to be recognized by the community. In such a situation Blockchain based crowdfunding application is a tough concept to be understood by everyone. We have taken that into consideration and designed this app so that even a common man can use it with ease. But this is not the end. With the evolution of Blockchain and introduction of ICOs, our application has a bright future and a large scope for improvement and evolution. In the future, we wish to provide an even easier and safer way for all ideas to get life through our crowdfunding application.

Finally, it is argued that blockchain-based crowdfunding is a relatively new notion in the ICT community. So far, the solidity code for the campaign contract has been successfully developed and built using the solidity compiler. The solidity compiler produces bytecode, and the interface was published onto the Ethereum network using metamask. Following the deployment of the project, a decentralised web app with a frontend for starting a new project, contributing to an existing project, generating a new request, accepting a request, and finishing a request is established. At the moment, the blockchain use in crowdfunding is still in its early stages, with several legal and specialist difficulties to be resolved.

With the advancement of blockchain technology, our suggested work has a bright future and plenty of room for progress and evolution. In the future, the planned research activity can move forward in a more straightforward and secure manner for all proposals.

REFERENCES

[1] https://www.investopedia.com/terms/c/crowdfunding.asp

[2] Prinsha K, “A Study on Crowd Funding and its Implications in India Paripex,” Indian Journal Of Research, Vol: 5, 1 January 2016.

[3] Arthur Gervais, Ghassan O. Karame, Karl Wust, Vasileios Glykantzis, ¨ Hubert Ritzdorf, and Srdjan Capkun, “A Study on Crowd Funding and its Implications in India Paripex,” In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA, 2016.

[4]https://due.com/blog/a-new-era-of-crowdfunding-blockchain

[5] Bachmann, Alexander Becker, Alexander Buerckner, Daniel Hilker, Michel Kock, Frank Lehmann, Mark Tiburtius, Phillip Funk, Burkhardt, “Online Peer-to-Peer Lending–A Literature,” Online Peer-toPeer Lending–A Literature. Journal of Internet Banking and Commerce, 2011.

[6] Zhu, H., Zhou, Z.Z. “Analysis and outlook of applications of blockchain technology to equity crowdfunding in China,” Financ Innov, 2016.

[7] Gebert, Michael, “Application of blockchain technology in crowdfunding,” New European.

[8] Dhokley, Er Gupta, Saurabh Pawar, Ganesh Shaikh, Abrar, “Crowdsourcing and Crowdfunding Platform using Blockchain and Collective Intelligence,” International Journal of Computer Sciences and Engineering, 2016.

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