IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.918 ⁜ Vol. 12, Issue 2, February 2023

DOI: 10.17148/IJARCCE.2023.12253

Smart Contract Using Solidity (Remix –Ethereum IDE)

Dr. Santosh Kumar Singh[1], Dr. Varun Tiwari[2], Dr. Vikas Rao Vadi[3]

Assistant Professor, Department of CS & IT, DBIT, New Delhi, India[1]

Associate Professor, Department of CS & IT, DBIT, New Delhi, Indi[2]

Professor, Director, DBIT, New Delhi, India[3]

Abstract: Smart contracts are programs organized in blockchain surroundings, which manage the performance of accounts inside the Ethereum state. It is the algorithmic explanation of a prescribed business deal procedure that is spontaneously executed and composed of the data delivered by its parties. Solidity is an object-oriented, high-level language for applying smart contracts. It is impacted by Python, C++, and JavaScript, and is intended to aim at the (EVM) Ethereum Virtual Machine. This article mainly emphasizes how a smart contract is written in the programming language Solidity. Smart contracts are an outstanding approach to generating contracts that can be placed out concretely without human sentiments. This article clarifies what smart contracts are and how to write a contract by displaying the Solidity programming language syntax, i.e. known as the smart contract language. The extension of this article, beginning with the description of smart contracts, also comprises in which regions and in which schemes smart contracts are used. The article also emphasizes the Solidity programming language syntax, which is a statically typed (characteristic of a programming language in which various types are explicitly declared and thus are determined at compile time) programming language considered to generate smart contracts executing on the EVM. The article ends by displaying showing steps to execute a simple solidity smart contract using Remix IDE. Finally.

Keywords: Blockchain, Ethereum, Remix, Solidity, Smart Contracts

I. INTRODUCTION

Blockchain has progressively been commercialized by its impartiality, transparency, and equality. To the development of Ethereum, Blockchain machinery has unlocked the gate. A blockchain platform along with smart contracts that could hold and spontaneously send the tokens. The word smart contract constructing Blocks (like a page of a ledger or record book) for virtual markets was developed through Szabo in 1994 that defines a computer program simplifying the terms and circumstances of an actual domain contract [1]. These discussions are displayed as “if-else” declarations.

For instance, “if (X) sends money to (Y), (X) possibly will gain entree to (Y’s) studio apartment”. Once the predefined conditions are satisfied smart contracts automatically enforce these negotiations, without the interference of an authorized notary or a trusted third-party whereas this is not found in normal contracts.

We can understand this with one more example, let’s assume

➢ My grandson is to be given twenty percent (20%) of my property.
➢ On the other hand, I have some situations i.e. condition, for instance, if my grandson is twenty-five years or older, he will be capable to take over this property.
➢ If we include this condition in the smart contract, then without the requirement of any third-party smart contract will grip the transactions.

Wherever a business deal wants to happen between two parties, you should practice a smart contract that provides you complete control and computerizes those business dealing transactions. This is the objective of smart contract logic. Since the early 90s supposing this idea has been existed, because of a misplaced precondition, specifically, a distributed platform, until the birth of blockchain technology smart contracts could not come into use. It is seen that smart contracts, which were put forward in 1994 by Nick Szabo is computer scientist, lawyer, and cryptographer have become extensive with Blockchain machinery nowadays.

In 2008, Nakamoto [2] presented the idea of blockchain, a huge extent of consideration has been rewarded to this noticeable machinery. At present, Nakamoto characterized his concept over and done with an implementation form called Bitcoin. Many interesting properties of blockchain technology are demanded to have, containing impartiality, transparency, and equality. Due to being developed and running on top of blockchain infrastructure, smart contracts inherit certain interesting properties. The bytecode of smart contracts is immutable i.e., cannot be tampered with once deployed. Moreover, the results returned after smart contract execution are irreversible and permanently recorded inside the blockchain database.This paper is structured as follows: Section 2 presents a general review of smart contracts. Section 3 explains Blockchain basics, Solidity, and EVM, as well as what smart contracts are and how to write a contract by showing the syntax of the Solidity programming language. Section 4 shows steps to execute a simple solidity smart contract using Remix IDE. Finally, Section 5 reports the conclusion.

II. SMART CONTRACT

Szabo [3] has been done initial effort on smart contracts, projected the primary idea and ideologies of smart contracts and examined the opportunities and benefits of building smart contracts by utilizing the Internet, workstations, and additional innovative technologies. Mark S. Miller [4] in 2000, deliberated an arrangement on Szabo's contract basis that is a protected dispersed determined semantic for competence-founded smart contracting. As a new technology, it is in computational rule, has very significant features- once convinced circumstances to come across, contracts will perform suitable movements spontaneously. On the other hand, this story has been functional in related machinery in different applications. For instance, in the 1980s knowledge-based systems had this feature. The rule-based system is the first one where once definite circumstance are encountered, the equivalent rule will be activated. If at the same time more than two rules are triggered, there will be equivalent resolve machinery to organize the implementation of these guidelines.

Blackboard architecture is the second one where various representative are observing all together. The equivalent representative will trigger its possess rule and implement the appropriate procedure, once an exact circumstance is encountered. These representatives can be clustered this is another point of the rule-based system, and these representatives who are in the identical cluster will be sharing the identical facts on a similar platform.

A database trigger is the third one where, an alteration in the information in the databank fulfills the circumstances for the databank trigger, and the equivalent program will be triggered to implement. The final one is the service-oriented system in which if the service caller encounters a convincing situation, then the system will deliver the equivalent facility to the service caller.

The improvement of smart contracts has been gentle since there were no confidence implementation surroundings to encounter the requirement for noticeable, confirmable, and self-imposed. The contracting party is not able to perceive and authenticate the enactment of other contract parties openly, before the blockchain. It frequently includes reliable third parties which is the most exclusive part, they are involved in the enactment stage of contracting. Therefore, we want a contract mechanism that agreements with contract handling spontaneously according to the enactment of contract parties, ho,wever the mechanism will not be exaggerated by any of them. The blockchain is a protected dispersed databank, everything in blockchain could not be altered by any of them however be able to be checked by all of them [5]. Ethereum [6] meaningfully encouraged the improvement of smart contracts to encounter the elementary necessities of smart contracts by Nick Szabo. States of contract in blockchain will not be altered short of accurate transactions, and every alteration of state on it requires to go over the Blockchains dependability set of rules. Ethereum stocks the contract itself as well as its status inside the blockchain.

The contract code kept in the blockchain will be accomplished once the terms and conditions circumstances of the contract are fulfilled. In Ethereum from the time when dispersed terminals finish the implementation of smart contracts, as a result, there is no particular terminal letdown, plus the smart contracts' accomplishment will be unchallengeable and confirmable. For that reason, to combine smart contracts and blockchain there is much room for improvement. A lot of businesses emphasize the exploration of smart contracts and blockchain, for instance Codius (which is a platform for securely executing smart contracts), IBM, SmartContract, Eris, and so on. To be implemented spontaneously Smart contracts need to be surrounded into hardware, software, hence the semantics of smart contracts must be paid consideration. But a program writing semantic must not be like the semantic in day-to-day life that is rich in semantics. The competence of the programming language will be reduced due to the semantically rich language of the contract. Furthermore, languages such as difficult and high-level programming languages are accompanied by possible safety dangers, susceptibilities, and numerous other problems. Several restricted ways out have been projected to discourse exact safety matters, on the other hand over smart contracts there is a lacking of systematic research. Singh et al. [7] examined the existing validation methods for smart contracts in huge aspects and recognized open problems with conceivable resolutions to alleviate these problems. To resolve the smart contract security issues Huang et al. [8] suggested a software development viewpoint by distributing studied investigation into improvement stages.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.918 ⁜ Vol. 12, Issue 2, February 2023

DOI: 10.17148/IJARCCE.2023.12253

III. SOLIDITY, BLOCKCHAIN, ETHEREUM VIRTUAL MACHINE

A. Solidity

Solidity is a statically (variable types are explicitly declared and are determined at compile time) typed programming language aimed to improve smart contracts executing on (EVM) the Ethereum Virtual Machine. For implementing smart contracts best language is Solidity which is a contract-oriented, high-level language. The behaviour of accounts state within the Ethereum is governed by programs i.e. Smart contracts. It is inclined through Python, C++, and JavaScript and was aimed to target the (EVM) Ethereum Virtual Machine. [[Solidity is compiled as bytecode executed in the EVM. Using Solidity, developers can write Decentralized Applications (dApp) that implement a self-enforcing business mechanism that is returned to smart contracts and leave an authoritative record of the transaction.

While discussing Solidity in this article, first the syntax is explained simply, then it is explained how to write an Inheritance contract using this syntax and codes. Contracts in Solidity are very similar to classes in object-oriented languages. They can contain state variables, functions, events, and struct types. When getting started with the Solidity programming language, you don't need any prior experience with the language. However, having some experience with Blockchain technology or at least understanding the basics of what Blockchain is and how it works will help you. You can develop smart contracts on a Mac, Windows, or Linux using the Solidity programming language. In other words, the environment you use will not have any effect on your language learning. And in most cases, you can use the web-based Remix environment, which we will review below. If you want to discover what others are doing, you can check out the link at stateofthedapps.com where you will see hundreds of games and apps created using Ethereum and Solidity [9].

```
pragma solidity >=0.4.16 <0.9.0;
contract Storage
{
   uint Data;
   function setdata(uint x) public
 {
      Data = x;
   }
   function getdata() public view returns (uint)
 {
      return Data;
   }
}
```

The first line tells you that the source code is licensed under the GPL version 3.0. Machine-readable license specifiers The next line specifies that the source code is written for Solidity version 0.4.16, or a newer version of the language up to, but not including version 0.9.0. This is to ensure that the contract is not compliable with a new (breaking) compiler version, where it could behave differently.

A contract in the sense of Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. The line uint stored data; declares a state variable called stored data of type uint (unsigned integer of 256 bits). You can think of it as a single slot in a database that you can query and alter by calling functions of the code that manages the database. In this example, the contract defines the functions set and get that can be used to modify or retrieve the value of the variable.

To access a member (like a state variable) of the current contract, you do not typically add this. prefix, you just access it directly via its name. Unlike in some other languages, omitting it is not just a matter of style, it results in a completely different way to access the member, but more on this later.

This contract does not do much yet apart from (due to the infrastructure built by Ethereum) allowing anyone to store a single number that is accessible by anyone in the world without a (feasible) way to prevent you from publishing this number. Anyone could call set again with a different value and overwrite your number, but the number is still stored in the history of the blockchain. Later, you will see how you can impose access restrictions so that only you can alter the number.

B. Blockchains Basics

Blockchains as a concept are not too hard to understand for programmers. The reason is that most of the complications (mining, hashing, elliptic-curve cryptography, peer-to-peer networks, etc.) are just there to provide a certain set of features and promises for the platform. Once you accept these features as given, you do not have to worry about the underlying technology - or do you have to know how Amazon's AWS works internally to use it? Transactions - A blockchain is a globally shared, transactional database. This means that everyone can read entries in the database just by participating in the network. If you want to change something in the database, you have to create a so-called transaction that has to be accepted by all others. The word transaction implies that the change you want to make (assume you want to change two values at the same time) is either not done at all or completely applied. Furthermore, while your transaction is being applied to the database, no other transaction can alter it. As an example, imagine a table that lists the balances of all accounts in an electronic currency. If a transfer from one account to another is requested, the transactional nature of the database ensures that if the amount is subtracted from one account, it is always added to the other account. If due to whatever reason, adding the amount to the target account is not possible, the source account is also not modified. Furthermore, a transaction is always cryptographically signed by the sender (creator). This makes it straightforward to guard access to specific modifications of the database. In the example of electronic currency, a simple check ensures that only the person holding the keys to the account can transfer money from it. Blocks - One major obstacle to overcome is what (in Bitcoin terms) is called a "double-spend attack": What happens if two transactions exist in the network that both want to empty an account? Only one of the transactions can be valid, typically the one that is accepted first. The problem is that "first" is not an objective term in a peer-to-peer network. The abstract answer to this is that you do not have to care. A globally accepted order of the transactions will be selected for you, solving the conflict. The transactions will be bundled into what is called a "block" and then they will be executed and distributed among all participating nodes. If two transactions contradict each other, the one that ends up being second will be rejected and not become part of the block. These blocks form a linear sequence in time and that is where the word "blockchain" derives from. Blocks are added to the chain in rather regular intervals - for Ethereum, this is roughly every 17 seconds. As part of the "order selection mechanism," (which is called "mining") blocks may be reverted from time to time, but only at the "tip" of the chain. The more blocks are added on top of a particular block, the less likely this block will be reverted. So it might be that your transactions are reverted and even removed from the blockchain, but the longer you wait, the less likely it will be [10]

C. The Ethereum Virtual Machine

Ethereum is a system first introduced at the North American Bitcoin Conference by Ethereum founder Vitalik Buterin. Although it is seen as an alt coin, Ethereum is an innovative system that aims to develop blockchain technology and use it in more areas. Buterin has changed the certificate used by Blockchain. And he developed the blockchain-based software Ethereum using the SHA-256 certificate. Ethereum made it possible to produce documents consisting of codes, called smart contracts. To explain shortly and understandably; as a function, smart contracts are non-intermediary systems that express the autonomous (so by itself) performance of the contractual acts without any additional discretion of the parties, in the event that the contractual terms agreed upon by the parties are fulfilled. Smart contracts are a computer protocol that enables the exchange of money, real estate, or any asset that can be traded without any intermediary. All transactions performed in these contracts are converted into computer code, stored, and controlled by users on the blockchain network. In addition, smart contracts provide accounting feedback of transactions such as money transfers, purchases, and sales of products or services. E.g.; if an individual who wants to buy a house uses the smart contract protocol in this transaction, the first thing to do is to create a digital contract with the seller on the blockchain network with the terms of purchase and sale. When a trigger transaction (payment, expiration, etc.) is initiated that is coded into the contract, the contract runs itself and the transaction is recorded in the blockchain network. At the end of the transaction, digital receipts are added to be kept in the agreements of the parties and the shopping is completed [11]

The Ethereum Virtual Machine or EVM is the runtime environment for smart contracts in Ethereum. Code running inside the EVM has no access to the network, file system, or other processes. Smart contracts even have limited access to other smart contracts. There are two kinds of accounts in Ethereum which share the same address space: **External accounts** that are controlled by public-private key pairs (i.e. humans) and **contract accounts** that are controlled by the code stored together with the account. The address of an external account is determined from the public key while the address of a contract is determined at the time the contract is created. Every account has a persistent key-value store mapping 256-bit words to 256-bit words called **storage**.

Furthermore, every account has a **balance** in Ether which can be modified by sending transactions that include Ether. A transaction is a message that is sent from one account to another account. It can include binary data (which is called "payload") and Ether. Upon creation, each transaction is charged with a certain amount of **gas**, whose purpose is to limit the amount of work that is needed to execute the transaction and to pay for this execution at the same time. While the EVM executes the transaction, the gas is gradually depleted according to specific rules.

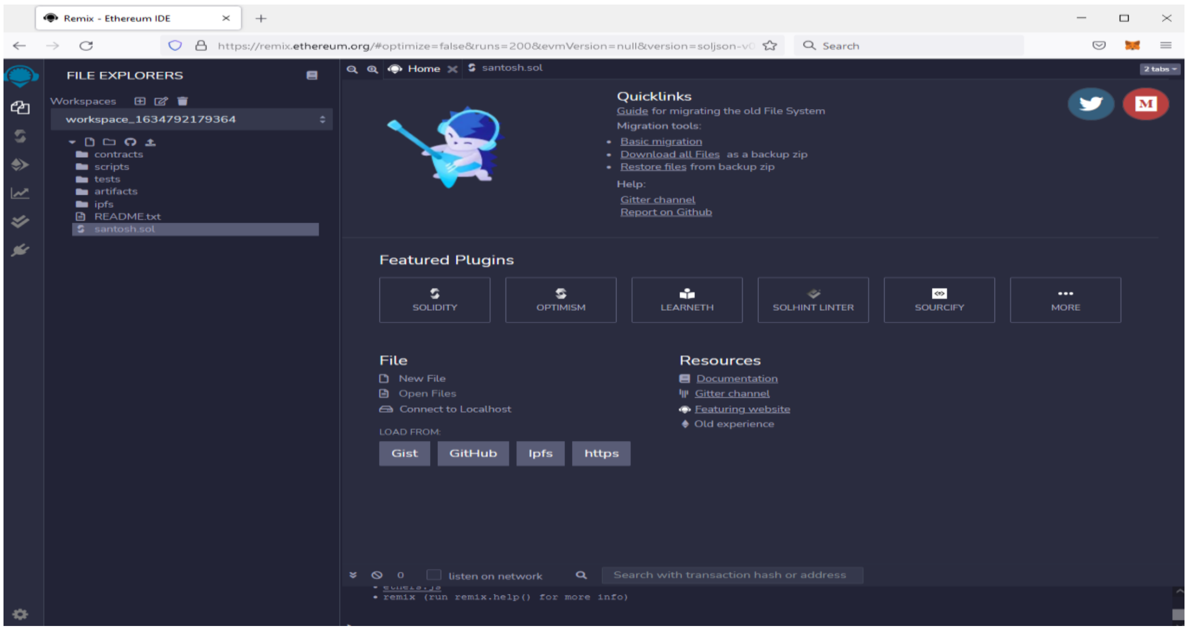
DOI: 10.17148/IJARCCE.2023.12253

The **gas price** is a value set by the creator of the transaction, If some gas is left after the execution, it is refunded to the creator in the same way. If the gas is used up at any point (i.e. it would be negative), an out-of-gas exception is triggered, which reverts all modifications made to the state in the current call frame.

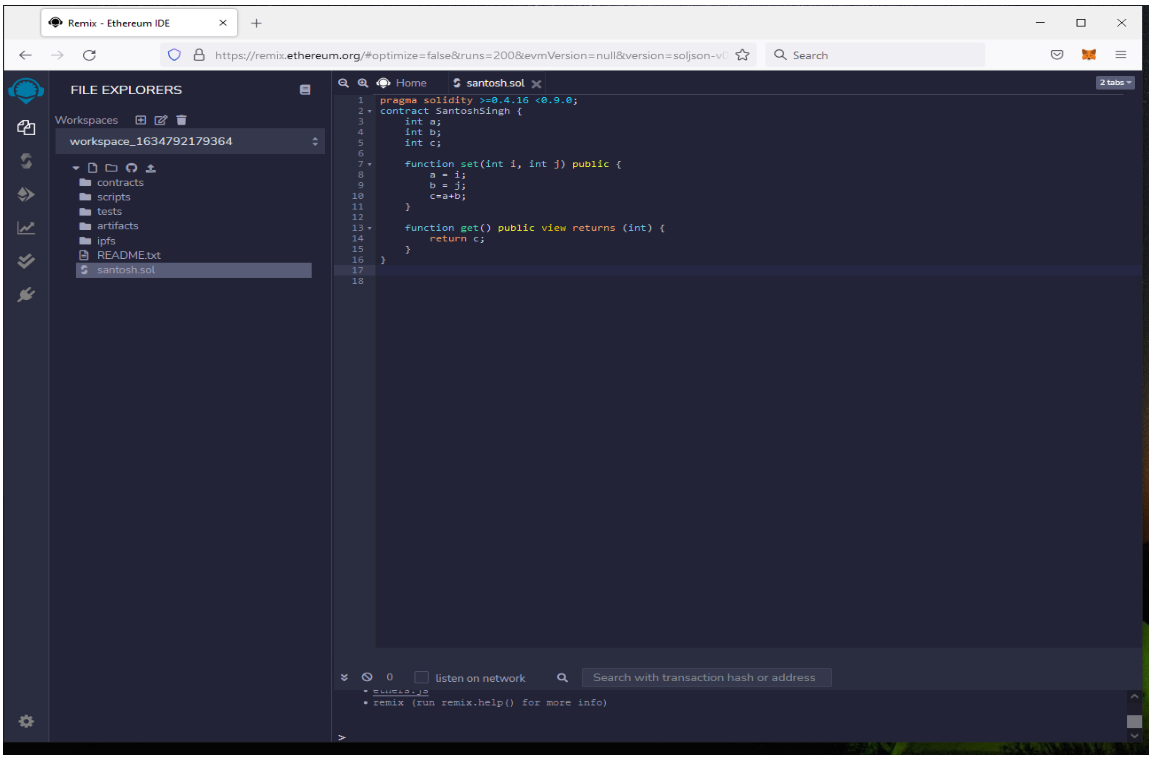
IV. STEPS TO EXECUTE SOLIDITY SMART CONTRACT USING REMIX IDE

Remix IDE is generally used to compile and run Solidity smart contracts.

Step 1: Open Remix IDE on any of your browsers, select the New File and click on Solidity to choose the environment.

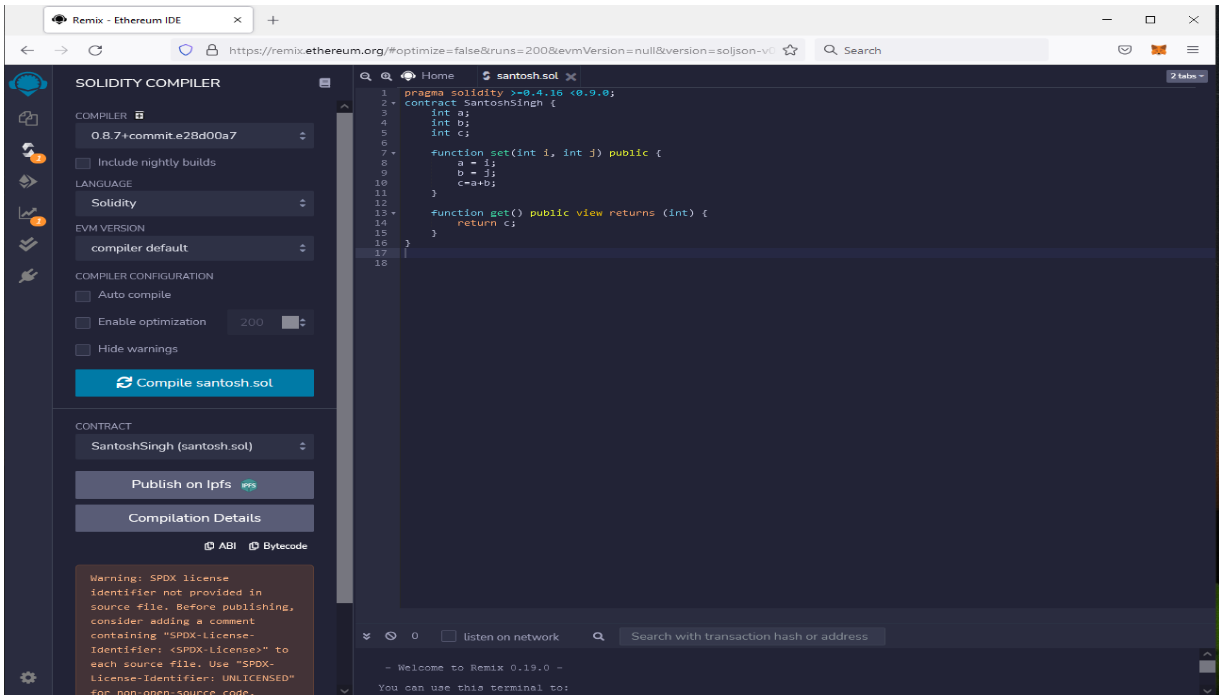


Step 2: Write the Smart contract in the code section.


Remix - Ethereum IDE
https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0
FILE EXPLORERS
Workspaces
workspace_1634792179364
contracts
scripts
tests
artifacts
ipfs
README.txt
santosh.sol
Home
santosh.sol
listen on network
Search with transaction hash or address
• remix (run remix.help() for more info)

```
pragma solidity >=0.4.16 <0.9.0;
contract SantoshSingh {
    int a;
    int b;
    int c;

    function set(int i, int j) public {
        a = i;
        b = j;
        c=a+b;
    }

    function get() public view returns (int) {
        return c;
    }
}
```

Step 3: Click the Compile button under the Compiler window to compile the contract.


Remix - Ethereum IDE
https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v
Search
Home
santosh.sol
2 tabs

SOLIDITY COMPILER

COMPILER: 0.8.7+commit.e28d00a7
Include nightly builds
LANGUAGE: Solidity
EVM VERSION: compiler default
COMPILER CONFIGURATION
Auto compile
Enable optimization 200
Hide warnings

Compile santosh.sol

CONTRACT: SantoshSingh (santosh.sol)
Publish on Ipfs
Compilation Details
ABI Bytecode

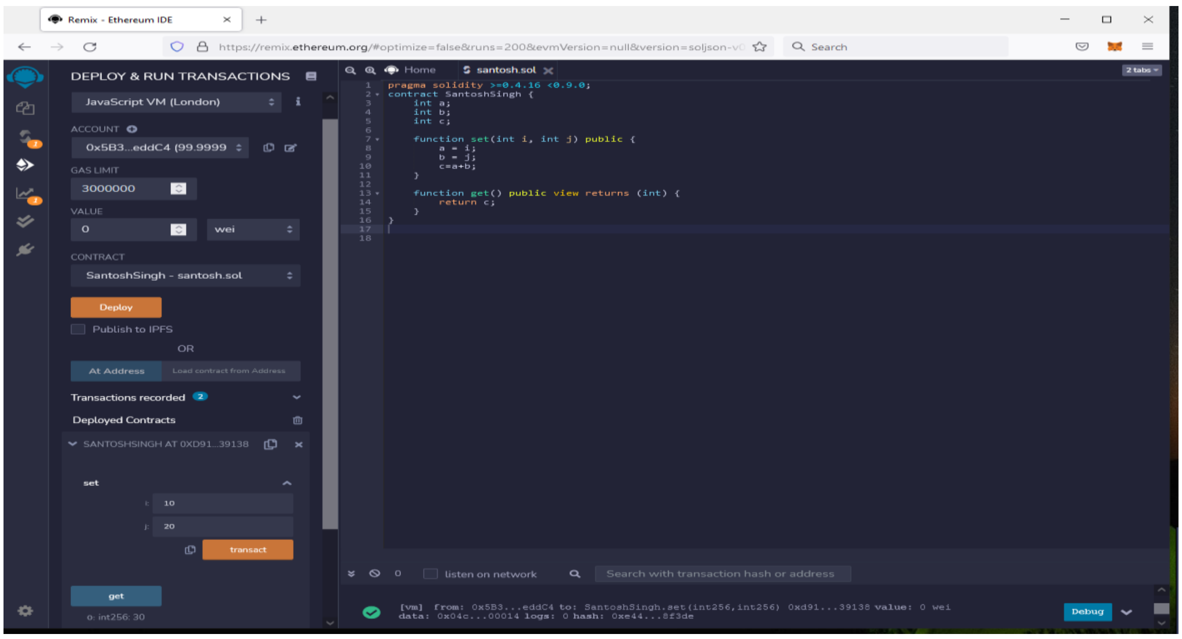
```
Warning: SPDX license
identifier not provided in
source file. Before publishing,
consider adding a comment
containing "SPDX-License-
Identifier: <SPDX-License>" to
each source file. Use "SPDX-
License-Identifier: UNLICENSED"
for non-open-source code.
```

```
pragma solidity >=0.4.16 <0.9.0;
contract SantoshSingh {
    int a;
    int b;
    int c;

    function set(int i, int j) public {
        a = i;
        b = j;
        c=a+b;
    }

    function get() public view returns (int) {
        return c;
    }
}
```

listen on network   Search with transaction hash or address

```
- Welcome to Remix 0.19.0 -

You can use this terminal to:
```

Step 4: To execute the code, click on the Deploy button under the Deploy and Run Transactions window. And After deploying the code click on the method calls under the drop-down of deployed contracts to run the program, and for output, check to click on the drop-down on the console.


Remix - Ethereum IDE
https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v
Search
Home
santosh.sol
2 tabs

DEPLOY & RUN TRANSACTIONS

JavaScript VM (London)
ACCOUNT: 0x5B3...eddC4 (99.9999
GAS LIMIT: 3000000
VALUE: 0 wei
CONTRACT: SantoshSingh - santosh.sol
Deploy
Publish to IPFS
OR
At Address   Load contract from Address
Transactions recorded 2
Deployed Contracts
SANTOSHSINGH AT 0XD91...39138
set
i: 10
j: 20
transact
get
0: int256: 30

```
pragma solidity >=0.4.16 <0.9.0;
contract SantoshSingh {
    int a;
    int b;
    int c;

    function set(int i, int j) public {
        a = i;
        b = j;
        c=a+b;
    }

    function get() public view returns (int) {
        return c;
    }
}
```

listen on network   Search with transaction hash or address

```
[vm] from: 0x5B3...eddC4 to: SantoshSingh.set(int256,int256) 0xd91...39138 value: 0 wei
data: 0x04c...00014 logs: 0 hash: 0xe44...8f3de
```

Debug

V. CONCLUSION

The upcoming of civilization is virtual i.e. digital, how to transmit the physical society's relationships into digital relationships in the virtual domain is a huge challenge in IT technologies. The idea of the smart contract is one of the simple concepts to resolve the code contract projected by Nick Szabo in 1994. The dealings in smart contracts are carried out with computer codes. This is significant for the improvement and development of technology law to understand the logic of smart contracts, where codes are law.

DOI: 10.17148/IJARCCE.2023.12253

In this article, we have discussed the Smart contract, Solidity programming using REMIX, EVM, and the basics of Blockchain. The main purpose of this article is to simplify the Solidity language so that everyone can understand it and to write a contract in that language. We have demonstrated with a simple example that, how to write, compile and execute solidity smart contracts using Remix IDE.

REFERENCES

[1]. Szabo, N. (1994). Smart contracts.
[2]. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.
[3]. Szabo, N. (1997). The idea of smart contracts. Nick Szabo's papers and concise tutorials, 6(1).
[4]. Miller, M. S., Morningstar, C., & Frantz, B. (2000, February). Capability-based financial instruments. In International Conference on Financial Cryptography (pp. 349-378). Springer, Berlin, Heidelberg.
[5]. Swanson, T. (2015). Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. The report is available online.
[6]. https://retreeb.medium.com/blockchain-and-csr-emergence-of-a-social-smart-contract-d14dd9aef8e4
[7]. Singh, A., Parizi, R. M., Zhang, Q., Choo, K. K. R., & Dehghantanha, A. (2020). Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities. Computers & Security, 88, 101654.
[8]. Huang, Y., Bian, Y., Li, R., Zhao, J. L., & Shi, P. (2019). Smart contract security: A software lifecycle perspective. IEEE Access, 7, 150184-150202.
[9]. https://docs.soliditylang.org/en/v0.8.9/
[10]. Singh, S. K., Manjhi, P. K., & Tiwari, R. K. (2021). Cloud Computing Security Using Blockchain Technology. In Transforming Cybersecurity Solutions using Blockchain (pp. 19-30). Springer, Singapore.
[11]. https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

BIOGRAPHY

Authors **Dr. Santosh Kumar Singh** is an Assistant Professor at Don Bosco Institute of Technology, GGSIPU, New Delhi. He has done Ph.D. from Vinoba Bhave University, Jharkhand. He has over 18 years of teaching experience and published more than 23 research papers in IEEE, Springer, and Scopus-indexed journals. His research work has been listed in H-INDEX as well as in I-INDEX also. His patent was published on 10th Feb 23 and the Title of the Invention is "Improvement of an Intelligent Transportation System by Using Blockchain Technology". He has written books on Data Structures, and Programming Using C. Also written guidebooks on Artificial Intelligence and Web Based Programming (PHP) published by AKASH Books, New Delhi. He is also serving as an academic counselor of (MCA/BCA) at IGNOU since 2006. In 2020 he was appointed as a reviewer at IGI Global Scopus Indexed Journals. He has reviewed the syllabus course titled "Web Technology" Amity University Greater Noida, and also reviewed the Springer Singapore book titled " Transforming Cyber Security Solutions Using Blockchain Technology". He was the primary evaluator of Toycathon 2021, an initiative of the Ministry of Education's Innovation Cell. He is a member of the (IAENG) International Association for Engineers, Hong Kong. His research interest areas are Blockchain, Cloud Computing, & Parallel and Distributed Computing.