COMPBEB1326
72017912C

GROUP C : ASSIGNMENT No. 3

Page :
Date : / /

Title: Smart Contract in Blockchain.

Objective: To understand what & are smart contracts in blockchain & how to write them.

Problem Statement:

Write a smart contract on a test network, for Bank account of a customer for following operations:

1. Deposit money
2. Withdraw money
3. Show balance

Software & Hardware Requirement:

1) Desktop / Laptop
2) Any Operating System
3) Internet Connection
4) IDE

Theory:

Smart Contract:

A smart contract (or cryptocontract) is a computer program that directly & automatically controls the transfer of digital assets between the parties under certain conditions. A smart contract works in the same way as a traditional contract while also automatically enforcing the contract.

Smart contracts are programs that execute exactly as they are set up (coded, programmed) by their creators. Just like a traditional contract is enforceable by law,

smart contracts are enforceable by code.

History of Smart Contract:

In 1994, Nick Szabo, a legal scholar, & a cryptographer recognized the application of a decentralized ledger for smart contracts. He theorized that these contracts could be written in code which can be stored & replicated on the system & supervised by the network of computers that constitute the blockchain. These smart contracts could also help in transferring digital assets between the parties under certain conditions.

Features of Smart Contracts:

1) Distributed
2) Deterministic
3) Immutable
4) Autonomy
5) Customizable
6) Transparent
7) Trustless
8) Self Verifying
9) Self Enforcing.

Capabilities of Smart Contract:

1) Accuracy
2) Automation
3) Speed

4) Backup
5) Security
6) Savings
7) Manages Information
8) Multi-signature accounts.

Example Use Cases:

1) Smart contracts provide utility to other contracts. For example, consider a smart contract that transfers funds to party A after 10 days. After 10 days, the above mentioned smart contract will execute another smart contract which checks if the required funds are available at the source account (let's say party B)
2) They facilitate the implementation of 'multi-signature' accounts, in which the assets are transferred only when a certain percentage of people agree to do so.
3) Smart contracts can map legal obligations into an automated process.
4) If smart contracts are implemented correctly, can provide a greater degree of contractual security.

Challenges of Smart Contracts:

1) No regulations: A lack of international regulations focusing on blockchain technology (and related technology like smart contracts, mining & use cases like cryptocurrency) makes these technologies difficult to oversee.

2) Difficult to implement: Smart contracts are also complicated to implement because it's still a relatively

new concept & research is still going on to understand the smart contract & its implications fully.

3) Immutable : They are practically immutable. Whenever there is a change that has to be incorporated into contract, a new contract has to be made & implemented in the blockchain.

4) Alignment : Smart contracts can speed the execution of the process that span multiple parties irrespective of the fact whether the smart contracts are in alignment with all the parties' intention & understanding.

Conclusion:

Successfully wrote a smart contract on a test network.

17/10/22

Code:

```
pragma solidity 0.6.6;

contract BankContract {
    struct client_account{
        int client_id;
        address client_address;
        uint client_balance_in_ether;
    }

    client_account[] clients;

    int clientCounter;
    address payable manager;
    mapping(address => uint) public interestDate;
    modifier onlyManager() {
    require(msg.sender == manager, "Only manager can call this!");
    _;
    }

    modifier onlyClients() {
        bool isclient = false;
        for(uint i=0;i<clients.length;i++){
        if(clients[i].client_address ==  msg.sender){
            isclient = true;
            break;
        }
    }

    require(isclient, "Only clients can call this!");
    _;
    }

    constructor() public{
        clientCounter = 0;
    }

    receive() external payable { }

    function setManager(address managerAddress) public returns(string memory){
        manager = payable(managerAddress);
        return "";
    }

    function joinAsClient() public payable returns(string memory){
        interestDate[msg.sender] = now;
        clients.push(client_account(clientCounter++,
        msg.sender, address(msg.sender).balance));
        return "";
    }

    function deposit() public payable onlyClients{
        payable(address(this)).transfer(msg.value);
```

```
    }

    function withdraw(uint amount) public payable onlyClients{
        msg.sender.transfer(amount * 1 ether);
    }

    function sendInterest() public payable onlyManager{
        for(uint i=0;i<clients.length;i++){
            address initialAddress =
            clients[i].client_address;
            uint lastInterestDate =
            interestDate[initialAddress];
            if(now < lastInterestDate + 10
            seconds){
            revert("It's just been less than 10 seconds!");
        }
        payable(initialAddress).transfer(1 ether);
        interestDate[initialAddress] = now;
    }

    }
    function getContractBalance() public view returns(uint){
        return address(this).balance;
    }
}
```

Output: