Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ Om Shete of Computer Department, Semester VII with Roll No. 2103163 has completed a course of the necessary experiments in the subject Blockchain under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the year 2024 - 2025

14/10/24

Teacher In-Charge

Head of the Department

Principal

Date 14/10/2024

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CONTENTS

15/07/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Om Shete
G1_07
2103163

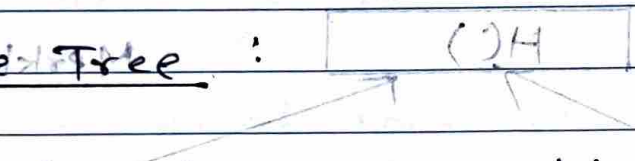
Experiment No : 1

Aim : Write a program to implement Merkle tree showing the cryptographic hash function in Blockchain.

Theory :

- A hash tree is also known as Merkle Tree. It is a tree in which each leaf node is labeled with hash value of data block and each non-leaf node is labeled with the hash value of its child node labels.

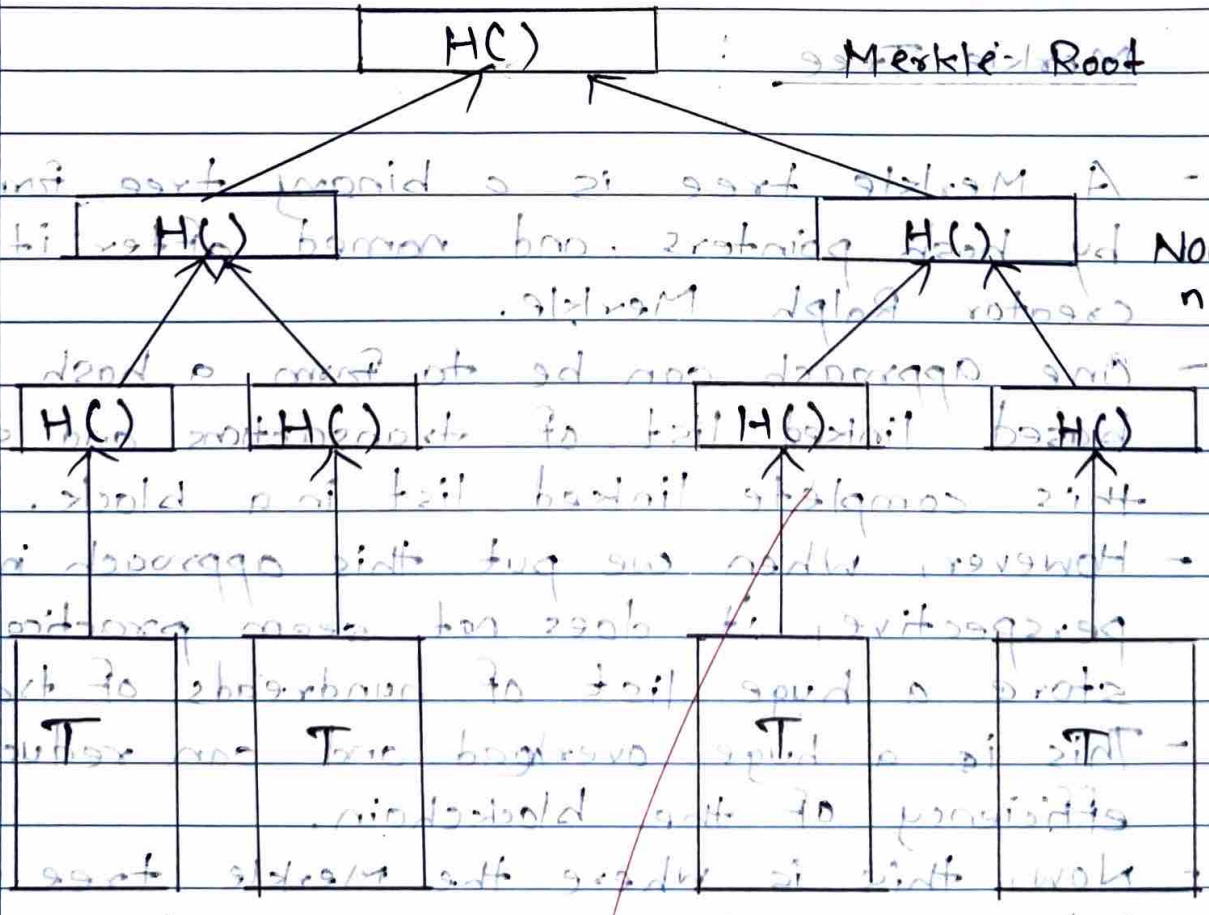
- **Merkle Tree** :



- A Merkle tree is a binary tree formed by hash pointers, and named after its creator Ralph Merkle.
- One approach can be to form a hash pointer-based linked list of transactions and store this complete linked list in a block.
- However, when we put this approach into perspective, it does not seem practical to store a huge list of hundreads of transactions
- This is a huge overhead and can reduce the efficiency of the blockchain.
- Now, this is where the Merkle tree comes into the picture. Merkle tree is a per-block tree of all the transactions that are included in the block.

- It allows us to hash of all transactions and proof of membership in a time-efficient manner.
- Hence, the blockchain is a hash-based linked list of blocks, where each block consists of a header and transactions.
- The transactions are arranged in a tree-like fashion, known as Merkle Tree.

Merkle Tree Structure



H() : Hash value
T : transaction

- A blockchain can potentially have thousands of blocks with thousands of transactions in each block. Therefore, memory space and computing power are two main challenges.
- It could be optimal to use as little data as possible for verifying transactions, which can reduce CPU processing and provide better security and this is exactly what Merkle tree offers.
- In a Merkle tree, transactions are grouped into pairs. The hash is computed for each pair and this is stored in the parent node.
- Now the parent nodes are grouped into pairs and their hash is stored in one level up in the tree. This continues till the root of the tree.
- The different types of nodes in Merkle tree is stored in root :-

1) Root node: The root of Merkle tree is known as the Merkle root and Merkle root is stored in the header of the block.

2) Leaf node: The leaf node contains the hash value of transaction data. Each transaction in the blocks has its data hashed and then this hash value is

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

stored in leaf nodes.

3) Non-leaf Node : The non-leaf nodes contain the hash value of transaction data of their respective children.

- Bitcoin uses the SHA-256 hash function to hash the transaction data continuously till the Merkle root is obtained.
- Further, a Merkle tree is binary in nature.

A+

29/7/24

Code:

```python
from typing import List
import hashlib


class Node:
    def __init__(self, left, right, value: str, content, is_copied=False) ->
        None:
        self.left: Node = left
        self.right: Node = right
        self.value = value
        self.content = content
        self.is_copied = is_copied

    @staticmethod
    def hash(val: str) -> str:
        return hashlib.sha256(val.encode('utf-8')).hexdigest()

    def __str__(self):
        return (str(self.value))

    def copy(self):
        """
        class copy function
        """
        return Node(self.left, self.right, self.value, self.content, True)

```

```
26
27  class MerkleTree:
28      def __init__(self, values: List[str]) -> None:
29          self.__buildTree(values)
30
31      def __buildTree(self, values: List[str]) -> None:
32
33          leaves: List[Node] = [Node(None, None, Node.hash(e), e)
34                                for e in values]
35          if len(leaves) % 2 == 1:
36
37              leaves.append(leaves[-1].copy())
38          self.root: Node = self.__buildTreeRec(leaves)
39
40      def __buildTreeRec(self, nodes: List[Node]) -> Node:
41          if len(nodes) % 2 == 1:
42              # duplicate last elem if odd number of elements
43              nodes.append(nodes[-1].copy())
44          half: int = len(nodes) // 2
45
46          if len(nodes) == 2:
47              return Node(nodes[0], nodes[1], Node.hash(nodes[0].value + nodes[1]
                    .value), nodes[0].content+"+"+nodes[1].content)
48
49          left: Node = self.__buildTreeRec(nodes[:half])
50          right: Node = self.__buildTreeRec(nodes[half:])
51
```

```
        value: str = Node.hash(left.value + right.value)
        content: str = f'{left.content}+{right.content}'
        return Node(left, right, value, content)

    def printTree(self) -> None:
        self.__printTreeRec(self.root)

    def __printTreeRec(self, node: Node) -> None:
        if node != None:
            if node.left != None:
                print("Left: "+str(node.left))
                print("Right: "+str(node.right))
            else:
                print("Input")

            if node.is_copied:
                print('(Padding)')
            print("Value: "+str(node.value))
            print("Content: "+str(node.content))
            print("")
            self.__printTreeRec(node.left)
            self.__printTreeRec(node.right)

    def getRootHash(self) -> str:
        return self.root.value
```

```


def mixmerkletree() -> None:
    elems = ["Aztec", "A", "Team",
             "Om", "Hamza", "Mohib", "Parth"]

    print("Inputs: ")
    print(*elems, sep=" | ")
    print("")
    mtree = MerkleTree(elems)
    print("Root Hash: "+mtree.getRootHash()+"\n")
    mtree.printTree()


mixmerkletree()
```

Output:

```
Inputs:
Aztec | A | Team | Om | Hamza | Mohib | Parth

Root Hash: 701fa83df0163756a4c8bd11ab5e69f5a763a386e29cb79265ec29ef840f4685

Left: 5a806abd172de56e2d43ebe13aece485e54f600361901219d788d0750913837b
Right: 545cf9a5a92db7affd5a4a20a4168337e485d70a195e13e035dc793923c8c7ab
Value: 701fa83df0163756a4c8bd11ab5e69f5a763a386e29cb79265ec29ef840f4685
Content: Aztec+A+Team+Om+Hamza+Mohib+Parth+Parth

Left: 7ef6c3416f199dde05fa2eec22864ecf829b7fb1f7fc97075e9e4958febc40ad
Right: c527ffbf4fe41d3285a20237c97b2a0f02accfecbd66dc42555f62f10d3a95c1
Value: 5a806abd172de56e2d43ebe13aece485e54f600361901219d788d0750913837b
Content: Aztec+A+Team+Om

Left: d28a9d9c11188360cc63bceaeff700835526a239329056daffa0cf1a6a856d4c
Right: 559aead08264d5795d3909718cdd05abd49572e84fe55590eef31a88a08fdffd
Value: 7ef6c3416f199dde05fa2eec22864ecf829b7fb1f7fc97075e9e4958febc40ad
Content: Aztec+A

Input
Value: d28a9d9c11188360cc63bceaeff700835526a239329056daffa0cf1a6a856d4c
Content: Aztec

Input
Value: 559aead08264d5795d3909718cdd05abd49572e84fe55590eef31a88a08fdffd
```

```
Input
Value: 559aead08264d5795d3909718cdd05abd49572e84fe55590eef31a88a08fdffd
Content: A

Left: 5985039f106df054b43c3529139613bf7c2372937539f344c7c5ee47d00a0d63
Right: 3c815b4770735e66231e08cb8865b50308ba7301c579d38c171f4ebec5e3e9fc
Value: c527ffbf4fe41d3285a20237c97b2a0f02accfecbd66dc42555f62f10d3a95c1
Content: Team+Om

Input
Value: 5985039f106df054b43c3529139613bf7c2372937539f344c7c5ee47d00a0d63
Content: Team

Input
Value: 3c815b4770735e66231e08cb8865b50308ba7301c579d38c171f4ebec5e3e9fc
Content: Om

Left: dcd78e1ce4bea5aa05d146cc658bafff7bdfff75d7f19042ea90b9b4382618f0
Right: 2c3b263e732b8c8a97c01b960d3e9e84e0fb7283a035f87d7dc1152d3468d8b0
Value: 545cf9a5a92db7affd5a4a20a4168337e485d70a195e13e035dc793923c8c7ab
Content: Hamza+Mohib+Parth+Parth

Left: 26ac05af275df37b36a23a75dfb701c48ecbdb6a5a578674c36274737864f4ce
Right: f3c95de41483179952d2b4aef3811383ac2deae8c33e708e764d81856d29f5c3
Value: dcd78e1ce4bea5aa05d146cc658bafff7bdfff75d7f19042ea90b9b4382618f0
```

```
Left: dcd78e1ce4bea5aa05d146cc658bafff7bdfff75d7f19042ea90b9b4382618f0
Right: 2c3b263e732b8c8a97c01b960d3e9e84e0fb7283a035f87d7dc1152d3468d8b0
Value: 545cf9a5a92db7affd5a4a20a4168337e485d70a195e13e035dc793923c8c7ab
Content: Hamza+Mohib+Parth+Parth

Left: 26ac05af275df37b36a23a75dfb701c48ecbdb6a5a578674c36274737864f4ce
Right: f3c95de41483179952d2b4aef3811383ac2deae8c33e708e764d81856d29f5c3
Value: dcd78e1ce4bea5aa05d146cc658bafff7bdfff75d7f19042ea90b9b4382618f0
Content: Hamza+Mohib

Input
Value: 26ac05af275df37b36a23a75dfb701c48ecbdb6a5a578674c36274737864f4ce
Content: Hamza

Input
Value: f3c95de41483179952d2b4aef3811383ac2deae8c33e708e764d81856d29f5c3
Content: Mohib

Left: 4ec1a76282e4f89a809b90cea83ef509a5eda8d8d128819905a7459830a2ebe5
Right: 4ec1a76282e4f89a809b90cea83ef509a5eda8d8d128819905a7459830a2ebe5
Value: 2c3b263e732b8c8a97c01b960d3e9e84e0fb7283a035f87d7dc1152d3468d8b0
```

```
Input
Value: f3c95de41483179952d2b4aef3811383ac2deae8c33e708e764d81856d29f5c3
Content: Mohib

Left: 4ec1a76282e4f89a809b90cea83ef509a5eda8d8d128819905a7459830a2ebe5
Right: 4ec1a76282e4f89a809b90cea83ef509a5eda8d8d128819905a7459830a2ebe5
Value: 2c3b263e732b8c8a97c01b960d3e9e84e0fb7283a035f87d7dc1152d3468d8b0
Content: Parth+Parth

Input
Value: 4ec1a76282e4f89a809b90cea83ef509a5eda8d8d128819905a7459830a2ebe5
Content: Parth

Input
(Padding)
Value: 4ec1a76282e4f89a809b90cea83ef509a5eda8d8d128819905a7459830a2ebe5
Content: Parth


=== Code Execution Successful ===
```

Experiment No: 2

Aim: Create Smart contract using Solidity and Remix IDE.

Theory:

- Solidity is a brand-new programming language created by Ethereum which is the second-largest market of cryptocurrency by capitalization.
- Solidity is a high-level programming language designed for implementing smart contracts.
- It is a statically typed object-oriented language.
- Solidity is high influenced by Python, C++, and JavaScript which run on the Ethereum Virtual machine (EVM).
- Solidity supports complex user-defined programming, libraries, and inheritance.
- Solidity is the primary language for blockchains running platform.
- Solidity can be used to create contracts like voting, blind auctions, crowdfunding, multi-signatures, wallets, etc.

1. **Smart Contract:**

- Smart contracts are high-level programs codes that are compiled to EVM byte code and deployed to the ethereum blockchain for

further execution.
- It allows us to perform credible transaction without any interference of the third party, these transactions are trackable and irreversible.
- Languages used to write smart contracts are Solidity, Serpent, LLL and Mutan.

- **The contract Keyword:**

⇒ The contract keyword declares a contract under which is the code encapsulated.

```
contract Test
{
// Functions and Data
}
```

- **State Variable**

⇒ State variables are permanently stored in contract storage that they are written in Ethereum Blockchain.
- The line uint public var1 declares a state variable called var1 of type uint.
- Similarly, goes with the declaration uint public var2 and uint public sum.

```
uint public setData;
```

- A function declaration

⇒ This is a function named set of access modifier type public which takes a variable x of datatype uint as a parameter.

- It's syntax is :-

```
function set (uint x) public

function get() public view returns (uint)
```

- This is an example of a simple smart contract which updates the value of setData.
- Anyone can call the function set and overwrite this value of setData which is stored in Ethereum blockchain and there is possibly no way for anyone to stop someone from using this function.

- Smart Contracts

⇒ Smart contracts are the block of instruction that are not dependent on any third party or centralized database.

- They are executable in decentralized environment.
- The benefits of smart contracts are as follows :-

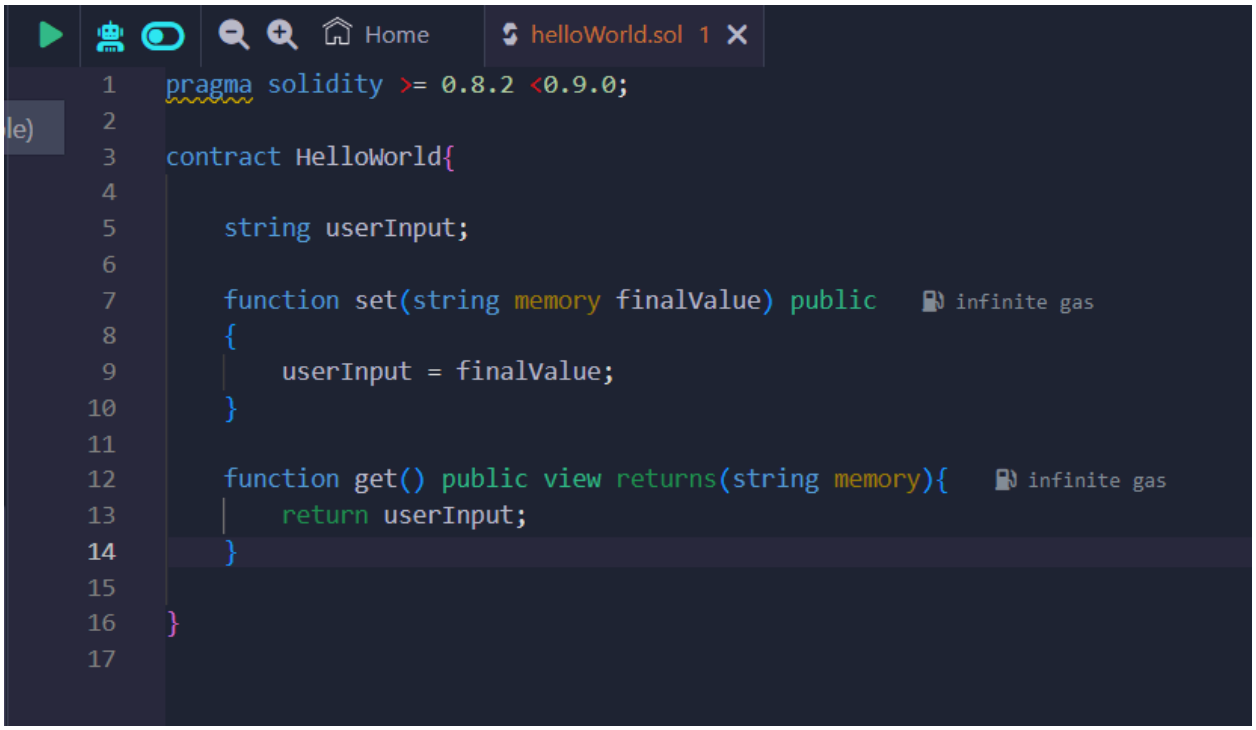
1) It removes centralized issues.
2) It is safe and more secure.

3) The terms and conditions are visible for the relevant parties. No way to change them.

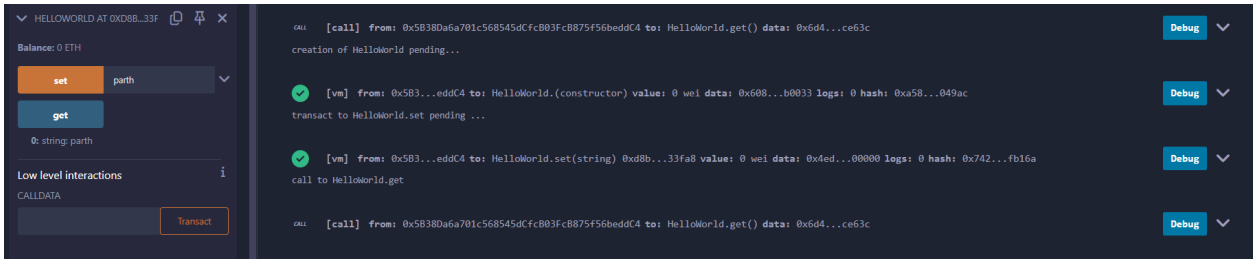
4) Makes the system more accurate.

3/10/24

Code:


Home
helloWorld.sol 1
pragma solidity >= 0.8.2 <0.9.0;
contract HelloWorld{
string userInput;
function set(string memory finalValue) public
infinite gas
{
userInput = finalValue;
}
function get() public view returns(string memory){
infinite gas
return userInput;
}
}

Output:


HELLOWORLD AT 0XD8B...33F
Balance: 0 ETH
set
parth
get
0: string: parth
Low level interactions
CALLDATA
Transact
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: HelloWorld.get() data: 0x6d4...ce63c
creation of HelloWorld pending...
[vm] from: 0x5B3...eddC4 to: HelloWorld.(constructor) value: 0 wei data: 0x608...b0033 logs: 0 hash: 0xa58...049ac
transact to HelloWorld.set pending ...
[vm] from: 0x5B3...eddC4 to: HelloWorld.set(string) 0xd8b...33fa8 value: 0 wei data: 0x4ed...00000 logs: 0 hash: 0x742...fb16a
call to HelloWorld.get
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: HelloWorld.get() data: 0x6d4...ce63c
Debug

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Om Shete
C1 07
2103163

Experiment NO: 3

Aim: Write a program in Solidity Programming to create Transaction using the Remix IDE

Theory:

- In blockchain, a transaction is a transfer of data or assets between participants.
- It involves recording information on the blockchain ledger, ensuring it is secure, transparent and immutable.
- Once a transaction is validated by the Network (through consensus mechanisms like pow / pos) it is grouped into a block and added to a blockchain, making it permanent part of themselves.
- Transactions are designed to be tamper proof and transparent.
- Each transactions typically includes details such as the sender, recepient, amount and a timestamp.
- After a transaction is initiated, it is broadcasted to the network where nodes verify its authenticity using cryptographic algorithms.
- Once validated, the transaction is included in a block, and the block is appended to the existing chain
- This process ensures that all participants in the network have a consistent and

accurate record of transaction, making fraud and double speeding difficult.
- The list of transaction is a component of the header.
- It forms a tree like structure called as Merkle tree.
- The Merkle root is used to construct the block hash.
- If you change a transaction, you need to change all the subsequent block hash.

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity >= 0.8.2 <0.9.0;
contract TransactionContract {
event TransactionMade(address indexed from, address indexed to, uint amount);
struct Transaction {
address from;
address to;
uint amount;
uint timestamp;
}
Transaction[] public transactions;
mapping(address => uint) public balances;
function createTransaction(address payable _to) public payable {
require(msg.sender.balance >= msg.value, "Insufficient balance");
transactions.push(Transaction(msg.sender, _to, msg.value, block.timestamp));
balances[msg.sender] -= msg.value;
balances[_to] += msg.value;
emit TransactionMade(msg.sender, _to, msg.value);
_to.transfer(msg.value);
}
function getAllTransactions() public view returns (Transaction[] memory) {
return transactions;
}
```

function getBalance(address _addr) public view returns (uint) {

return balances[_addr];

}

}

Output:

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Om Shete
CL 07
2103163

Experiment NO: 4

Aim: Write a program in Solidity to implement transactions by embedding of wallet

Theory:

- Blockchain is a decentralized digital ledger that records transactions across many computers so that the record cannot be altered actively.
- So that the record cannot be alter
- It operates in a peer-to-peer network where each participant maintains a copy of the ledger.
- This ledger is immutable and transparent ensures that once recorded.
- In this decentralized ecosystem, smart contracts are self executing agreements coded directly into the blockchain, automating and enforcing terms without intermediations.
- Smart contracts are self-executing contracts with the terms written directly into code
- They run on blockchain platforms like Ethereum.
- These contracts automatically enforces and executes terms of an agreement based on predefined rules and conditions

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

- Ethereum is a blockchain platform that enables developers to deploy and run smart contracts and decentralized applications.
- It has its own cryptocurrency ether (ETH) which is used to pay for transaction fees and computational services on the network.
- Events are used to log activities such as deposits, withdrawals and transfers enabling realtime updates for off-chains applications.
- Mapping stores users balances, allowing efficient data retrieval, functions define the contract's logic, including methods for depositing funds and transfering ether between addresses with built-in checks to ensure correctness & security.
- In context with blockchain a wallet refers to an Ethereum address that can hold and manage Ether (ETH) or other tokens.
- Transactions in Ethereum involve sending ether or tokens from one address to another.
- Each transaction is signed by the sender's private key and is validated by the network.

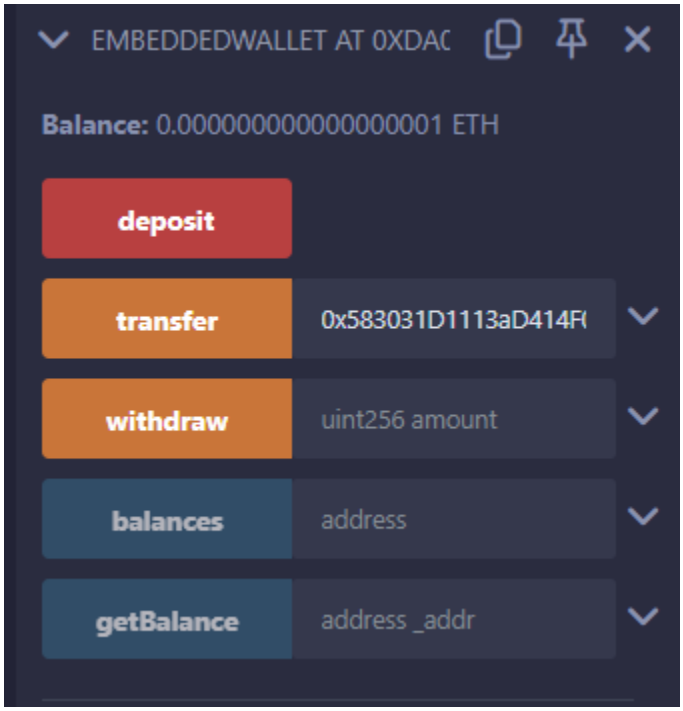
10/24

Code:

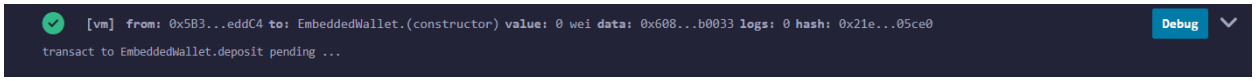
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract EmbeddedWallet {
event Deposit(address indexed user, uint amount);
event Withdrawal(address indexed user, uint amount);
event Transfer(address indexed from, address indexed to, uint amount);
mapping(address => uint) public balances;
modifier hasEnoughFunds(uint amount) {
require(balances[msg.sender] >= amount, "Insufficient funds");
_;
}
function deposit() public payable {
require(msg.value > 0, "Deposit must be greater than 0");
balances[msg.sender] += msg.value;
emit Deposit(msg.sender, msg.value);
}
function withdraw(uint amount) public hasEnoughFunds(amount) {
balances[msg.sender] -= amount;
payable(msg.sender).transfer(amount);
emit Withdrawal(msg.sender, amount);
}
function transfer(address to, uint amount) public hasEnoughFunds(amount) {
require(to != address(0), "Invalid recipient address");
```

```
require(to != msg.sender, "Cannot transfer to yourself");

balances[msg.sender] -= amount;

balances[to] += amount;

emit Transfer(msg.sender, to, amount);

}

function getBalance(address _addr) public view returns (uint) {

return balances[_addr];

}

}
```

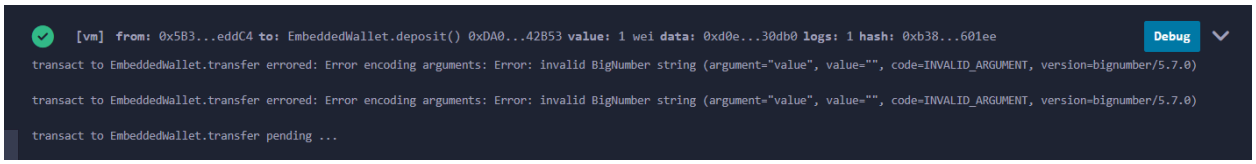
Output:



Deposit function -



Transfer function -

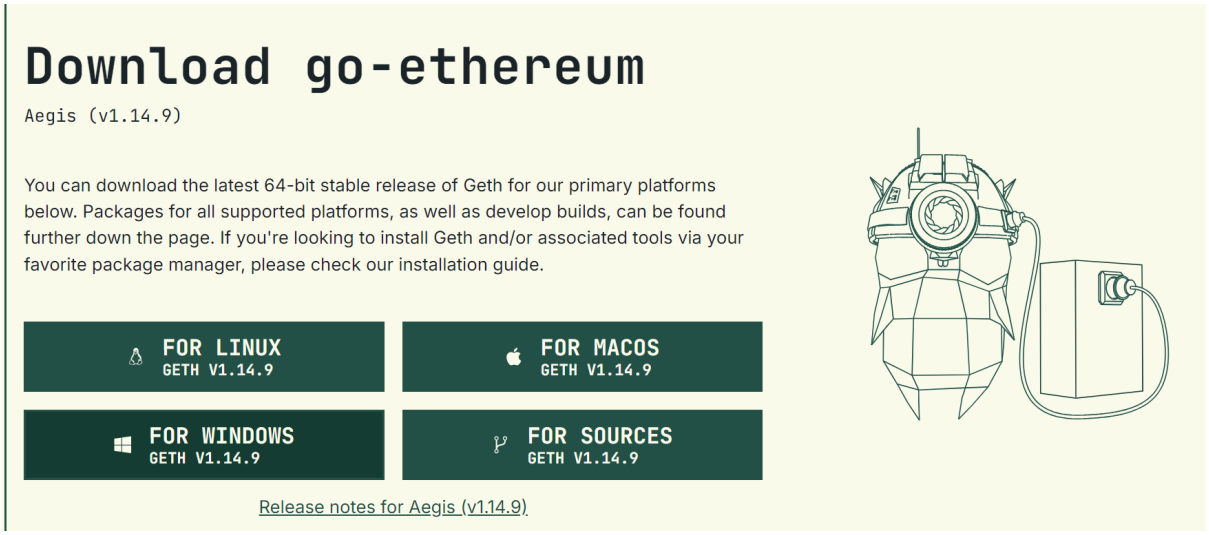


Experiment 5

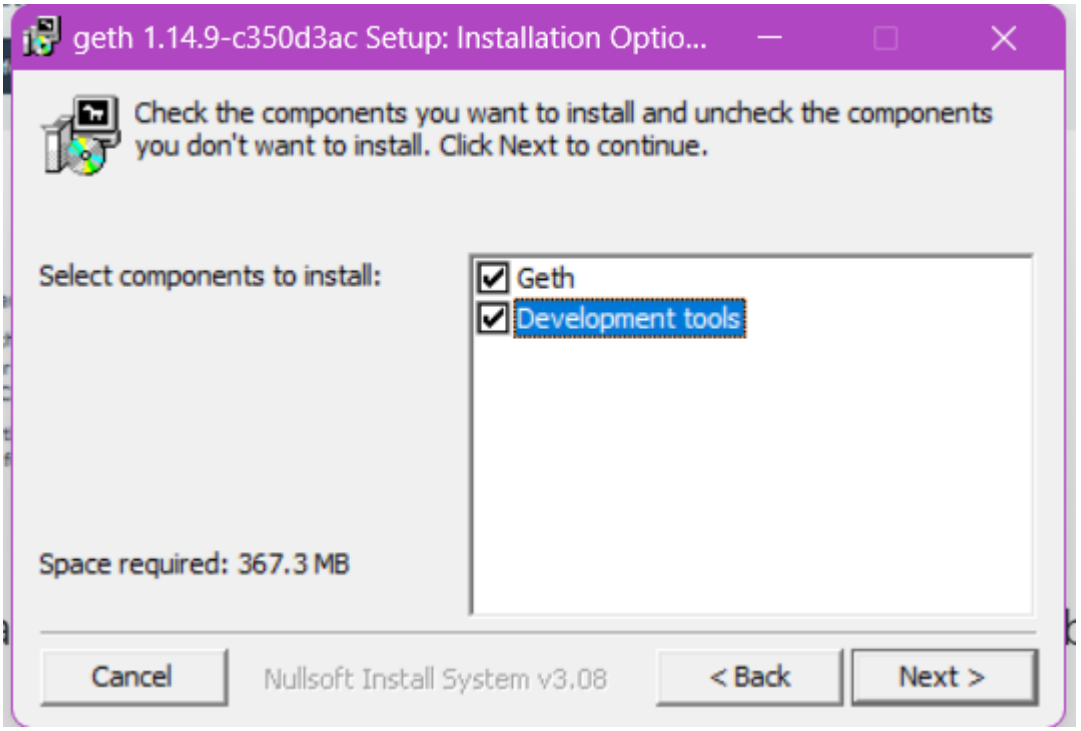
Aim – Implementation of the blockchain platform Ethereum using Geth

Code and Output

Step 1: Install Geth on Your System



While installing Geth make sure to select both checkboxes as shown below.
Select checkboxes

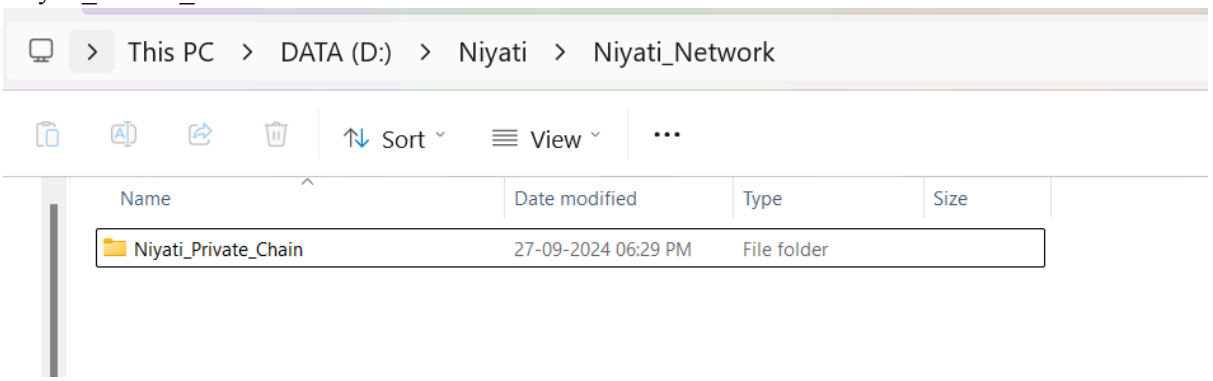


After installing Geth on your system open PowerShell or command prompt and type geth and press enter, the following output will be displayed.

```
PS C:\Users\Niyati Savant> cd D:\Niyati
PS D:\Niyati> .\geth.exe
INFO [09-27|18:27:17.710] Starting Geth on Ethereum mainnet...
INFO [09-27|18:27:17.710] Bumping default cache on mainnet         provided=1024 updated=4096
INFO [09-27|18:27:17.713] Maximum peer count                       ETH=50 total=50
WARN [09-27|18:27:17.713] Sanitizing cache to Go's GC limits       provided=4096 updated=2692
INFO [09-27|18:27:17.719] Set global gas cap                       cap=50,000,000
INFO [09-27|18:27:17.719] Initializing the KZG library             backend=gokzg
INFO [09-27|18:27:17.750] Allocated trie memory caches             clean=403.00MiB dirty=673.00MiB
INFO [09-27|18:27:17.751] Defaulting to pebble as the backing database
INFO [09-27|18:27:17.751] Allocated cache and file handles         database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata" cache=1.31GiB handles=8192
INFO [09-27|18:27:17.791] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\chain" readonly=false
INFO [09-27|18:27:17.791] State schema set to default              scheme=path
ERROR[09-27|18:27:17.791] Head block is not reachable
INFO [09-27|18:27:17.791] Initialising Ethereum protocol           network=1 dbversion=<nil>
WARN [09-27|18:27:17.794] Sanitizing invalid node buffer size      provided=673.00MiB updated=256.00MiB
INFO [09-27|18:27:17.820] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\state" readonly=false
INFO [09-27|18:27:17.820] Writing default main-net genesis block
INFO [09-27|18:27:18.081]
INFO [09-27|18:27:18.081] ---------------------------------------------------------------------------------------------------
----------------------------------------------------------
INFO [09-27|18:27:18.081] Chain ID:  1 (mainnet)
```

Step 2: Create a Folder For Private Ethereum

Create a separate folder for this project. In this case, the folder is ‘Niyati_Network’.
Create a new folder inside the folder ‘Niyati_Network’ for the private Ethereum network as it keeps your Ethereum private network files separate from the public files. In this example folder is ‘Niyati_Private_Chain’.



Step 3: Create a Genesis Block

The blockchain is a distributed digital register in which all transactions are recorded in sequential order in the form of blocks. There are a limitless number of blocks, but there is always one separate block that gave rise to the whole chain i.e. the genesis block.

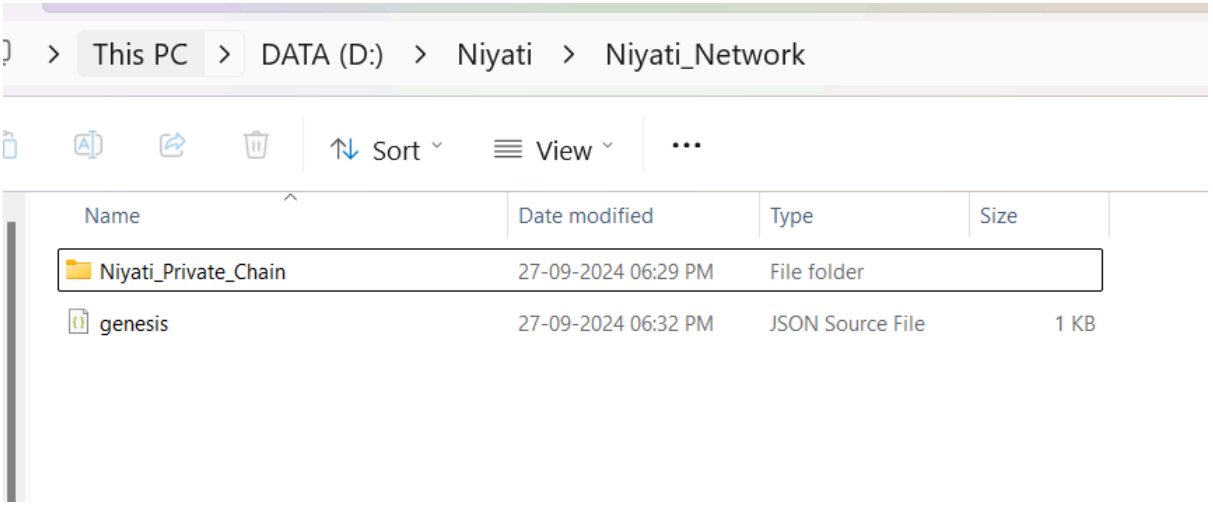
To create a private blockchain, a genesis blockis needed. To do this, create a genesis file, which is a JSON file with the following commands-

```
{

    “config”:{

      “chainId”:987,
```

```
    “homesteadBlock”:0,

    “eip150Block”:0,

    “eip155Block”:0,

    “eip158Block”:0

  },

  “difficulty”:”0x400″,

  “gasLimit”:”0x8000000″,

  “alloc”:{}

}
```


- config: It defines the blockchain configuration and determines how the network will work.
- chainId: This is the chain number used by several blockchains. The Ethereum main chain number is “1”. Any random number can be used, provided that it does not match with another blockchain number.
- homesteadBlock: It is the first official stable version of the Ethereum protocol and its attribute value is “0”.
- One can connect other protocols such as Byzantium, eip155B, and eip158. To do this, under the homesteadBlock add the protocol name with the Block prefix (for example, eip158Block) and set the parameter “0” to them.
- difficulty: It determines the difficulty of generating blocks. Set it low to keep the complexity low and to avoid waiting during tests.
- gasLimit: Gas is the “fuel” that is used to pay transaction fees on the Ethereum network. The more gas a user is willing to spend, the higher will be the priority of his transaction in the queue. It is recommended to set this value to a high enough level to avoid limitations during tests.
- alloc: It is used to create a cryptocurrency wallet for our private blockchain and fill it with fake ether. In this case, this option will not be used to show how to initiate mining on a private blockchain.

This file can be created by using any text editor and save the file with JSON extension in the folder Niyati_Network.



Step 4: Execute genesis file

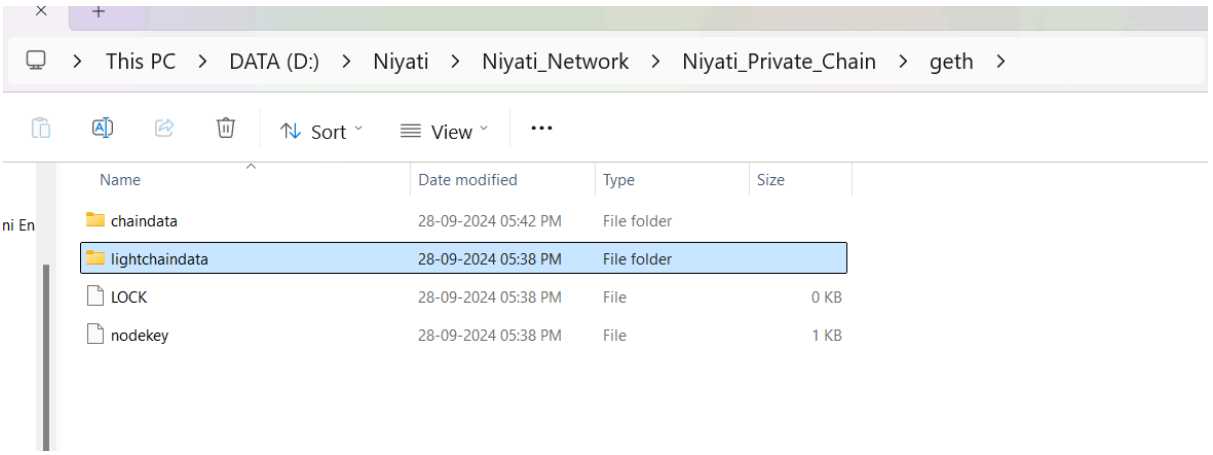
Open cmd or PowerShell in admin mode enter the following command-

geth –identity "yourIdentity" init \path_to_folder\CustomGenesis.json –datadir \path_to_data_directory\MyPrivateChain

```
PS D:\Niyati> .\geth.exe init "D:\Niyati\Niyati_Network\genesis.json"
INFO [09-27|21:20:35.462] Maximum peer count                       ETH=50 total=50
INFO [09-27|21:20:35.468] Set global gas cap                       cap=50,000,000
INFO [09-27|21:20:35.469] Initializing the KZG library             backend=gokzg
INFO [09-27|21:20:35.498] Using pebble as the backing database
INFO [09-27|21:20:35.498] Allocated cache and file handles         database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata" cache=16.00MiB handles=16
INFO [09-27|21:20:35.525] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\chain" readonly=false
INFO [09-27|21:20:35.525] State scheme set to already existing     scheme=path
INFO [09-27|21:20:35.538] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\state" readonly=false
```

Parameters-
path_to_folder- Location of Genesis file.
path_to_data_directory- Location of the folder in which the data of our private chain will be stored.
The above command instructs Geth to use the CustomGenesis.json file. After executing the above command Geth is connected to the Genesis file and it seems like this:

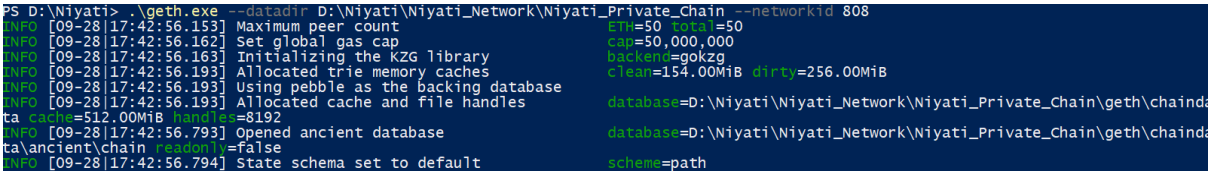
```
PS D:\Niyati> .\geth.exe --datadir .\Niyati_Network\Niyati_Private_Chain init .\Niyati_Network\genesis.json
INFO [09-28|17:38:10.904] Maximum peer count                       ETH=50 total=50
INFO [09-28|17:38:10.917] Set global gas cap                       cap=50,000,000
INFO [09-28|17:38:10.917] Initializing the KZG library             backend=gokzg
INFO [09-28|17:38:10.973] Defaulting to pebble as the backing database
INFO [09-28|17:38:10.973] Allocated cache and file handles         database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\chaindat
a cache=16.00MiB handles=16
INFO [09-28|17:38:11.782] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\chaindat
a\ancient\chain readonly=false
INFO [09-28|17:38:11.782] State schema set to default              scheme=path
ERROR[09-28|17:38:11.783] Head block is not reachable
INFO [09-28|17:38:12.287] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\chaindat
a\ancient\state readonly=false
INFO [09-28|17:38:12.287] Writing custom genesis block
INFO [09-28|17:38:12.652] Successfully wrote genesis state         database=chaindata hash=b052b0..1553c1
INFO [09-28|17:38:12.656] Defaulting to pebble as the backing database
INFO [09-28|17:38:12.656] Allocated cache and file handles         database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\lightcha
indata cache=16.00MiB handles=16
INFO [09-28|17:38:13.522] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\lightcha
indata\ancient\chain readonly=false
INFO [09-28|17:38:13.529] State schema set to default              scheme=path
ERROR[09-28|17:38:13.529] Head block is not reachable
INFO [09-28|17:38:14.048] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\lightcha
indata\ancient\state readonly=false
INFO [09-28|17:38:14.055] Writing custom genesis block
INFO [09-28|17:38:14.431] Successfully wrote genesis state         database=lightchaindata hash=b052b0..1553c1
PS D:\Niyati>
```



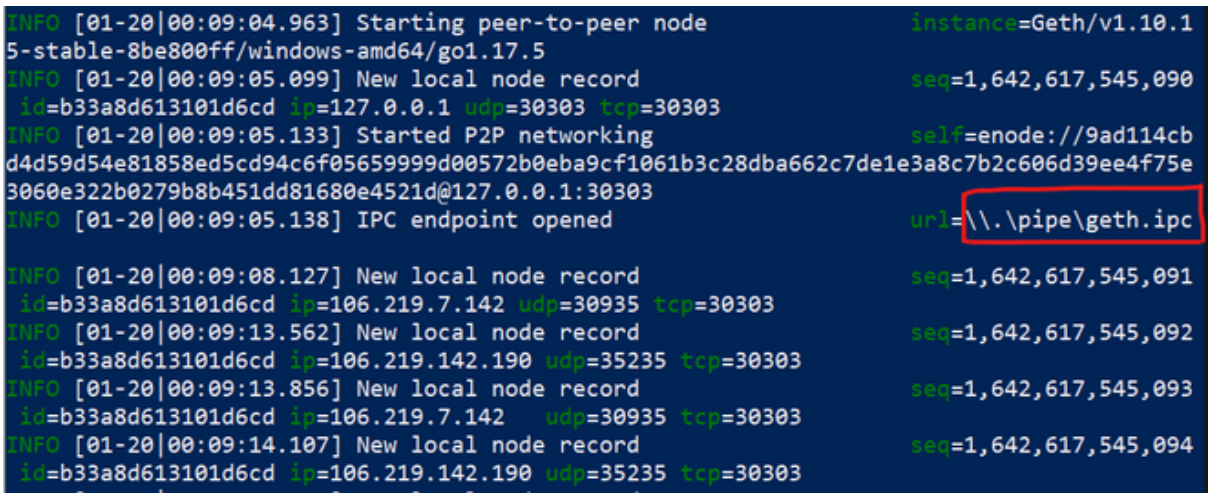
Step 5: Initialize the private network

Launch the private network in which various nodes can add new blocks for this we have to run the command-

geth –datadir \path_to_your_data_directory\MyPrivateChain –networkid 8080



The command also has the identifier 8080. It should be replaced with an arbitrary number that is not equal to the identifier of the networks already created, for example, the identifier of the main network Ethereum (“networkid = 1”). After successfully executing the command we can see like this-



Note:
The highlighted text is the address of geth.ipc file finds it in your console and copy it for use in the next step.
Every time there is a need to access the private network chain, one will need to run commands in the console that initiate a connection to the Genesis file and the private network.

Now a personal blockchain and a private Ethereum network is ready.

Step 6: Create an Externally owned account(EOA)
Externally Owned Account(EOA) has the following features-
Controlled by an External party or person.
Accessed through private Keys.
Contains Ether Balance.
Can send transactions as well as 'trigger' contract accounts.
Steps to create EOA are:
To manage the blockchain network, one need EOA. To create it, run Geth in two windows. In the second window console enter the following command-
geth attach \path_to_your_data_directory\YOUR_FOLDER\geth.ipc
or
geth attach \\.\pipe\geth.ipc

This will connect the second window to the terminal of the first window. The terminal will display the following-

```
PS C:\WINDOWS\system32> geth attach \\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.15-stable-8be800ff/windows-amd64/go1.17.5
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
 datadir: E:\MyNetwork\MyPrivateChain
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

Create an account by using the command-
personal.newAccount()

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x125c7bce5af112d0e271092be64c87ce5c31696c"
>
```

After executing this command enter Passphrase and you will get your account number and save this number for future use.

save account number

To check the balance status of the account execute the following command-

ether balance

```
> eth.getBalance("0x125c7bce5af112d0e271092be64c87ce5c31696c")
0
```

It can be seen from the above screenshot that it shows zero balance. This is because when starting a private network in the genesis file, we did not specify anything in the alloc attribute.

Step 7: Mining our private chain of Ethereum

If we mine in the main chain of Ethereum it will require expensive equipment with powerful graphics processors. Usually, ASICs are used for this but in our chain high performance is not required and we can start mining by using the following command-

miner.start()

```
> miner.start()
null
```

If the balance status is checked after a couple of seconds the account is replenished with fake ether. After that, one can stop mining by using the following command-

miner.stop()

```
> eth.getBalance("0x125c7bce5af112d0e271092be64c87ce5c31696c")
2000000000000000000000
> miner.stop()
null
>
```

Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
C1 07
2103163

Experiment No: 5

Aim: Show implementation of the blockchain platform Ethereum using Geth.

Theory:

- Ethereum is a decentralized platform that enables the creation and execution of smart contracts and decentralized apps.
- Geth is one of the three original implementations of the ethereum protocol, written in a GO Language.
- Steps to build your private blockchain:

Step 1: Install Geth
Geth is a CLI with some resources to connect you to Ethereum network. You should have access to Geth Binary before moving ahead.

Step 2: Create genesis.json
Every block chain starts with a genesis (first) block. The genesis block is configured using genesis.json file for a private network.

Step 3: Initiate the private blockchain
We have the configuration ready for the genesis block. We need to run first geth command to initialize the private blockchain.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

geth --datadir ./node1 init genesis.json

Step 4: Add the first node to the private blockchain.

Step 5: Add more nodes

Step 6: Connect node 2 with node 1 as peer.

step 7: Mining blocks and creating transactions.

- Mining is the process of validating transactions and adding them to transactions.

8/10/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Om Shete
C107
2103163

Experiment No: 6

Aim: Shows implementation of blockchain platform Ganache.

Theory:

- Ganache is a personal blockchain for Ethereum development that you can use to deploy smart contracts, develop applications and run tests.

- **Steps for Implementation:**

Step 1: Download Ganache from truffle suite: Ganache is a part of the truffle suite, a popular platform.

Step 2: Install and launch Ganache: After downloading, select "Quickstart Ethereum" from Ganache GUI.

Step 3: Explore accounts and addresses. Ganache automatically generates 10 accounts each loaded with 100 ETH for testing purpose.

Step 4: Write a simple smart contract using remix IDE, write simple smart contract for adding two numbers.

```
pragma solidity ^0.8.2;

contract Addition{
    function add(uint a, uint b) public
    pure returns (uint){
    return a+b;
    }
}
```

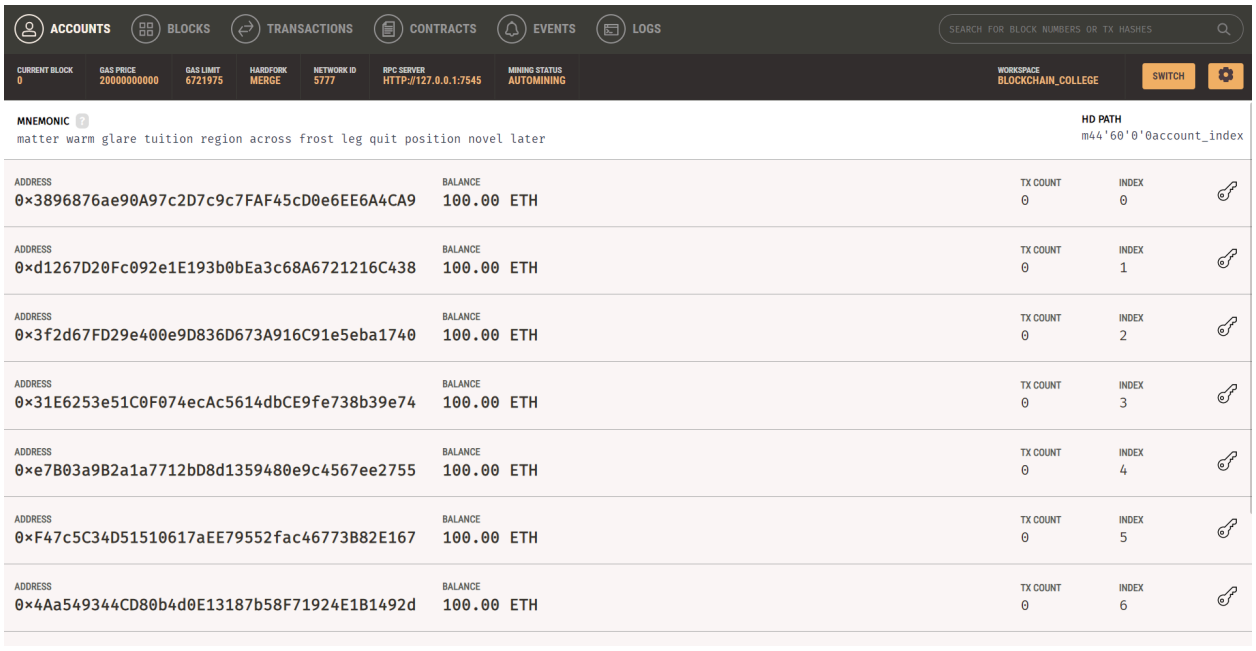
Step 5: Connect Remix IDE with Ganache
Link Remix IDE with local blockchain running on Ganache to deploy your smart contract.

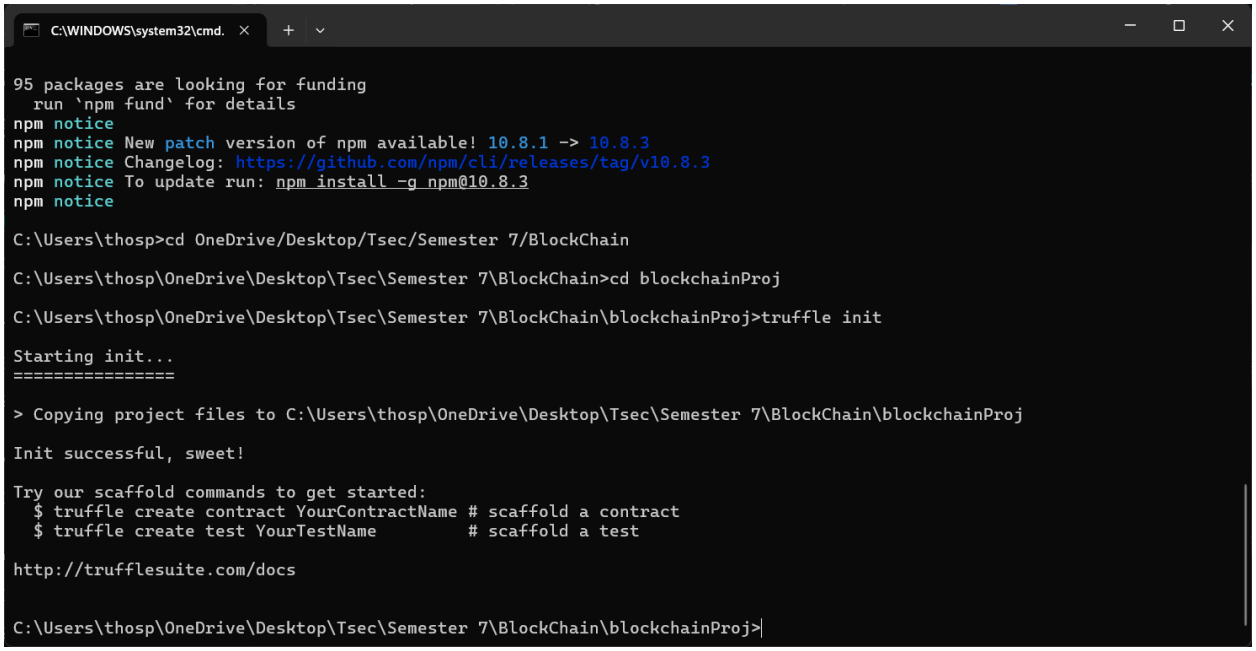
Step 6: Deploy the contract.
Using Remix IDE, deploy the smart contract to the local blockchain.

Step 7: Observe ether transactions
Monitor the effects and contract deployment on account balances.

1/10/24

Output:


ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 0
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MERGE
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE BLOCKCHAIN_COLLEGE
SWITCH
MNEMONIC
matter warm glare tuition region across frost leg quit position novel later
HD PATH
m44'60'0'0account_index
ADDRESS 0×3896876ae90A97c2D7c9c7FAF45cD0e6EE6A4CA9 BALANCE 100.00 ETH TX COUNT 0 INDEX 0
ADDRESS 0×d1267D20Fc092e1E193b0bEa3c68A6721216C438 BALANCE 100.00 ETH TX COUNT 0 INDEX 1
ADDRESS 0×3f2d67FD29e400e9D836D673A916C91e5eba1740 BALANCE 100.00 ETH TX COUNT 0 INDEX 2
ADDRESS 0×31E6253e51C0F074ecAc5614dbCE9fe738b39e74 BALANCE 100.00 ETH TX COUNT 0 INDEX 3
ADDRESS 0×e7B03a9B2a1a7712bD8d1359480e9c4567ee2755 BALANCE 100.00 ETH TX COUNT 0 INDEX 4
ADDRESS 0×F47c5C34D51510617aEE79552fac46773B82E167 BALANCE 100.00 ETH TX COUNT 0 INDEX 5
ADDRESS 0×4Aa549344CD80b4d0E13187b58F71924E1B1492d BALANCE 100.00 ETH TX COUNT 0 INDEX 6


C:\WINDOWS\system32\cmd.
95 packages are looking for funding
run `npm fund` for details
npm notice
npm notice New patch version of npm available! 10.8.1 -> 10.8.3
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.3
npm notice To update run: npm install -g npm@10.8.3
npm notice
C:\Users\thosp>cd OneDrive/Desktop/Tsec/Semester 7/BlockChain
C:\Users\thosp\OneDrive\Desktop\Tsec\Semester 7\BlockChain>cd blockchainProj
C:\Users\thosp\OneDrive\Desktop\Tsec\Semester 7\BlockChain\blockchainProj>truffle init
Starting init...
> Copying project files to C:\Users\thosp\OneDrive\Desktop\Tsec\Semester 7\BlockChain\blockchainProj
Init successful, sweet!
Try our scaffold commands to get started:
$ truffle create contract YourContractName # scaffold a contract
$ truffle create test YourTestName # scaffold a test
http://trufflesuite.com/docs
C:\Users\thosp\OneDrive\Desktop\Tsec\Semester 7\BlockChain\blockchainProj>

Welcome myContract.sol

contracts > myContract.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract MyContract {
5      string public greeting;
6
7      constructor(string memory _greeting) {
8          greeting = _greeting;
9      }
10
11     function setGreeting(string memory _greeting) public {
12         greeting = _greeting;
13     }
14 }
15
```

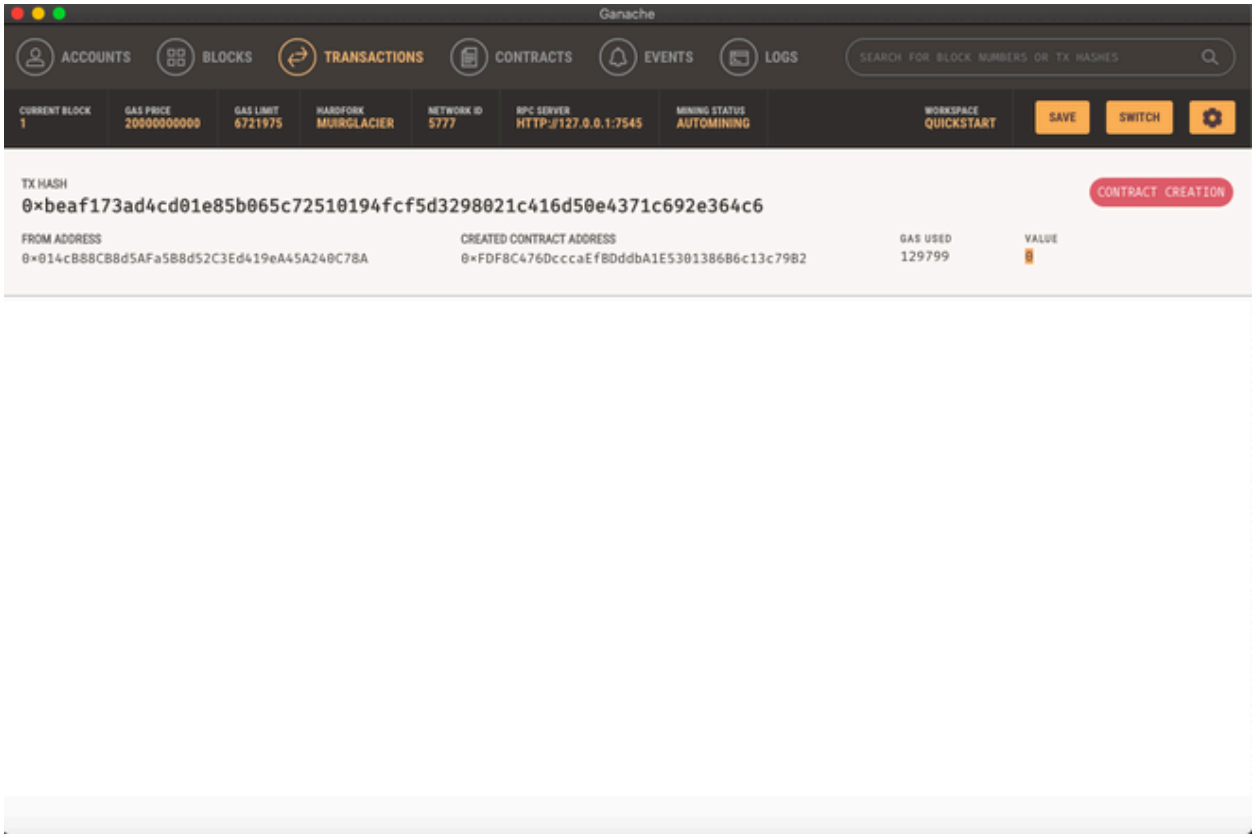
```
Compiling your contracts...
===========================
✓ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
> Compiling .\contracts\myContract.sol
> Artifacts written to C:\Users\thosp\OneDrive\Desktop\Tsec\Semester 7\BlockChain\blockchainProj\build\contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
```

```
module.exports = {

  networks: {

    development: {
     host: "127.0.0.1",
     port: 7545,
     network_id: "*",
    },
  },

  mocha: {
  },

  compilers: {
    solc: {
      version: "0.8.21",

    }
  },
};
```

myContract.sol × | 2_deploy_contracts.js | truffle-config.js

contracts > myContract.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract MyContract {
    string public greeting;

    constructor(string memory _greeting) {
        greeting = _greeting;
    }

    function setGreeting(string memory _greeting) public {
        greeting = _greeting;
    }
}

```


Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK
1
GAS PRICE
20000000000
GAS LIMIT
6721975
HARDFORK
MUIRGLACIER
NETWORK ID
5777
RPC SERVER
HTTP://127.0.0.1:7545
MINING STATUS
AUTOMINING
WORKSPACE
QUICKSTART
SAVE
SWITCH
TX HASH
0×beaf173ad4cd01e85b065c72510194fcf5d3298021c416d50e4371c692e364c6
CONTRACT CREATION
FROM ADDRESS
0×014cB88CB8d5AFa5B8d52C3Ed419eA45A240C78A
CREATED CONTRACT ADDRESS
0×FDF8C476DcccaEfBDddbA1E5301386B6c13c79B2
GAS USED
129799
VALUE
0

Case Study on Hyperledger

Hyperledger is an open-source project under the Linux Foundation where people can come and work on the platform to develop blockchain-related use cases.

Hyperledger provides the platform to create personalized blockchain services according to the need of business work. Unlike other platforms for developing blockchain-based software, Hyperledger has the advantage of creating a secured and personalized blockchain network.

- It is created to support the development of blockchain-based distributed ledgers.
- It includes a variety of enterprise-ready permissioned blockchain platforms.
- It is a global collaboration for developing high-performance and reliable blockchain and distributed ledger-based technology frameworks.

Hyperledger in the field of Pharmaceuticals:

Problem - The pharmaceutical industry is facing significant challenges related to the safety, authenticity, and traceability of drugs as they move from manufacturers to consumers. These issues directly affect patient safety, healthcare costs, and regulatory compliance

1. **Counterfeit Drugs**: A major concern is the infiltration of fake drugs into the supply chain, risking patient safety and damaging company reputations.
2. **Lack of Transparency**: The fragmented supply chain, with multiple stakeholders using siloed systems, makes it hard to track drug authenticity, storage conditions, and movement.
3. **Regulatory Compliance**: Meeting regulatory demands for drug traceability and quality control is challenging due to reliance on inefficient, manual systems.
4. **Slow Recalls**: Drug recalls are delayed because it's difficult to trace faulty drugs quickly and accurately.
5. **High Costs**: Inefficiencies from manual record-keeping and disjointed systems increase operational costs and the price of drugs.

The industry needs a transparent, secure system for tracking drugs to ensure safety, improve compliance, and cut costs.

Solution - Hyperledger, particularly Hyperledger Fabric, offers a blockchain-based solution that addresses the key challenges of the pharmaceutical supply chain, providing enhanced security, transparency, and traceability.

1. Blockchain for End-to-End Drug Traceability:

- Single Source of Truth: By implementing Hyperledger Fabric, pharmaceutical companies can create a unified, immutable ledger that tracks every step in the drug's lifecycle—from manufacturing to distribution to final consumption.
- Unique Drug IDs: Every drug batch or unit is assigned a unique identifier (often a QR code or RFID tag) that gets registered on the blockchain. As the drug moves through the supply chain, each stakeholder (manufacturer, distributor, wholesaler, pharmacy) scans and updates the blockchain with relevant data like batch number, storage conditions, and timestamps.
- Real-Time Updates: Since all stakeholders access the same blockchain ledger, updates to a drug's journey are reflected in real-time, allowing for immediate visibility into its status. This ensures that drugs can be traced from origin to destination with full transparency, helping to mitigate issues like counterfeiting and mismanagement.

2. Preventing Counterfeit Drugs:

- Immutable Ledger: Hyperledger ensures that once data is written to the blockchain, it cannot be altered or tampered with. This creates a tamper-proof record of every transaction and transfer of drugs. Any attempt to introduce counterfeit drugs can be easily detected because the chain of custody is transparent and unchangeable.
- Authentication at Each Step: As drugs pass through the supply chain, the blockchain authenticates each party involved. By the time a drug reaches the pharmacy or the end consumer, its journey can be verified to ensure that it has not been tampered with or replaced with a counterfeit product.

3. Regulatory Compliance and Auditability:

- Automated Compliance Checks: Hyperledger Fabric enables the use of smart contracts, which are self-executing contracts programmed to perform tasks like verifying compliance. For instance, when a drug is shipped, a smart contract could automatically verify that it meets the required temperature conditions and regulatory standards before it's transferred to the next party.
- Transparent Audit Trail: Hyperledger provides a detailed, verifiable audit trail for regulators. Instead of relying on manual records, companies can grant regulatory bodies access to the blockchain ledger. This enables authorities to see a drug's entire history, ensuring compliance with safety standards, proper storage, and handling conditions.
- Simplified Reporting: The immutable nature of blockchain allows pharmaceutical companies to easily generate reports for regulatory bodies, reducing the risk of

non-compliance, errors, or incomplete data, which can lead to fines or approval delays.

4. Efficient Drug Recalls:

- Fast Identification of Faulty Batches: With a complete and transparent record of each drug's journey, pharmaceutical companies can quickly identify and isolate faulty or contaminated batches. For example, if a batch of drugs is discovered to be defective, the company can instantly trace the affected batch to its exact location within the supply chain.
- Minimized Recall Scope: Hyperledger allows companies to narrow down recalls to specific batches or locations, reducing the scope and financial impact of recalls. This ensures that only the affected drugs are pulled from circulation, avoiding unnecessary recalls of unaffected products.

5. Enhanced Drug Security and Data Privacy:

- Permissioned Blockchain: Hyperledger Fabric operates on a permissioned network, meaning that only authorized parties can access specific data. This ensures that sensitive information, like patient data or proprietary manufacturing processes, remains confidential.
- Encryption and Privacy: Data on the blockchain can be encrypted, with different levels of access granted to different participants. For example, a distributor might be able to see shipment details, but not the precise formulation of the drug, which is only accessible to the manufacturer.

6. Supply Chain Efficiency and Cost Reduction:

- Automation of Manual Processes: By leveraging blockchain, manual tasks such as data entry, paperwork, and document verification are automated, reducing human error and streamlining processes. Smart contracts also automate key processes, such as payment releases upon verification of delivery or the satisfaction of storage conditions.
- Reduced Operational Costs: By eliminating the need for third-party intermediaries, reducing paperwork, and cutting down the time spent on verification processes, companies can reduce their overall operational costs. This also lowers the price of drugs for consumers.
- Inventory Management: Hyperledger helps with real-time inventory tracking and management. Companies can optimize their stock levels by using blockchain to monitor drug demand and supply trends, minimizing wastage or shortages.

7. Global Supply Chain Coordination:

- Interoperability Across Borders: Hyperledger Fabric's flexibility allows it to integrate with different systems and regulatory environments. As pharmaceutical companies distribute drugs globally, Hyperledger can accommodate the various regulatory requirements of different countries, ensuring compliance no matter where the drugs are being shipped.
- Cross-Border Transparency: A global supply chain involves multiple parties across different regions, which often leads to fragmentation. Hyperledger creates a global ledger that records the journey of drugs across borders in a unified, tamper-proof way, ensuring consistency in tracking regardless of geographic location.

8. Pharmacovigilance and Patient Safety:

- Tracking Drug Efficacy and Side Effects: Once drugs reach patients, Hyperledger can be extended to track patient outcomes, side effects, or issues through the use of IoT devices or healthcare systems that feed data back into the blockchain. This creates a transparent system for monitoring the long-term efficacy and safety of pharmaceuticals.
- Enhanced Patient Trust: Patients can scan a drug's packaging to verify its authenticity, trace its journey, and view safety information. This fosters greater trust in the pharmaceutical company and ensures patients are confident in the medications they're taking.

9. Collaboration with IoT and AI Technologies:

- IoT Integration: IoT devices, such as temperature sensors or GPS trackers, can feed real-time data into the Hyperledger blockchain, providing constant monitoring of drug storage and shipping conditions. Any deviations, such as a temperature breach, are automatically logged on the blockchain, triggering alerts and corrective actions.
- AI for Predictive Analysis: AI algorithms can analyze blockchain data to predict supply chain disruptions, forecast drug demand, or identify patterns that could indicate the risk of counterfeit drugs entering the supply chain.

Implementation - The implementation of Hyperledger in the pharmaceutical supply chain begins with identifying key stakeholders such as manufacturers, distributors, logistics providers, pharmacies, and regulators, each assigned specific roles and permissions within the blockchain network. This ensures that only authorized participants have access to relevant data. The next step involves designing the blockchain architecture, where channels are created to allow private transactions between subsets of participants, such as manufacturers and distributors. Smart contracts are developed to automate processes like verifying drug authenticity, ensuring proper storage conditions,

and triggering payments upon delivery. It's also essential to determine which data will be stored on-chain (e.g., drug batch numbers, timestamps) and what should remain off-chain, such as proprietary formulations.

Once the blockchain framework is in place, IoT devices like temperature sensors, GPS trackers, and RFID scanners are integrated to automatically capture real-time data about drug shipments. This data feeds directly into the blockchain, ensuring that critical information, such as the location of shipments or any temperature deviations, is recorded instantly and immutably. Existing ERP systems are also connected to the blockchain, enabling a seamless flow of information between legacy systems and the new network. Before full-scale deployment, a pilot test is conducted to track a specific drug or region within the supply chain. This phase ensures that all systems function as expected, smart contracts execute correctly, and data flows smoothly across the blockchain. After a successful pilot, the system is scaled to cover the entire supply chain, providing real-time traceability, security, and efficiency for pharmaceutical products.

Benefits and Outcomes - The successful implementation of Hyperledger in the pharmaceutical supply chain yields numerous benefits, fundamentally transforming how drugs are tracked, verified, and distributed. One of the primary advantages is enhanced drug traceability, providing end-to-end visibility of drug movement and ensuring every step—from manufacturing to patient delivery—is securely recorded. This transparency allows companies to quickly verify the authenticity of drugs and address any irregularities, such as counterfeit products or deviations in storage conditions.

Additionally, Hyperledger significantly reduces the risk of counterfeit drugs infiltrating the supply chain. Its immutable ledger ensures that once a drug's information is added, it cannot be altered or falsified, increasing trust among consumers, pharmacies, and healthcare providers regarding the authenticity and safety of the medications they handle. The implementation also improves regulatory compliance by providing a transparent, auditable trail of all transactions. Smart contracts automate compliance checks, ensuring drugs meet necessary standards, while regulators can access real-time data, which reduces delays in audits.

In the event of a faulty or contaminated drug batch, Hyperledger enables rapid and accurate tracing of affected products, minimizing the scope of recalls and thereby reducing costs and preventing unnecessary wastage. Moreover, the system enhances operational efficiency by automating processes such as compliance checks and data entry, which lowers the administrative burden on pharmaceutical companies. This reduction in the need for intermediaries and paperwork leads to lower operational costs, which can, in turn, lower drug prices for consumers.

Patient safety is also significantly improved, as the ability to verify drug authenticity and track its entire journey ensures patients are less likely to receive counterfeit or expired medications. Real-time monitoring of storage conditions via IoT integration further safeguards drug integrity. Lastly, Hyperledger fosters greater trust and collaboration across the supply chain, as each stakeholder has access to the same source of truth. This level of transparency encourages better collaboration between manufacturers, distributors, and regulators, improving the overall integrity of the pharmaceutical industry. In summary, implementing Hyperledger enhances security, reduces risks, streamlines operations, and promotes greater transparency and accountability, ultimately leading to safer, more reliable drug distribution.

1/10/24

Case Study on BlockChain Platform(Corda)

Corda is an open-source blockchain platform developed by R3, specifically designed for businesses, particularly within the financial services sector. Unlike traditional blockchains that are open and public, Corda allows participants to share information on a need-to-know basis. This unique approach addresses critical challenges faced by the financial industry, such as data silos, security vulnerabilities, regulatory compliance, and slow transaction times. By enabling direct transactions among financial institutions, Corda streamlines processes, reduces reliance on intermediaries, and fosters collaboration in a highly regulated environment.

The financial services industry grapples with several significant challenges that impede efficiency and transparency. One major issue is the existence of data silos, where different institutions operate independently, leading to inefficiencies and a lack of trust among participants. Additionally, the industry faces persistent threats of fraud and security breaches, which undermine confidence in financial systems. Regulatory compliance poses another challenge, as institutions must navigate complex regulations while ensuring data integrity and transparency. Finally, traditional systems often result in slow transaction times, affecting liquidity and customer satisfaction.

Overview

Corda addresses these challenges with its innovative architecture designed specifically for businesses. The platform allows financial institutions to transact directly with each other, eliminating unnecessary intermediaries and enhancing operational efficiency. Key features of Corda include privacy by design, ensuring that only relevant parties see transaction data, which enhances confidentiality. Smart contracts automate various processes, such as verifying the conditions of transactions and enforcing agreements, leading to faster and more accurate transactions. Moreover, Corda's interoperability allows it to connect with existing systems and other blockchains, enabling broader collaboration across the financial ecosystem.

The implementation of Corda within the financial services sector is a comprehensive process that involves various steps and considerations to ensure a seamless transition from traditional systems to a blockchain-based model. This section outlines the key stages of Corda's implementation in a hypothetical scenario, emphasizing its application in a global trade finance platform that involves multiple stakeholders.

1. Stakeholder Identification

The first step in implementing Corda is identifying the key stakeholders involved in the process. In the context of a global trade finance platform, stakeholders may include banks, exporters, importers, logistics companies, and regulatory authorities. Each participant has distinct roles and requirements, necessitating a collaborative approach to design a system that meets the needs of all parties. Engaging stakeholders early in the process is crucial to gather insights and ensure that the platform addresses their specific challenges and objectives.

2. System Design and Architecture

Once stakeholders are identified, the next step is to design the system architecture. This involves defining how participants will interact with the Corda network and what functionalities will be included. The architecture must accommodate the privacy requirements of each stakeholder, enabling them to access only the information relevant to their transactions. During this phase, the design of smart contracts is also undertaken, outlining the specific conditions and actions that will govern transactions between parties. The architecture should also allow for scalability, anticipating future growth and the potential addition of new stakeholders.

3. Setting Up Corda Nodes

Each participant in the trade finance platform must set up a **Corda node** to connect to the network. A Corda node serves as the participant's private ledger and handles transaction processing. Setting up these nodes requires technical expertise and infrastructure to ensure they can operate securely and efficiently. Participants may need to collaborate with their IT teams or external vendors to ensure their systems are compatible with Corda's requirements. Proper configuration of nodes is essential for enabling point-to-point communication, which is a hallmark of Corda's transaction model.

4. Developing Smart Contracts

The development of smart contracts is a critical component of the implementation process. Smart contracts are programmed to automatically execute specific actions based on predefined conditions agreed upon by the stakeholders. For the trade finance platform, this could involve automating processes such as verifying the authenticity of trade documents, authorizing payments when conditions are met, and ensuring compliance with regulations. The development phase should include thorough testing of smart contracts to identify and address any potential issues before deployment. This ensures that contracts function as intended and mitigate risks associated with automated processes.

5. Data Migration and Integration

As stakeholders transition to the Corda platform, data migration from existing legacy systems becomes a crucial task. This involves transferring relevant historical data to the new system while ensuring data integrity and compliance with regulations. Integration with existing systems is also essential, as stakeholders may want to retain certain legacy processes while benefiting from Corda's features. This phase may require custom APIs or connectors to facilitate seamless communication between the Corda network and existing applications.

6. User Training and Change Management

Successful implementation of Corda requires effective user training and change management strategies. Stakeholders must be educated on how to use the new platform, understand the functionalities of their Corda nodes, and interact with smart contracts. Training sessions should be tailored to the specific roles of participants to ensure that each user feels confident in navigating the system. Additionally, organizations should foster a culture of adaptability to embrace the changes associated with implementing a blockchain solution. This might involve providing ongoing support and resources to assist users during the transition.

7. Pilot Testing

Before a full-scale rollout, conducting a pilot test is essential to validate the system's functionality and performance. The pilot should involve a limited number of transactions among selected stakeholders to identify any potential issues and gather feedback on the platform's usability. This testing phase allows for adjustments and refinements based on real-world interactions, ensuring that the system is robust and effective in meeting stakeholder needs.

8. Full Deployment

Following successful pilot testing and any necessary adjustments, the platform can be fully deployed. At this stage, all stakeholders can begin using Corda for their trade finance transactions. Continuous monitoring is crucial during the initial rollout to quickly address any issues that arise and ensure that the system operates smoothly. Stakeholders should remain engaged in providing feedback to facilitate ongoing improvements to the platform.

9. Regulatory Compliance and Monitoring

Given the heavily regulated nature of the financial services sector, implementing mechanisms for regulatory compliance and monitoring is essential. Corda allows

stakeholders to maintain an auditable trail of transactions, which can be accessed by regulators when needed. Establishing protocols for compliance monitoring ensures that the platform meets all necessary regulatory requirements and can adapt to any changes in the legal landscape.

Smart contracts are utilized to automate processes such as document verification, payment authorization, and compliance checks. This automation significantly reduces transaction times and minimizes the potential for errors. Instead of exchanging physical documents, participants share verified data directly on the Corda network, ensuring that only authorized users have access to sensitive information. Additionally, Corda provides tools for real-time monitoring, allowing regulators to access relevant transaction data while preserving client confidentiality.

Benefits and Outcomes

The implementation of Corda in the financial services sector results in numerous advantages. One of the most significant benefits is increased efficiency; transactions that previously took days can now be completed in hours or even minutes, greatly improving liquidity for all parties involved. Enhanced security is another key advantage, as the design allows information to be shared only with relevant parties, thus reducing the risk of data breaches and fraud. Furthermore, Corda facilitates easier regulatory compliance by providing an auditable trail of transactions that regulators can access without compromising client confidentiality.

The reduction of manual processes and the need for intermediaries leads to significant cost savings for banks and financial institutions. Moreover, Corda's transparent and secure framework fosters greater trust among stakeholders, enhancing collaboration in the financial ecosystem. As participants engage more freely, the overall integrity and efficiency of the financial system improve.

Challenges and Considerations

Despite its many advantages, Corda does face certain challenges that must be addressed for successful implementation. One significant barrier is adoption resistance; traditional financial institutions may be slow to embrace new technologies due to concerns over existing legacy systems and regulatory compliance. Additionally, interoperability issues can arise when integrating Corda with various legacy systems and standards, complicating the transition process. Establishing a clear governance model for the network is also essential, as it ensures that all participants can collaborate effectively and maintain trust in the system.

Conclusion

In summary, Corda represents a powerful blockchain solution specifically tailored for the financial services industry. By addressing key challenges such as data silos, security risks, and regulatory compliance, Corda enhances efficiency and trust in financial transactions. Its focus on privacy and smart contracts positions it as a valuable tool for transforming financial operations, paving the way for a more collaborative and secure financial ecosystem. As the industry continues to evolve, Corda stands out as a promising option for institutions seeking to leverage blockchain technology to enhance their operations and better serve their clients.

11/10/24

Report on Mini Project

Subject: Blockchain

AY: 2024-25

E-VOTING SYSTEM USING BLOCKCHAIN AND PYTHON

Parth Puranik: 2103142

Mohib Abbas Sayed: 2103158

Hamza Sayyed: 2103159

Om Shete: 2103163

Guided By

(Tanuja Sarode)

PROBLEM STATEMENT

Electronic voting systems are increasingly becoming an alternative to traditional paper ballots. However, they are vulnerable to cyber-attacks that can compromise the integrity of votes, reveal voter identities, or tamper with ballots. Current methods to anonymize voters, like encryption, are not foolproof, as intelligence agencies or hackers can intercept communications across the internet, potentially undermining the confidentiality of the votes.

The goal of this project is to design and implement a blockchain-based e-Voting system that ensures complete voter anonymity, vote integrity, and protection against tampering. This system will allow voters to securely cast their votes while ensuring that no external influence can manipulate or identify the vote.

PROPOSED SYSTEM

- Authentication: Only people already registered to vote can cast a vote. Our system will not support a registration process. Registration usually requires verification of certain information and documents to comply with current laws, which could not be done online in a secure manner. Therefore, the system should be able to verify voters' identities against a previously verified database, and then let them vote only once.
- Anonymity: The e-Voting system should not allow any links between voters' identities and ballots. The voter has to remain anonymous during and after the election.
- Accuracy: Votes must be accurate; every vote should be counted, and can't be changed, duplicated or removed.
- Verifiability: The system should be verifiable to make sure all votes are counted correctly. Beside the main requirement, our solution supports mobility, flexibility, and efficiency. However, we will limit this paper's discussion to the four main requirements.

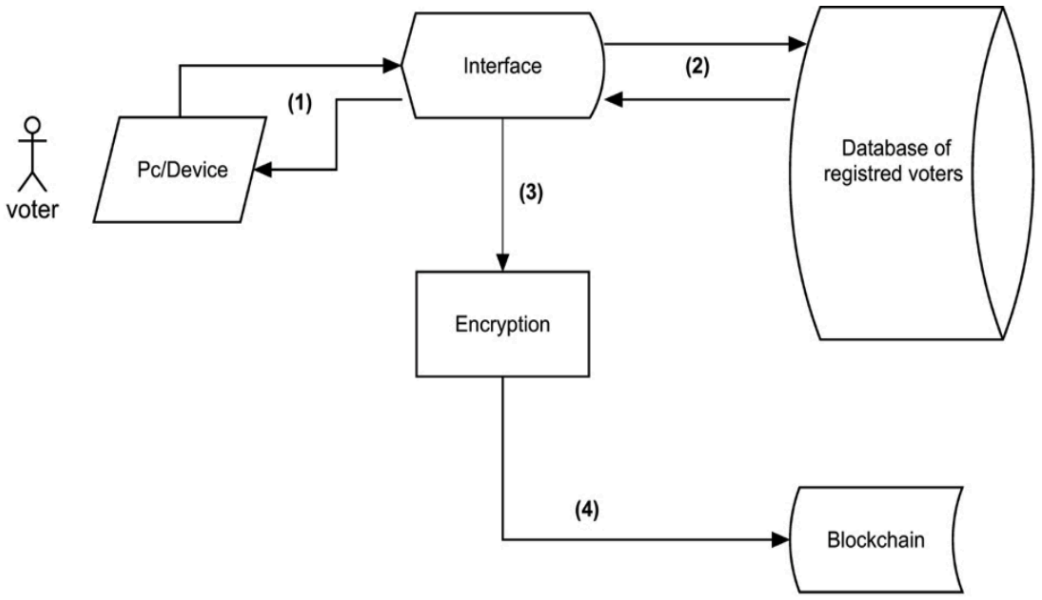


Figure 5. Simplified Representation of the e-Voting System

To ensure that the system is secure, the block will contain the previous voter's information. If any of the blocks were compromised, then it would be easy to find out since all blocks are connected to each other. The Blockchain is decentralized and cannot be corrupted; no single point of failure exists. The Blockchain is where the actual voting takes place. The user's vote gets sent to one of the nodes on the system, and the node then adds the vote to the Blockchain. The voting system will have a node in each district to ensure the system is decentralized.

BLOCKCHAIN CONCEPTS USED:

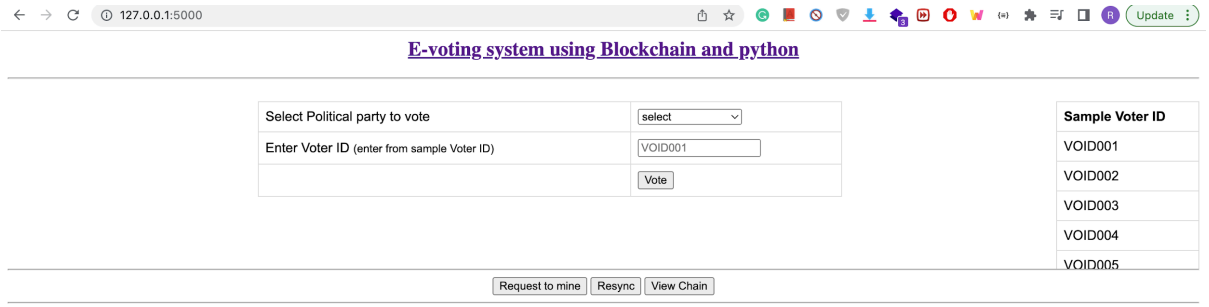
The technologies used in our system are:

1. Decentralization: By leveraging blockchain's decentralized nature, the voting process is distributed across multiple nodes. This prevents any single entity from having control over the system, reducing the risk of manipulation.
2. Immutability: Once a vote is cast and recorded on the blockchain, it cannot be altered or deleted, ensuring that every vote remains intact and unchanged throughout the voting process

3. Smart Contracts: Smart contracts are used to automate vote validation and counting processes, ensuring that rules are enforced without human intervention. This provides transparency and trust in the vote-counting mechanism.
4. Encryption and Cryptographic Hashing: Cryptographic algorithms ensure that votes remain private and secure during transmission and storage. Public and private keys are used to anonymize voters and prevent unauthorized access to vote data.
5. Consensus Mechanism: A proof-of-stake or proof-of-work consensus mechanism can be used to validate transactions (votes) on the blockchain, ensuring that only legitimate votes are counted while preventing fraudulent activity.

RESULTS:

Posting vote



Requesting the node to mining

Resyncing with the chain for updated data

Result Detail

Voter ID	Political Party Name	Voted Time
VOID007	Republican Party	2022-04-10 11:58
VOID006	Democratic Party	2022-04-10 11:58
VOID005	Democratic Party	2022-04-10 11:58
VOID004	Democratic Party	2022-04-10 11:58
VOID003	Democratic Party	2022-04-10 11:57
VOID002	Republican Party	2022-04-10 11:57
VOID001	Democratic Party	2022-04-10 11:56

Result Summary

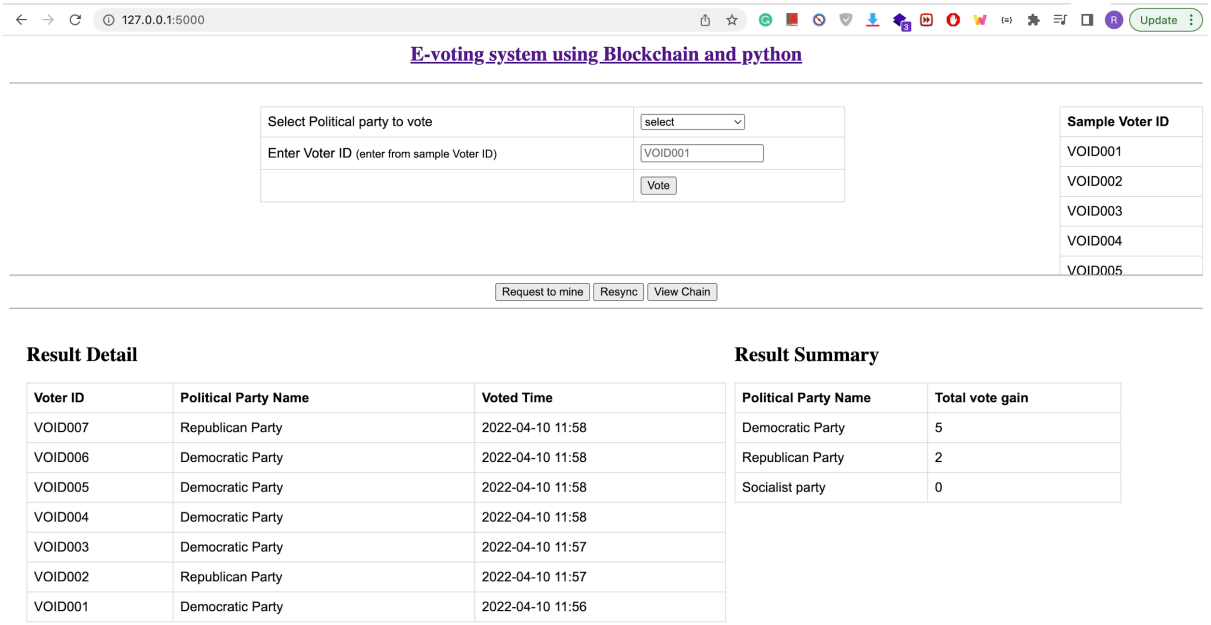
Political Party Name	Total vote gain
Democratic Party	5
Republican Party	2
Socialist party	0

Chain of the transaction

127.0.0.1:8000/chain


```
// 20220410115920
// http://127.0.0.1:8000/chain

{
  "length": 4,
  "chain": [
    {
      "index": 0,
      "transactions": [

      ],
      "timestamp": 0,
      "previous_hash": "0",
      "nonce": 0,
      "hash": "6dbf23122cb5046cc5c0c1b245c75f8e43c59ca8ffeac292715e5078e631d0c9"
    },
    {
      "index": 1,
      "transactions": [
        {
          "voter_id": "VOID001",
          "party": "Democratic Party",
          "timestamp": 1649571086.02753
        }
      ],
      "timestamp": 1649571091.222877,
      "previous_hash": "6dbf23122cb5046cc5c0c1b245c75f8e43c59ca8ffeac292715e5078e631d0c9",
      "nonce": 291,
      "hash": "0023b7c389e5c9e0cb5bef2a89f5b4c279bdfc982431eeae443d9cb2601804da"
    },
    {
      "index": 2,
      "transactions": [
        {
          "voter_id": "VOID002",
          "party": "Republican Party",
          "timestamp": 1649571134.078284
        },
        {
          "voter_id": "VOID003",
          "party": "Democratic Party",
          "timestamp": 1649571145.802302
        }
      ],
      "timestamp": 1649571147.0531268,
      "previous_hash": "0023b7c389e5c9e0cb5bef2a89f5b4c279bdfc982431eeae443d9cb2601804da",
      "nonce": 57,
      "hash": "00c418152b97383dbea68234cd37a5ad97194701e51dccb1176012bac71b0144"
    },
```

App Screenshot



Result Detail

Voter ID	Political Party Name	Voted Time
VOID007	Republican Party	2022-04-10 11:58
VOID006	Democratic Party	2022-04-10 11:58
VOID005	Democratic Party	2022-04-10 11:58
VOID004	Democratic Party	2022-04-10 11:58
VOID003	Democratic Party	2022-04-10 11:57
VOID002	Republican Party	2022-04-10 11:57
VOID001	Democratic Party	2022-04-10 11:56

Result Summary

Political Party Name	Total vote gain
Democratic Party	5
Republican Party	2
Socialist party	0

CONCLUSION

The blockchain-based secure e-Voting system developed in this project addresses key challenges associated with traditional electronic voting systems, such as vote manipulation, tampering, and lack of voter anonymity. By integrating blockchain technology, the system ensures transparency, immutability, and decentralization, creating a more secure voting environment. The use of cryptographic techniques guarantees voter privacy, while smart contracts automate and secure the vote validation process.

This project demonstrates the potential of blockchain to revolutionize the voting process by providing a secure, transparent, and trustless system, ensuring that every vote is cast and counted with integrity. While blockchain can significantly improve security, further research and real-world testing are needed to optimize scalability, performance, and accessibility for larger-scale elections. Nonetheless, the project lays a strong foundation for developing robust e-Voting solutions in the future.

8/7/24

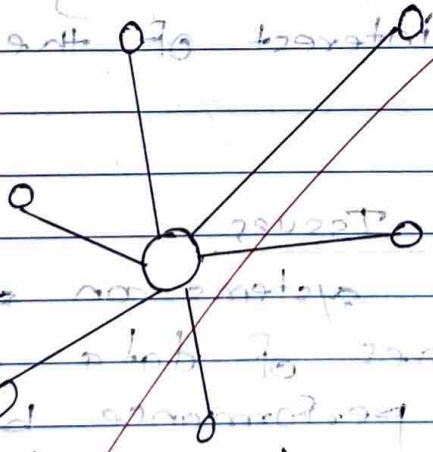
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Om Shete
C1 07
2103163

Assignment No : 1

Q.1 What are the problems associated with a centralized system?

⇒

- Centralized systems are type of computing architecture where all or most of the processing and data storage is done on a single server or group of closely connected servers.
- This central server manages all operations, resources and data acting as the hub through which all clients requests are processed.



○ – server / master

○ – slave

- The clients or nodes connected to the central server typically have minimal processing powers and rely on the server for most computational tasks.
- The problems associated with a centralized systems are :-

1) Single point of failure

⇒ Centralized system have a single point of failure, meaning if this central entity fails or

is compromised, the entire network can be affected.
- For e.g., in traditional centralized banking system, if a central bank's database is hacked or its operations disrupted, it can lead to widespread financial problems affecting millions of customers.

2) Control and Governance
⇒ Centralization often leads to a concentration of decision-making power in the hands of a few entities, which may not always act in the best interest of the community or users.

3) Scalability Issues
⇒ Centralized systems can struggle to handle large volumes of data and high traffic, leading to performance bottlenecks.
For eg., in some centralized payment processing system, during peak times such as holidays or sales events, the system may become overloaded, causing delays or even failures in processing transactions.

4) Lack of transparency
⇒ Centralized systems may lack transparency in decision-making processes, resources allocation, or even in the validation and

auditing of transactions.
- For e.g., in a centralized supply chain management system based on blockchain, if the central authority controlling the system does not provide open access to transaction data or supply chain management information or stakeholders may not have a clear view of the entire process, leading to the distinct and inefficiencies.

5) Regulatory Risks

⇒ Centralized systems are more susceptible to government regulations and interventions which can impact their operations and sustainability.

- For e.g. a centralized cryptocurrency exchange could be shut down or heavily regulated by the government, disrpting users access and the market.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q.2 Differentiate between centralized, decentralized and distributed systems.

⇒

Centralized Systems	Decentralized Systems	Distributed Systems
1) Single central server controls & manages all operations	1) Multiple nodes with independent control, no central authority	1) Multiple interconnected nodes working together as a single system.
2) Centralized control with single point of management.	2) Distributed control, each node operates independently.	2) Shared control, nodes collaborate to achieve common goals.
3) High risk; if the central server fails, the whole system fails.	3) Reduced risk; failure of one node does not impact entire system.	3) Reduced risk; designed for fault tolerance & redundancy.
4) Limited scalability, can become a bottleneck.	4) More scalable, can add nodes independently.	4) High scalable, can add nodes to distribute the load.
5) Easier to manage centrally.	5) More complex, requires managing multiple node.	5) Complex, requires coordination & management of many nodes.
6) Lower latency, as operations are managed centrally.	6) Can vary, depends on the distance between nodes.	6) Potentially higher latency due to network communication.

29/7/24 (A+)

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Date 21/03/23

Assignment 2

Q.1 Explain transactions in blockchain and UTXO and Double Spending.

⇒

Transaction :

- A blockchain's basic building block is a transaction.
- Currency is transferred from one address to another during a transaction.
- A transaction is a piece of cryptographically signed instructions issued by an externally owned account, serialized and then uploaded to a blockchain.
- On Ethereum, a transaction is a data packet that has been digitally signed using a private key and contains the instruction that, when followed, either lead to a message call or the establishment of a contract.
- Based on the output they generate, transactions may be classified into two types :

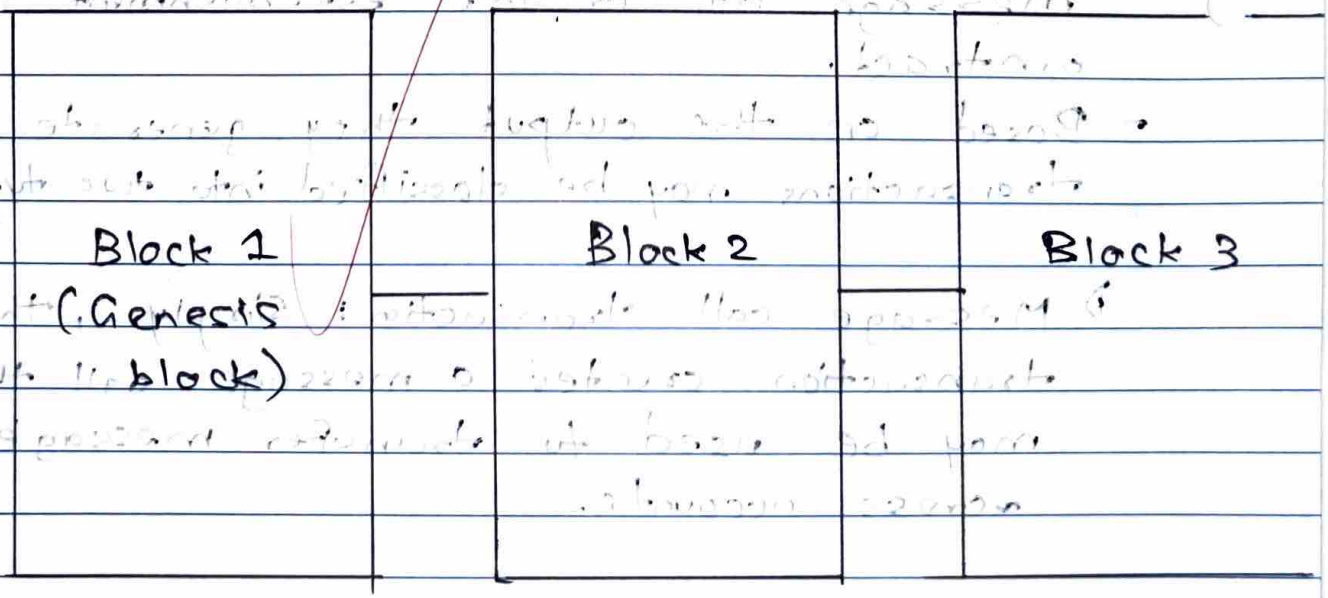
1) Message call transaction : Simply, this transaction creates a message call that may be used to transfer message across accounts.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

2) Contract creation transaction: As the name implies, a new contract is made as a result of these transactions. This indicates when this transaction is completed successfully, an account is created with the corresponding code.

- **Unspent Transaction Output (UTXO):**

- Blockchain is a digital, decentralized, distributed ledger.
- Blockchain utilizes a P2P (Peer-to-Peer) network, where participants present on the network are called nodes.
- The ledger stores data about transaction. It is a chain of blocks, where its most significant feature is that blocks are cryptographically linked together.



- In bitcoin, the transaction lives until it has been executed till the time another transaction is done out of that UTXO.
- It is the amount of digital currency someone has left remaining after executing a transaction.
- When a transaction is completed, the unspent output is deposited back into the database as input which can be used later for another transaction.
- UTXOs are created through the consumption of existing UTXOs. Every Bitcoin is composed of inputs and outputs.
- Inputs consumes an existing UTXO, while outputs create a new UTXO.

Double Spending :

- Although blockchain is secured, still it has some loopholes.
- Hackers or malicious users take advantage of these loopholes to perform their activities.
- Double spending means the expenditure of the same digital currency twice or more to avail the multiple services. It is a technical flow that allows users to duplicate money.

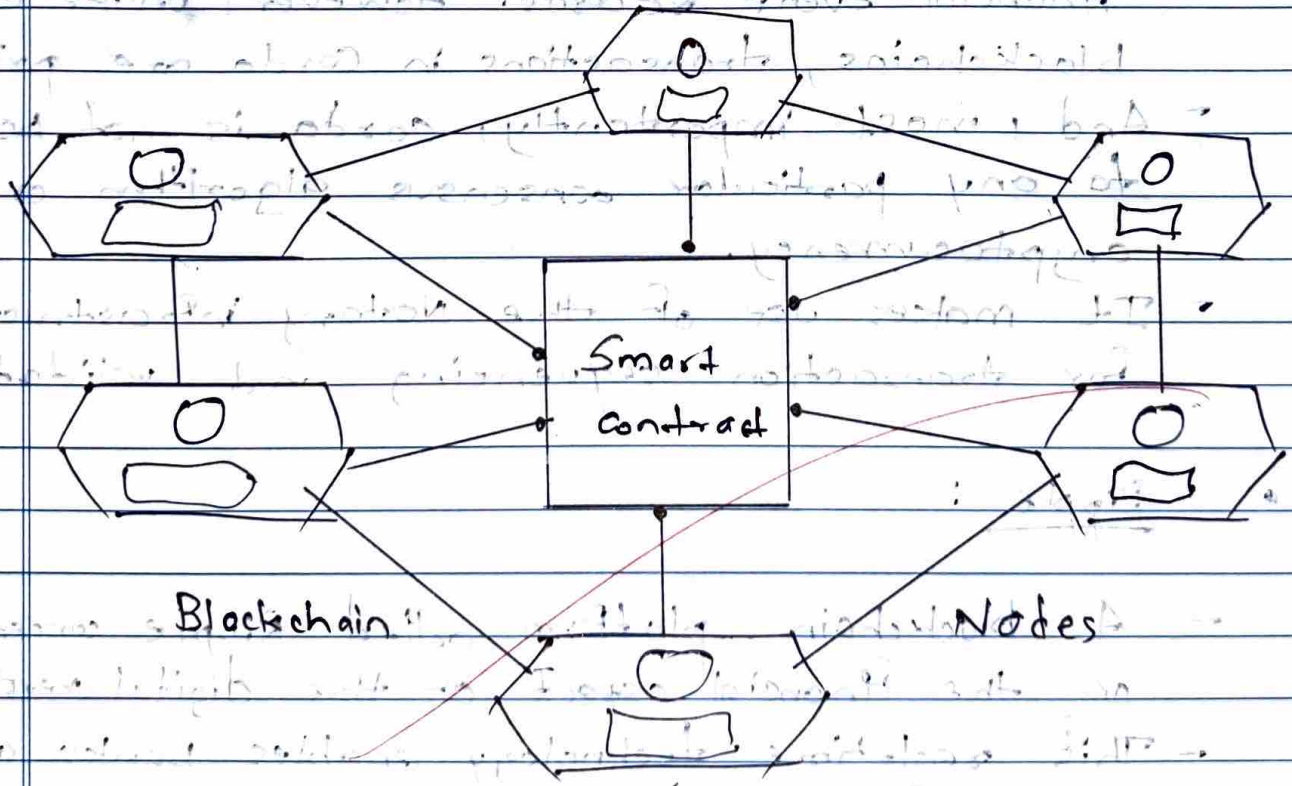
- Since digital currencies are nothing but files, a malicious users can create multiple copies of the same currency file and can use it in multiple places.
- This issue can also occur if there is an alteration in the network or copies of the currency are only used and not the original one.
- There are also double spends that allows hackers to reverse transactions so that transaction happens two times.
- By doing this, the user loses money two times one for the fake block created by the hacker and for the original block as well.
- The hacker gets incentives as well for the fake blocks that have been mined and confirmed.

Q.2 Explain in detail CORDA, Ripple and Quorum

⇒

- **Corda :**

- It is a blockchain that is open-source and uses smart contracts to let companies deal privately and directly.
- Corda is a distributed ledger technology (DLT) that R3 created specifically for enterprise use, with privacy as the guiding principle.



- The Corda protocol is built on a strong identity model, where every node's identity must be proven to have been properly on boarded by using X.509 certificate

- Corda is a custom-designed for financial industry.
- It reduces record-keeping expenses and offers development services like Corda App Consulting, User Interfaces, Regulated tokens, and others.
- The distributed apps created with Corda are known as Cordapps.
- Signing identities are only available on Corda Nodes.
- Corda enables the creation of immutable financial event records. However, unlike other blockchains, transactions in Corda are private.
- And, most importantly, Corda is not bound to any particular consensus algorithm or cryptocurrency.
- It makes use of the Notary infrastructure for transaction sequencing and validation.

- **Ripple :**

- A blockchain platform called Ripple concentrates on the financial aspect of the digital revolution.
- This real-time technology enables banks and other financial institutions to transmit payment transactions across the globe in real time.
- Ripple is an open-source person-to-person payment network that allows anyone in the world to send money to anyone

elese for free.
- For banks and payment processors, Ripple Labs aims to make it simple and affordable to send money abroad.
- There is a big problem when your want to send money from one country to another country.
- Sometimes it takes days and it is usually expensive, and in some countries you even have to do illegal things to make it happen.
- Ripple's primary goal is to eliminate the need for older system such as Western Union or Swift.
- Ripple is a distributed payment protocol.
- Ripple is emerging as the next promising virtual currency.

- **Quorum :**

- A distributed ledger protocol called Quorum is based on Ethereum.
- As Quorum only modifies Ethereum's core slightly, it is intended to grow and change alongside Ethereum.
- It is a JP Morgan's Enterprise focused blockchain which is open source.
- Quorum is a free and open source blockchain protocol designed specifically for use in a private blockchain network, in which multiple

members each own a portion of the network.

- It is a software of the very well-known public Ethereum blockchain.
- Quorum is Ethereum blockchain aimed specifically at the finance sector.
- It is created solely to provide privacy for private transactions between nodes.
- The main feature of Quorum is privacy. Transactions and smart contracts on the blockchain can be private.
- Quorum features multiple voting based consensus mechanisms that do away with the current proof-of-work consensus algorithm used by public Ethereum blockchain.
- Quorum can be run with two different privacy options, Tessera and Constellation, which are used by Quorum to encrypt and data among nodes.
- Transactions on the Quorum network don't cost Ether. Voting is the primary method of transaction verification, a mechanism for agreement.

Q.3 Explain 0, 1 and 6 confirmation transaction.

⇒

In the context of blockchain technology, the terms "0 confirmation", "1 confirmation" and "6 confirmation" refers to the number of blocks that have been mined on top of a given transaction's block in a blockchain.

0 transaction

- A 0 transaction is one that has been broadcasted to the network but has not yet been included in a block.
- This means the transaction is still in the mempool where transaction's are stored while waiting to be included in block by miners.
- Status: Pending, not yet confirmed by the blockchain.
- Risk: The transaction can still be canceled or replaced by the sender, making it susceptible to a "double-spend-attack" where the sender tries to use the same funds in another transaction.
- Use: Merchants or individuals usually avoid accepting 0 confirmation transaction for large payments because they are not secure.

1 Confirmation Transaction :

- A 1 confirmation transaction has been included in a block, meaning that one block has been mined containing the transaction, and it is officially part of the blockchain.
- Status : The transaction is confirmed and recorded in the blockchain, but only one block has been mined after the transaction.
- Risk : Although it is more secure than 0 confirmation transaction, a 1 confirmation transaction could theoretically still be reversed if a malicious miner reorganizes the blockchain by creating a longer competing chain.
- Use : Some businesses or individuals may accept transactions with 1 confirmation for smaller or lower-risk transactions because the chances of a reversal are low, but not negligible.

6 confirmation Transaction :

- A 6 confirmation transaction has been confirmed by the network and included in a block, and five additional blocks have been mined on top of it.

- Status: Very secure. The transaction is now deeply embedded in the blockchain and the probability of it being reversed is extremely low.
- Risks: The risk of reversal at this point is practically negligible because altering a transaction this deep in the chain would require an attacker to recognize the six blocks.
- Use: Most exchanges, merchants and businesses consider 6 confirmations as the gold standards for security, especially for high-value transactions. Once a transaction has six confirmations, it is generally considered irreversible.

Q.4: What is smart contract? How crowd funding platforms can be managed using smart contracts?

⇒ Smart Contract:

- The smart contract is the software program implementing a set of rules or conditions and operates on the top of a blockchain.
- These set of rules are used by different parties to that smart contract to transfer the digital assets between them.
- Based on set of rules implemented in smart contracts, different parties to the smart contracts agree to interact with each other.
- This agreement automatically gets executed after fulfilling the predefined rules.
- The smart contracts code execute, validate and ensure the transaction or agreement.
- Hence, they are fully automated and it is possible to automate the records of real-life agreements related to purchase and sale of assets.
- A smart contract implements security coding of the blockchain and include details and permissions that require a precise sequence of events to happen to trigger agreement of the terms stated in the smart contract.

- **How Crowdfunding Platforms can be managed using smart contracts :**

- Crowdfunding platforms can be greatly enhanced using smart contracts to automate the process of collecting and distributing funds while ensuring transparency and security.

1) **Decentralized Fund Collection**

⇒ Smart contracts can be programmed to accept funds from investors or backers of a crowdfunding project.
- Contributors send funds directly to the smart contract, eliminating the need for intermediaries like traditional crowdfunding platforms.
- A goal amount can be set in the smart contract

2) **Conditional based Fund Release**

⇒ IF the fund raising goal is met by a specific deadline, the smart contract automatically releases the funds to the project creator.
- IF the goal is not met, the smart contract automatically refunds the contributors, ensuring transparency and reducing the risk of fraud.

30/09/24