Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./Miss Armav Malvia of Computer Engg Department, Semester VII with Roll No. 2103109 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

18/10/24

Teacher In-Charge

Head of the Department

Date 18/10/2024

Principal

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1.	Write a program to implement Merkle Tree showing the cryptographic hash function in blockchain.		16/7/24	
2.	Create Smart Contract using Solidity and Remix IDE.		23/7/24	
3.	Write aprogram in Solidity programming to create transactions using the Remix IDE.		30/7/24	
4.	Write a program in solidity to implement transactions by embedding of wallet.		6/8/24	
5.	Show implementation of blockchain platform Ethereum using Geth.		13/8/24	
6.	Show implementation of blockchain platform Ganache.		20/8/24	18/10/24
7.	Conduct a Case Study on Hyperledger.		27/8/24	
8.	Conduct a Case study on Blockchain Platform.		3/9/24	
9.	Mini Project		24/9/24	
10.	Assignment :- 1		13/7/24	
11.	Assignment :- 2		10/9/24	

Arnav Malvia
C-32
2103109
Blockchain
Date:- 16/7/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

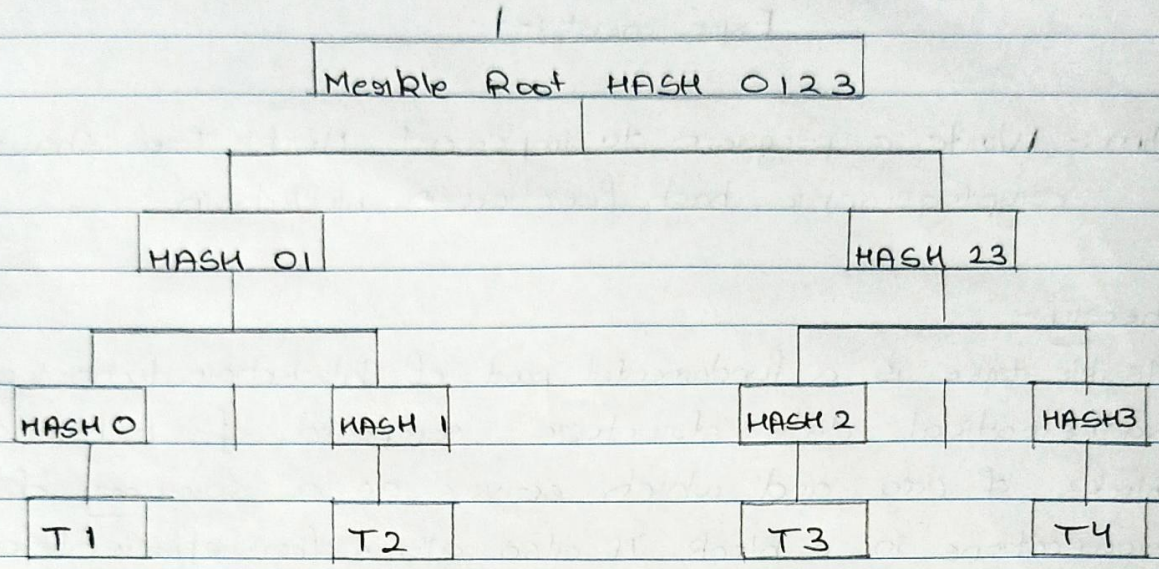
Experiment :- 1

Aim :- Write a program to implement Merkle Tree showing the cryptographic hash function in blockchain.

Theory :-
Merkle tree is a fundamental part of blockchain technology. It is a mathematical data structure composed of hashes of different blocks of data and which serves as a summary of all the transactions in a block. It also allows for efficient and secure verification of contrast in a large body of data. It also helps to verify the consistency and contrast of the data. Both bitcoin and ethereum use Merkle Tree structures. Merkle tree is also known as Hash Tree.

The concept of Merkle tree is named after Ralph Merkle, who patented this idea in 1979. Fundamentally, it is a data structure in which every leaf node labelled with the hash of a data block and the non-leaf node labelled with cryptographic hash of the labels of its child nodes. The leaf nodes are the lowest node in the tree.

Working of Merkle Tree-
A merkle tree stores all the transactions in a block by providing a digital fingerprint of the entire set of transactions. It allows the user to verify whether a transaction can be included in a block or not.



BASIC STRUCTURE OF MERKLE TREE

Merkle trees are created by repeatedly calculating hashing pairs of nodes until there is only one hash left. This hash is called the merkle root or root mesh. The merkle tree is constructed in bottom up approach. Every leaf node is a hash of transactional data and the non-leaf node is a hash of its previous hashes. Merkle trees are in a binary tree so it requires an even number of leaf nodes. If there is an odd number of transactions, the list hash will be duplicated once to create an even number of leaf nodes. The example drawn is the most common and simple form of merkle tree ie. Binary Merkle Tree.

Advantages of Merkle Tree-

1) It provides means to maintain the integrity and validity of data.
2) It helps in securing the memory or disk space as proofs, computationally easy and fast.
3) Their proofs and management require tiny amount of information to be transmitted across networks.

A 30/7/24

Conclusion:- Hence we have studied and implemented Merkle Trees showing the cryptographic hash function in Blockchain.

Exp 1

INPUT :

```
import hashlib
class MerkleTree:
def __init__(self, data_blocks):
self.leaves = [(data, self.hash_data(data)) for data in data_blocks]
self.intermediate_hashes = [] # To store intermediate hashes
self.root = self.build_tree([leaf[1] for leaf in self.leaves])
def hash_data(self, data):
# Hashes data using SHA-256 and returns the hexadecimal digest
return hashlib.sha256(data.encode(&#39;utf-8&#39;)).hexdigest()
def build_tree(self, leaves):
if len(leaves) == 1:
return leaves[0]
# If odd number of leaves, duplicate the last leaf
if len(leaves) % 2 != 0:
leaves.append(leaves[-1])
# Compute the parent level
parent_level = []
for i in range(0, len(leaves), 2):
combined_hash = leaves[i] + leaves[i + 1]
parent_hash = self.hash_data(combined_hash)
parent_level.append(parent_hash)
self.intermediate_hashes.append((leaves[i], leaves[i + 1], parent_hash)) # Store intermediate
hashes
# Recursively build the tree until we get the root hash
return self.build_tree(parent_level)
def get_root(self):
return self.root
def get_leaves(self):
return self.leaves
def get_intermediate_hashes(self):
return self.intermediate_hashes
if __name__ == &quot;__main__&quot;:
data_blocks = []

print(&quot;Enter data blocks for the Merkle Tree (type &#39;done&#39; to finish):&quot;)
while True:
data = input(&quot;Enter data block: &quot;)
if data.lower() == &#39;done&#39;:
break
data_blocks.append(data)
if data_blocks:
```

```
merkle_tree = MerkleTree(data_blocks)
print(&quot;Hashes of each block:&quot;)
for data, hash_value in merkle_tree.get_leaves():
print(f&quot;Data: {data}, Hash: {hash_value}&quot;)
print(&quot;\nIntermediate hashes:&quot;)
for leaf1, leaf2, parent_hash in merkle_tree.get_intermediate_hashes():
print(f&quot;Combined: {leaf1} + {leaf2} -&gt; Hash: {parent_hash}&quot;)
print(f&quot;\nMerkle Root: {merkle_tree.get_root()}&quot;)
else:
print(&quot;No data blocks entered.&quot;)
```

OUTPUT :

```
C:\Users\anish\PycharmProjects\pythonProject1\venv\Scripts\python.exe
C:\Users\anish\PycharmProjects\pythonProject1\bc_merkle.py
Enter data blocks for the Merkle Tree (type &#39;done&#39; to finish):
Enter data block: arnav
Enter data block: maithili
Enter data block: jash
Enter data block: chetana
Enter data block: done
Hashes of each block:
Data: arnav, Hash:
40fc3451e882282df7cc37fdc6d3fd9b4a1d3673c1a05c0350ed4ce31a3ce49b
Data: maithili, Hash: 4cdef747af0872ef28f794224daf75944d0e31c9cf1f1885d2d18b45f41f51bc
Data: jash, Hash: bd878fa2293de72e9c871463e1c2ef2964acb0ea282540c9d2f52fe32c6d18d1
Data: chetana, Hash:
d8d6881de3a6bfab0bfd8496c5ae90e7f6c3a8e66f4b36ff9f9137037991f3fc

Intermediate hashes:

Combined: 40fc3451e882282df7cc37fdc6d3fd9b4a1d3673c1a05c0350ed4ce31a3ce49b +
4cdef747af0872ef28f794224daf75944d0e31c9cf1f1885d2d18b45f41f51bc -&gt; Hash:
1654fa61d0d7a11add50bbc3d15be9f4b9d098e181b28138a8afeb24ed32314f
Combined: bd878fa2293de72e9c871463e1c2ef2964acb0ea282540c9d2f52fe32c6d18d1 +
d8d6881de3a6bfab0bfd8496c5ae90e7f6c3a8e66f4b36ff9f9137037991f3fc -&gt; Hash:
13f33e5718920f76cf1e4acdbbbe10fad666d13c7c8fa5989c3ff82f60cd3283
Combined: 1654fa61d0d7a11add50bbc3d15be9f4b9d098e181b28138a8afeb24ed32314f +
13f33e5718920f76cf1e4acdbbbe10fad666d13c7c8fa5989c3ff82f60cd3283 -&gt; Hash:
ca851364fdeb77f51580f608252284ecebfebd8dd9cc8c845ee9dea8839ce367

Merkle Root: ca851364fdeb77f51580f608252284ecebfebd8dd9cc8c845ee9dea8839ce367

Process finished with exit code 0
```

Arnav Malvia
C-32
2103109
Blockchain
Date :- 23/7/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment :- 2

Aim :- Creating 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 capitalisation.

Ethereum -
Ethereum is ~~de~~ a decentralised open source platform based on the Blockchain domain, used to run smart contracts i.e. applications that execute the program exactly as it was programmed without the possibility of any fraud, interference from a third party censorship or downtime. It serves as a platform for many 2,60,000 different cryptocurrencies.

Ethereum Virtual Machine -
Ethereum Virtual Machine abbreviated as EVM is a runtime environment for executing smart contracts in ethereum. It focuses wildely on providing security and execution of untrusted code using an international network of public nodes. EVM is specified to prevent Denial of Service Attack and cautions that the progress does not have any access to each other's state.

Smart Contract -
Smart Contract are high level program codes that are compiled to EVM byte code and deployed to the

Ethereum Blockchain for further execution. It allows us to provide perform credible transactions without any interference of the third party these transactions are traceable and irreversible.

Basic syntax of Solidity-

```
pragma solidity ^0.8.0;
contract HelloWorld{
    unit a;
    function set(uint x) public{
        a = x;
    }
    function get() public view returns (uint){
        return a;
    }
}
```

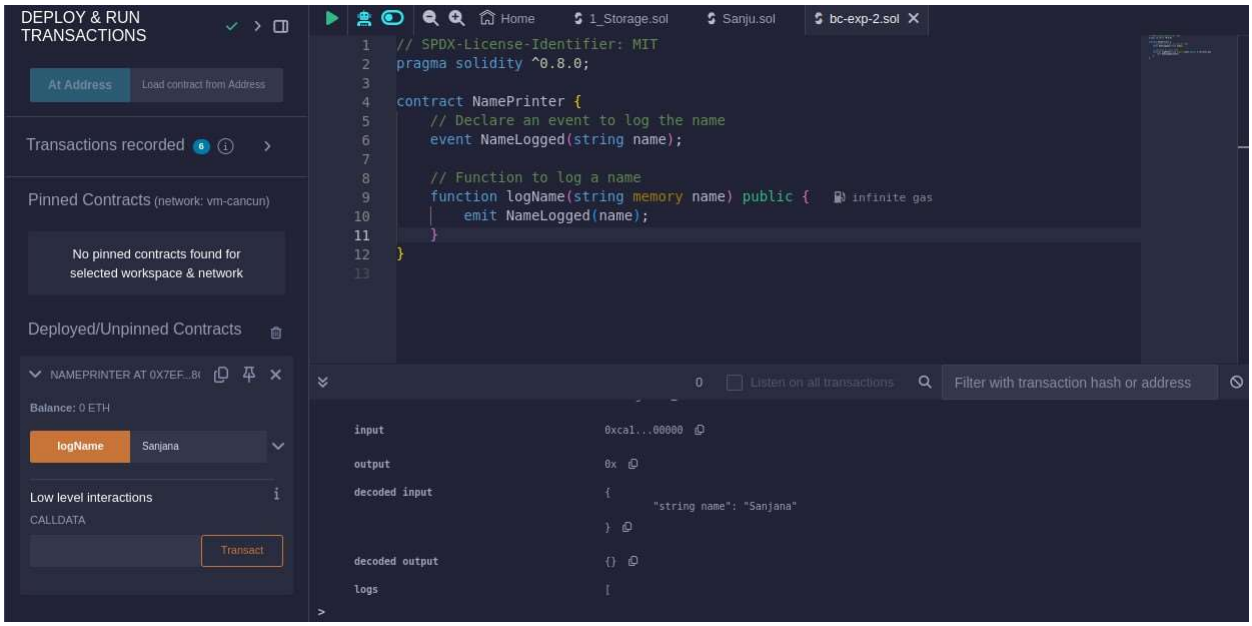
Conclusion:- Thus we have studied and implemented smart contract using Solidity and Remix IDE.

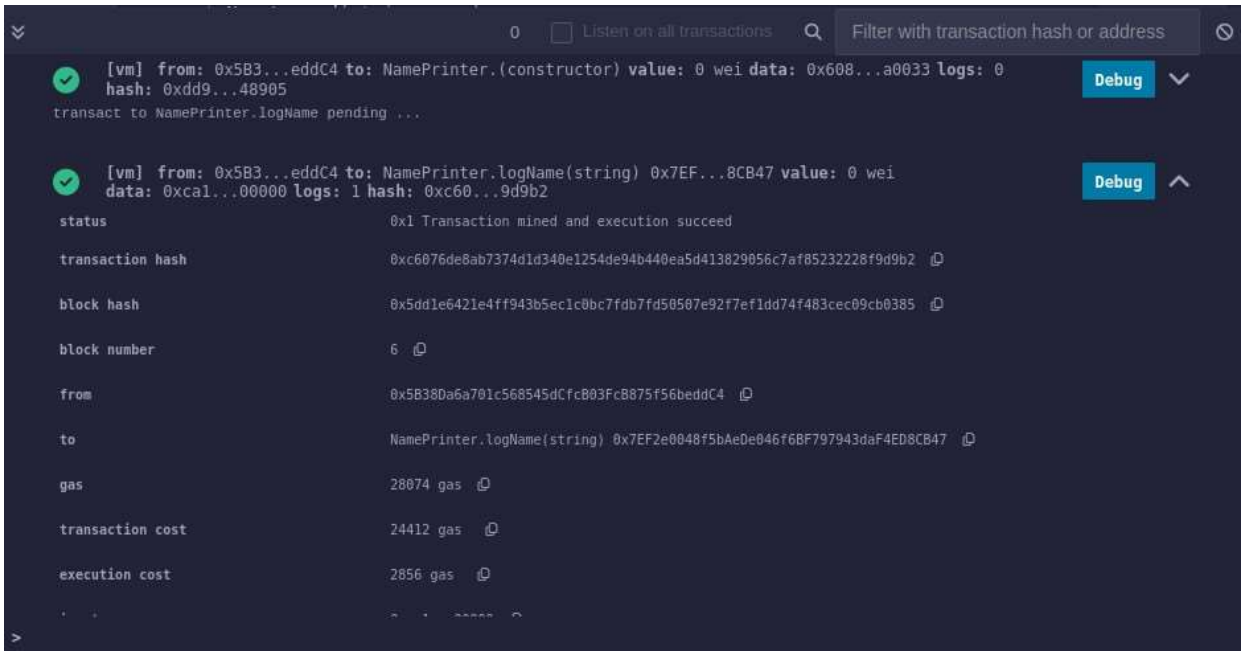
30/7/24

A

Exp 2

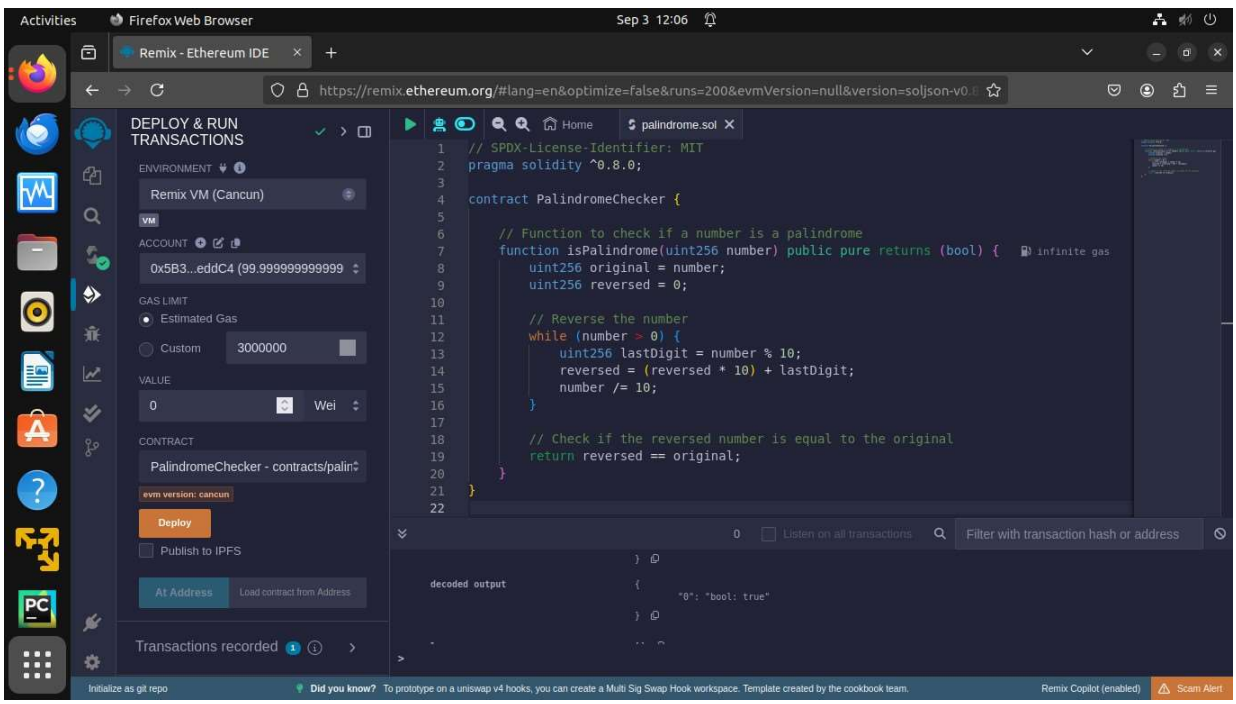
Print name

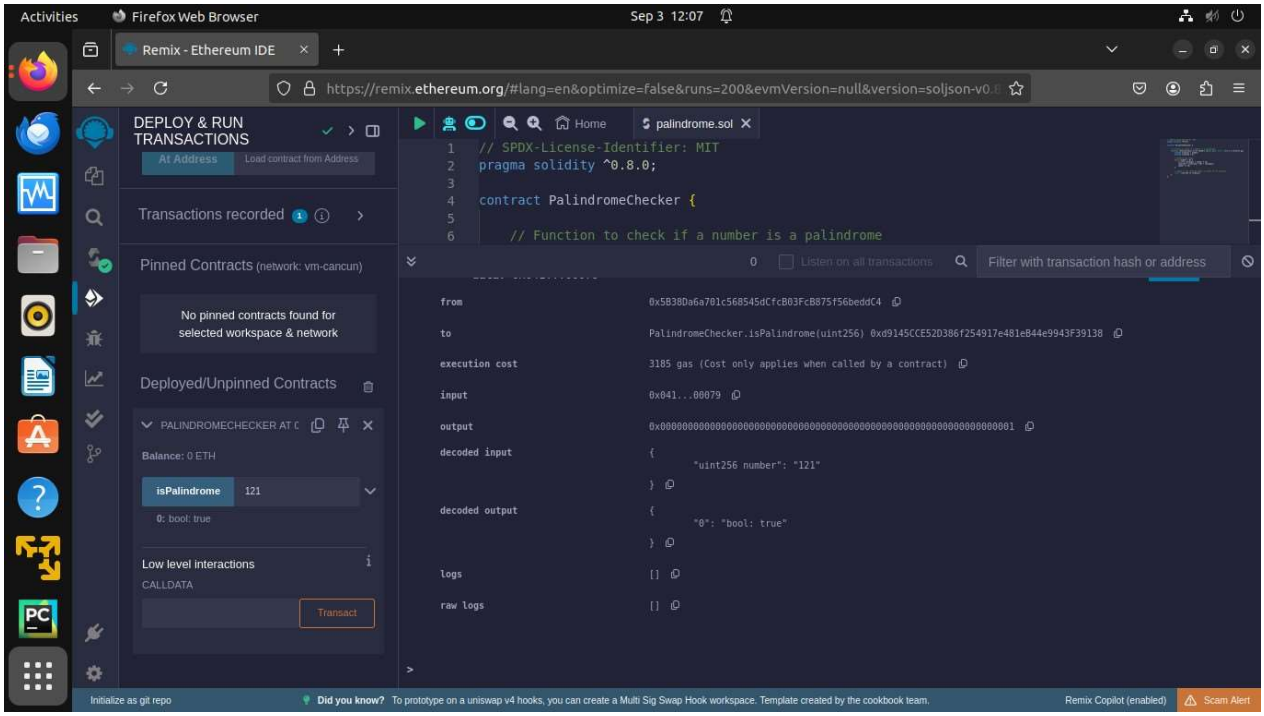


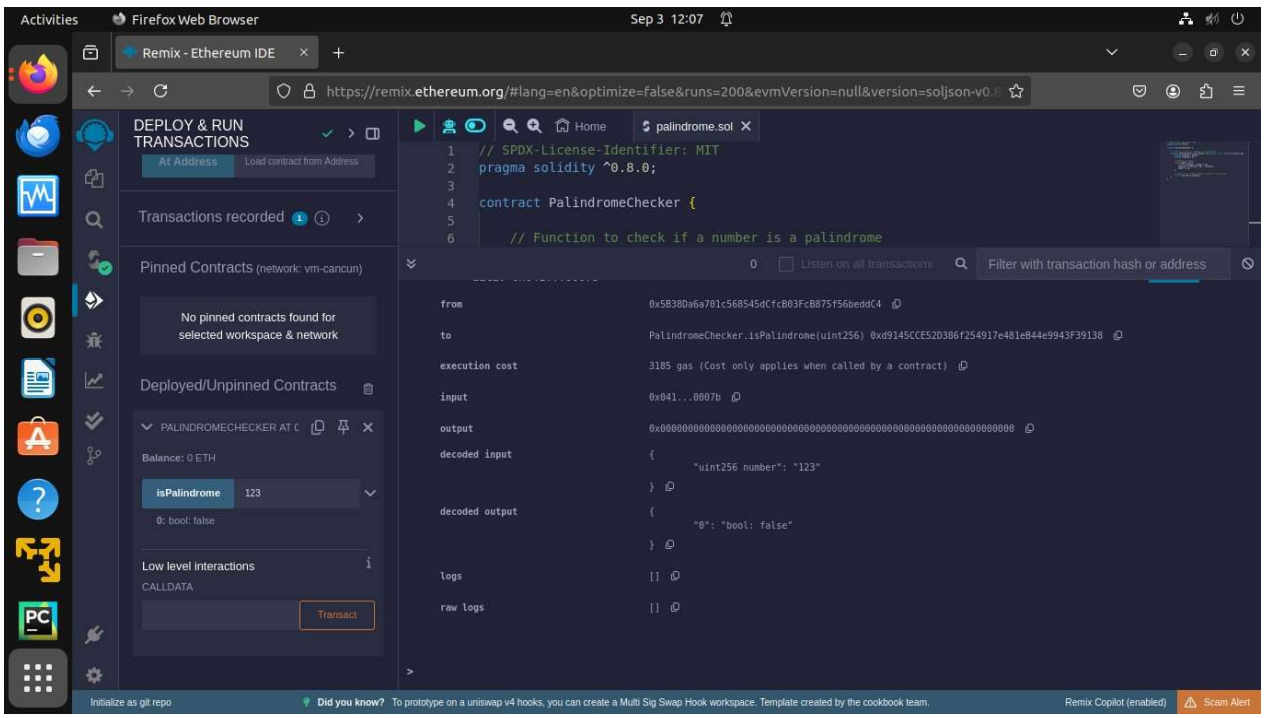


0
Listen on all transactions
Filter with transaction hash or address
output
0x
decoded input
{
"string name": "Sanjana"
}
decoded output
{}
logs
[
{
"from": "0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47",
"topic": "0xc2c76c3f1eadf31309c26094101c81ec76f1996ce2d6aa8e83f5db5c18ce00cd",
"event": "NameLogged",
"args": {
"0": "Sanjana",
"name": "Sanjana"
}
}
]
raw logs
[
{
"logIndex": "0x1",
"blockNumber": "0x6",
"blockHash": "0x5dd1e6421e4ff943b5ec1c0bc7fdb7fd50507e92f7ef1dd74f483cec09cb0385",
"transactionHash": "0xc6076de8ab7374d1d340e1254de94b440ea5d413829056c7af85232228f9d9b2",
"transactionIndex": "0x0",
"address": "0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47",
>

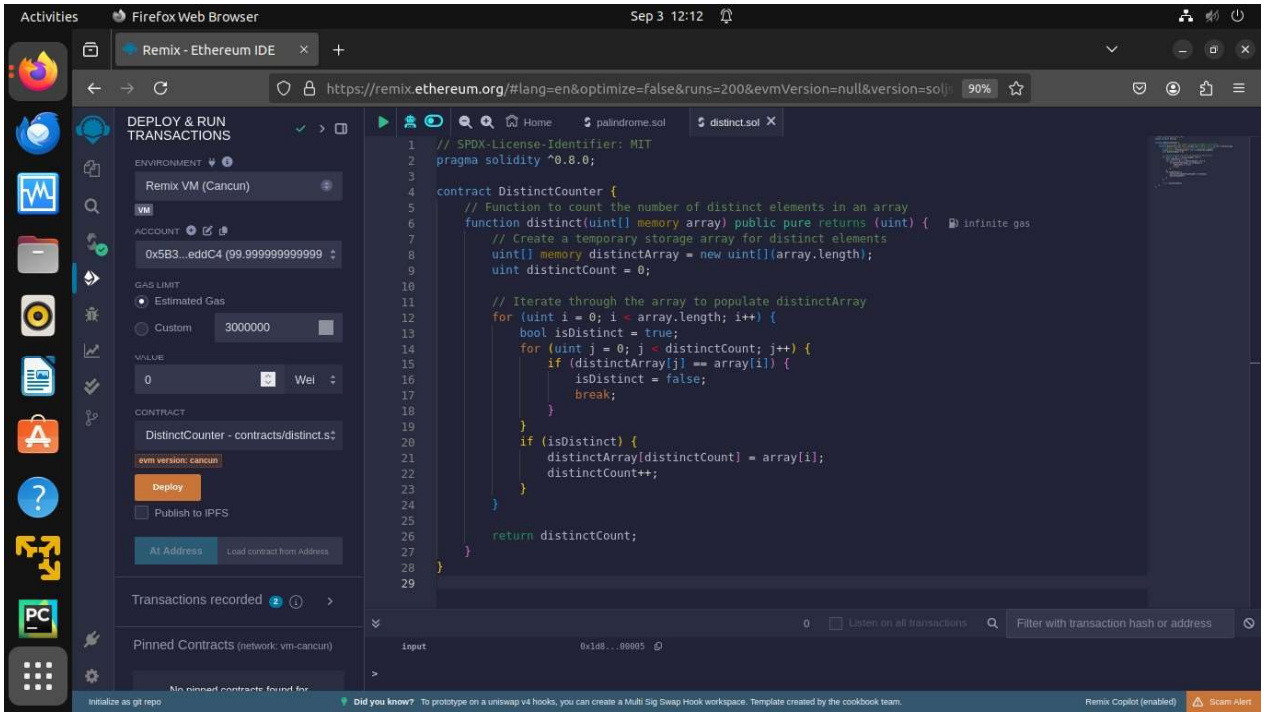
Palindrome

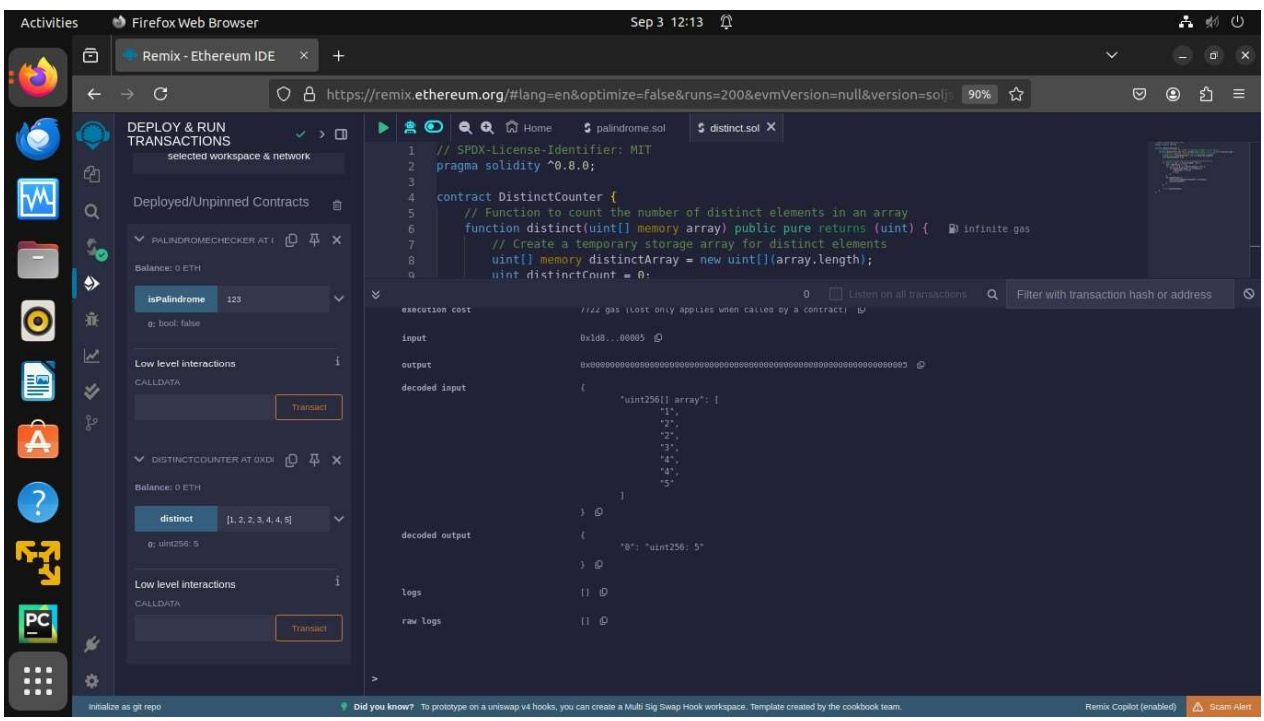

Activities
Firefox Web Browser
Sep 3 12:06
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8
DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (99.999999999999
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
PalindromeChecker - contracts/palin
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Home
palindrome.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract PalindromeChecker {
// Function to check if a number is a palindrome
function isPalindrome(uint256 number) public pure returns (bool) {
infinite gas
uint256 original = number;
uint256 reversed = 0;
// Reverse the number
while (number > 0) {
uint256 lastDigit = number % 10;
reversed = (reversed * 10) + lastDigit;
number /= 10;
}
// Check if the reversed number is equal to the original
return reversed == original;
}
}
0
Listen on all transactions
Filter with transaction hash or address
decoded output
{
"0": "bool: true"
}
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert


Activities
Firefox Web Browser
Sep 3 12:07
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.
DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded
Pinned Contracts (network: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
PALINDROMECHECKER AT
Balance: 0 ETH
isPalindrome
121
0: bool: true
Low level interactions
CALLDATA
Transact
Home
palindrome.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract PalindromeChecker {
// Function to check if a number is a palindrome
Listen on all transactions
Filter with transaction hash or address
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
PalindromeChecker.isPalindrome(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost
3185 gas (Cost only applies when called by a contract)
input
0x041...00079
output
0x0000000000000000000000000000000000000000000000000000000000000001
decoded input
"uint256 number": "121"
decoded output
"0": "bool: true"
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert

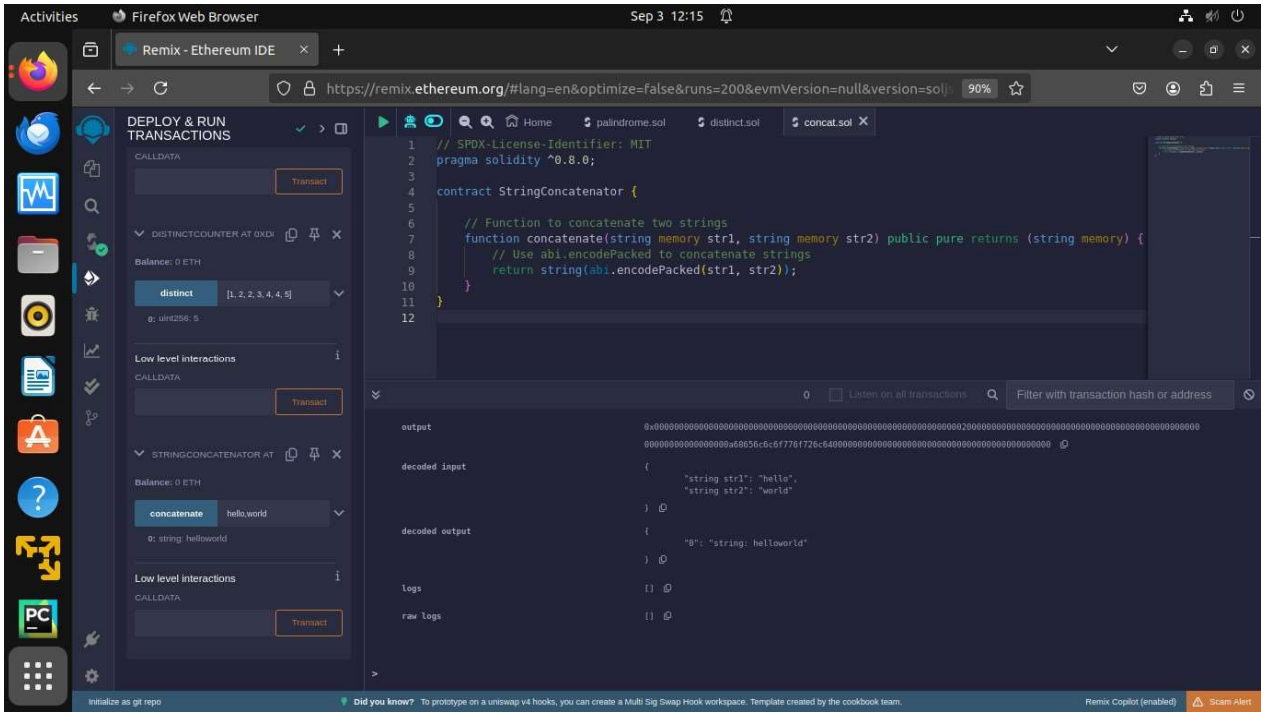

Activities
Firefox Web Browser
Sep 3 12:07
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.
DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded
Pinned Contracts (network: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
PALINDROMECHECKER AT
Balance: 0 ETH
isPalindrome
123
0: bool: false
Low level interactions
CALLDATA
Transact
Home
palindrome.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract PalindromeChecker {
// Function to check if a number is a palindrome
Listen on all transactions
Filter with transaction hash or address
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
PalindromeChecker.isPalindrome(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost
3185 gas (Cost only applies when called by a contract)
input
0x041...0007b
output
0x0000000000000000000000000000000000000000000000000000000000000000
decoded input
"uint256 number": "123"
decoded output
"0": "bool: false"
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert

Distinct -


Activities
Firefox Web Browser
Sep 3 12:12
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soli
90%
DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (99.999999999999
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
DistinctCounter - contracts/distinct.s
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Pinned Contracts (network: vm-cancun)
Home
palindrome.sol
distinct.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract DistinctCounter {
// Function to count the number of distinct elements in an array
function distinct(uint[] memory array) public pure returns (uint) {
infinite gas
// Create a temporary storage array for distinct elements
uint[] memory distinctArray = new uint[](array.length);
uint distinctCount = 0;
// Iterate through the array to populate distinctArray
for (uint i = 0; i < array.length; i++) {
bool isDistinct = true;
for (uint j = 0; j < distinctCount; j++) {
if (distinctArray[j] == array[i]) {
isDistinct = false;
break;
}
}
if (isDistinct) {
distinctArray[distinctCount] = array[i];
distinctCount++;
}
}
return distinctCount;
}
}
Listen on all transactions
Filter with transaction hash or address
input
0x1d8...00005
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert


Activities
Firefox Web Browser
Sep 3 12:13
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soli
90%
DEPLOY & RUN TRANSACTIONS
selected workspace & network
Deployed/Unpinned Contracts
PALINDROMECHECKER AT
Balance: 0 ETH
isPalindrome
123
0: bool: false
Low level interactions
CALLDATA
Transact
DISTINCTCOUNTER AT 0XDI
Balance: 0 ETH
distinct
[1, 2, 2, 3, 4, 4, 5]
0: uint256: 5
Low level interactions
CALLDATA
Transact
Home
palindrome.sol
distinct.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract DistinctCounter {
// Function to count the number of distinct elements in an array
function distinct(uint[] memory array) public pure returns (uint) {
infinite gas
// Create a temporary storage array for distinct elements
uint[] memory distinctArray = new uint[](array.length);
Listen on all transactions
Filter with transaction hash or address
execution cost
input
0x1d8...00005
output
decoded input
"uint256[] array": [
"1",
"2",
"2",
"3",
"4",
"4",
"5"
]
decoded output
"0": "uint256: 5"
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert

Concat


Activities
Firefox Web Browser
Sep 3 12:15
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljs
90%
DEPLOY & RUN TRANSACTIONS
CALLDATA
Transact
DISTINCTCOUNTER AT 0XDI
Balance: 0 ETH
distinct
[1, 2, 2, 3, 4, 4, 5]
0: uint256: 5
Low level interactions
CALLDATA
Transact
STRINGCONCATENATOR AT
Balance: 0 ETH
concatenate
hello,world
0: string: helloworld
Low level interactions
CALLDATA
Transact
Home
palindrome.sol
distinct.sol
concat.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract StringConcatenator {
// Function to concatenate two strings
function concatenate(string memory str1, string memory str2) public pure returns (string memory) {
// Use abi.encodePacked to concatenate strings
return string(abi.encodePacked(str1, str2));
}
}
0
Listen on all transactions
Filter with transaction hash or address
output
0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000
0000000000000000a68656c6c6f776f726c640000000000000000000000000000000000000000
decoded input
{
"string str1": "hello",
"string str2": "world"
}
decoded output
{
"0": "string: helloworld"
}
logs
[]
raw logs
[]
>
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert

Arnav Malvia
C-32
2103109
Blockchain
Date - 30/7/2024

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment :- 3

Aim:- Write a program in Solidity Programming to create transactions 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 proof of work or proof of stake ~~is~~ it is grouped into a block added to a blockchain, making it a permanent part of the network. Transactions are designed to be tamper proof and transparent. Each transaction 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 ~~algorithm~~ 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 ~~partici~~ participants in the network have a consistent and accurate record of transactions, making fraud and double speeding difficult.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

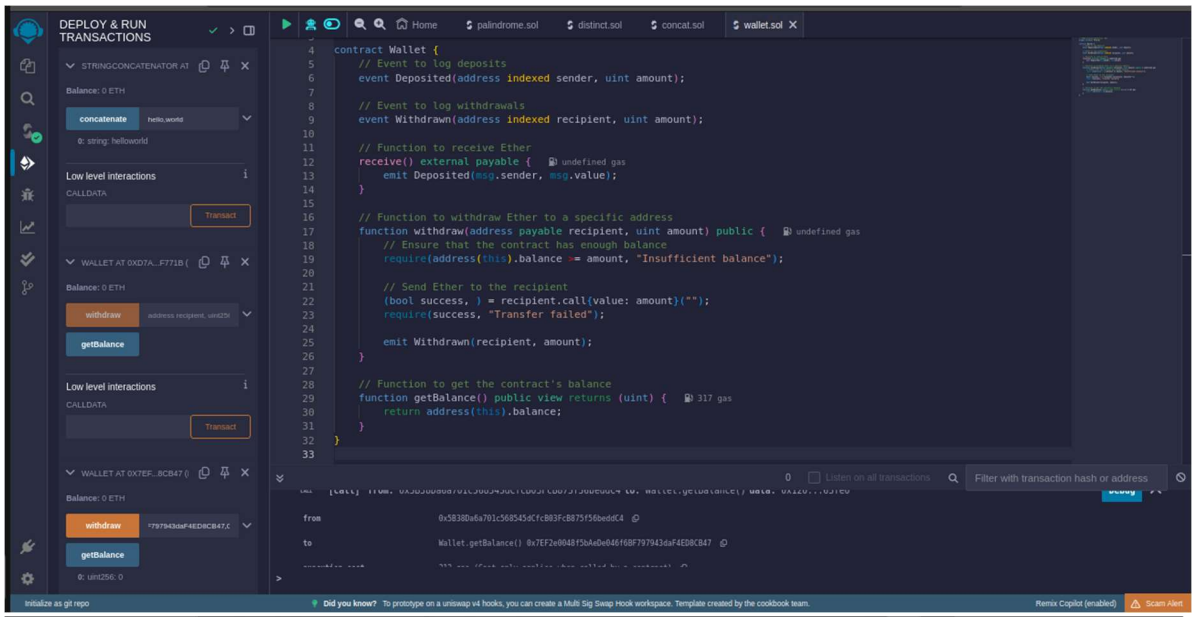
The list of transactions is a component of header. It forms a tree like structure called as Merkel tree. ~~and~~ The Merkel root is used to construct the block hash. If you change ~~all the tr~~ a transaction you need to change all the subsequent blockhash.

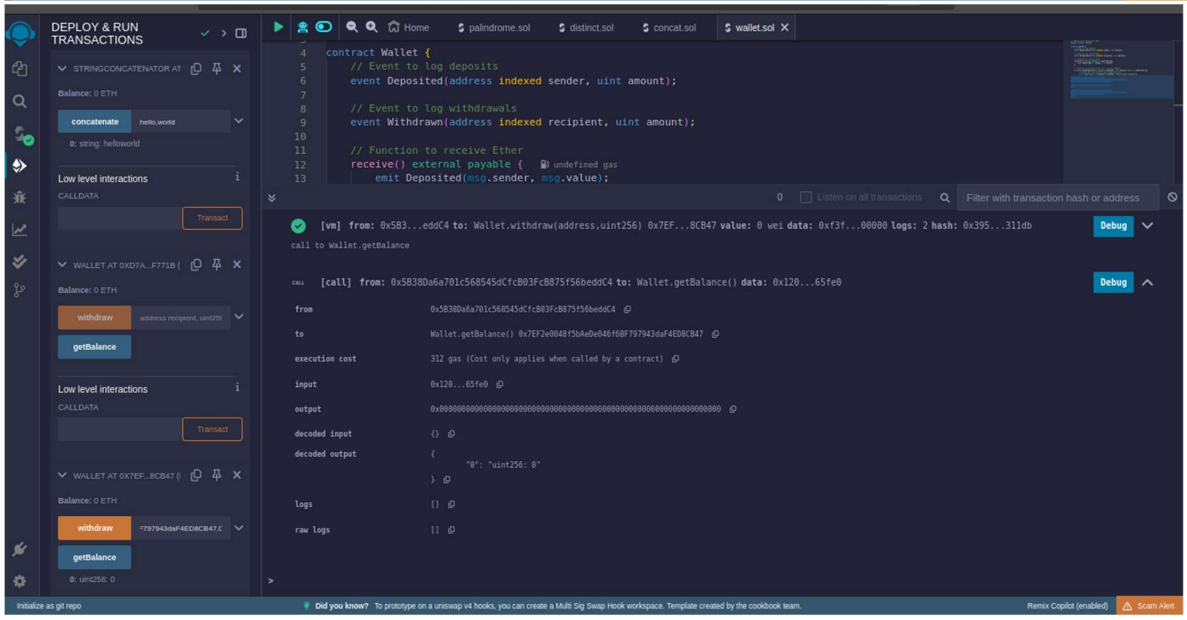
Conclusion:- Thus we have studied and implemented transactions using Solidity programming and Remix IDE.

30/9/24

Exp 3

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

contract TransactionHandler {
	// Event to log transactions
	event Sent(address indexed from, address indexed to, uint256 amount);

	// Function to receive Ether. This function is called when Ether is sent to the contract.
	receive() external payable {}

	// Function to withdraw Ether from the contract
	function withdraw(uint256 amount, address payable recipient) public {
	require(amount <= address(this).balance, "Insufficient balance in the contract");
	require(amount > 0, "Amount must be greater than zero");

	// Transfer the specified amount of Ether to the recipient
	(bool success, ) = recipient.call{value: amount}("");
	require(success, "Failed to send Ether");

	// Emit an event to log the transaction
	emit Sent(address(this), recipient, amount);
	}

	// Function to check the contract's balance
	function getBalance() public view returns (uint256) {
	return address(this).balance;
	}
}
```


DEPLOY & RUN TRANSACTIONS
Home
palindrome.sol
distinct.sol
concat.sol
wallet.sol
contract Wallet {
// Event to log deposits
event Deposited(address indexed sender, uint amount);
// Event to log withdrawals
event Withdrawn(address indexed recipient, uint amount);
// Function to receive Ether
receive() external payable {
emit Deposited(msg.sender, msg.value);
}
// Function to withdraw Ether to a specific address
function withdraw(address payable recipient, uint amount) public {
// Ensure that the contract has enough balance
require(address(this).balance >= amount, "Insufficient balance");
// Send Ether to the recipient
(bool success,) = recipient.call{value: amount}("");
require(success, "Transfer failed");
emit Withdrawn(recipient, amount);
}
// Function to get the contract's balance
function getBalance() public view returns (uint) {
return address(this).balance;
}
}
STRINGCONCATENATOR AT
Balance: 0 ETH
concatenate
hello,world
0: string: helloworld
Low level interactions
CALLDATA
Transact
WALLET AT 0XD7A...F771B (
Balance: 0 ETH
withdraw
getBalance
Low level interactions
CALLDATA
Transact
WALLET AT 0X7EF...8CB47 ()
Balance: 0 ETH
withdraw
getBalance
0: uint256: 0
Listen on all transactions
Filter with transaction hash or address
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to Wallet.getBalance() 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert


DEPLOY & RUN TRANSACTIONS
Home
palindrome.sol
distinct.sol
concat.sol
wallet.sol
contract Wallet {
// Event to log deposits
event Deposited(address indexed sender, uint amount);
// Event to log withdrawals
event Withdrawn(address indexed recipient, uint amount);
// Function to receive Ether
receive() external payable {
emit Deposited(msg.sender, msg.value);
Listen on all transactions
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Wallet.withdraw(address,uint256) 0x7EF...8CB47 value: 0 wei data: 0xf3f...00000 logs: 2 hash: 0x395...311db
Debug
call to Wallet.getBalance
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Wallet.getBalance() data: 0x120...65fe0
Debug
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to Wallet.getBalance() 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47
execution cost 312 gas (Cost only applies when called by a contract)
input 0x120...65fe0
output 0x0000000000000000000000000000000000000000000000000000000000000000
decoded input {}
decoded output { "0": "uint256: 0" }
logs []
raw logs []
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert

Arnav Malvia
C-32
2103109
Blockchain
Date - 6/8/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment :- 4

Aim :- Write a program in solidity to implement transactions by embedding of wallet.

Theory :-
Blockchain is a decentralised digital ledger that records transactions across many computers so that the record cannot be altered retroactively. It operates in a peer to peer network where each participant maintains a copy of the ledger. This ledger, immutable and transparent ensures that once recorded. In this decentralised ecosystem, smart contracts are self executing agreements coded directly into the blockchain, automating and enforcing terms without intermediaries.

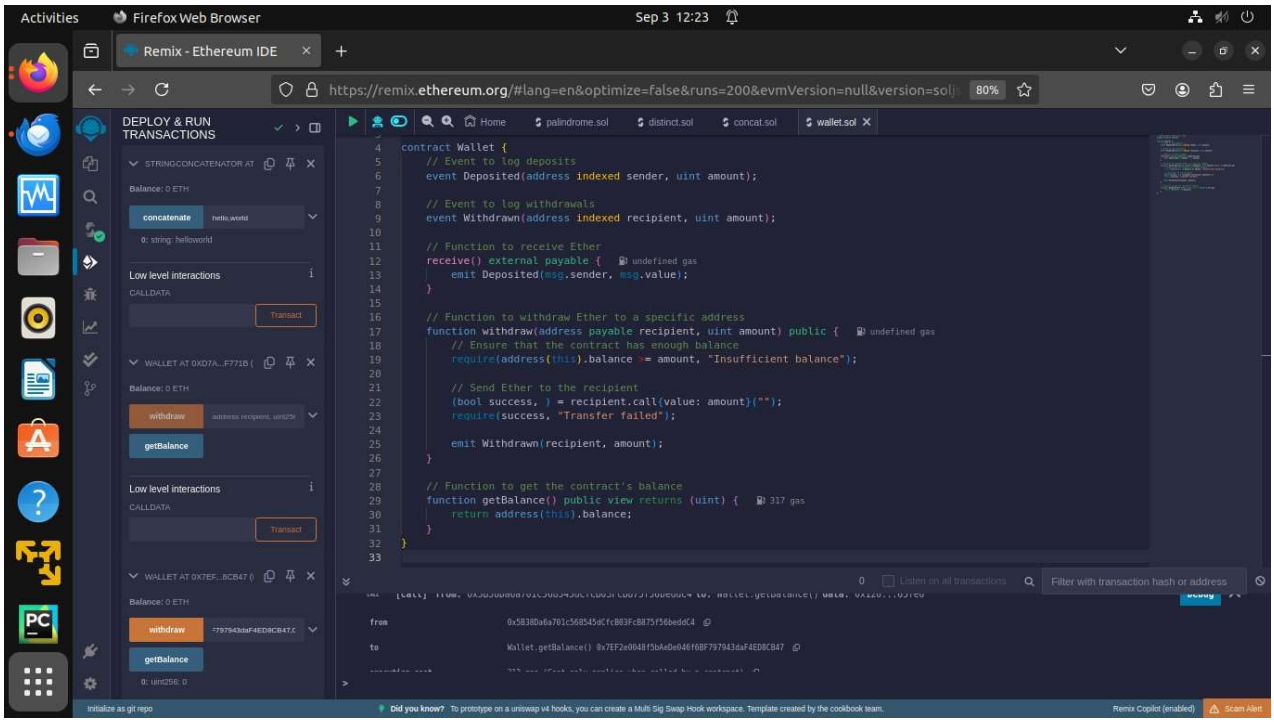
Smart contracts are self executing contracts with the terms written directly into code. They run on blockchain platforms like Ethereum. These contracts automatically enforce and execute terms of an agreement based on predefined rules and conditions. Ethereum is a blockchain platform that enables developers to deploy and run smart contracts and decentralised applications. It has its own cryptocurrency ether (ETH) which is used to pay for transaction fees and computational services on the network.

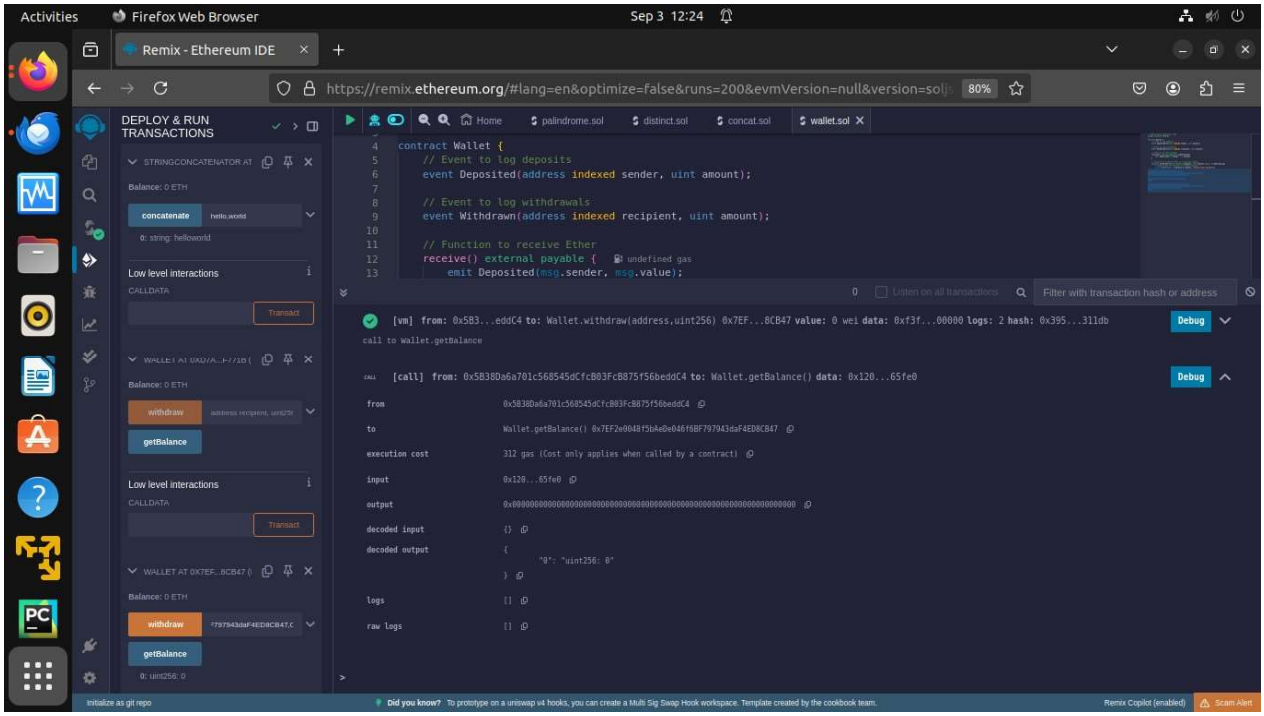
Events are used to log activities such as deposits, withdrawals and transactions, enabling real time updates for off chain applications. Mappings store user balances, allowing ~~different~~ efficient data retrieval. Functions define the contract's logic, including methods for depositing funds, withdrawing funds and transfering ether between addresses with built in checks to ensure correctness and 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.

Conclusion:- Thus we have studied and implemented transactions by embedding of wallet in solidity.

30/9/24

Exp 4


Activities
Firefox Web Browser
Sep 3 12:23
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljs
80%
DEPLOY & RUN TRANSACTIONS
Home
palindrome.sol
distinct.sol
concat.sol
wallet.sol
contract Wallet {
// Event to log deposits
event Deposited(address indexed sender, uint amount);
// Event to log withdrawals
event Withdrawn(address indexed recipient, uint amount);
// Function to receive Ether
receive() external payable {
emit Deposited(msg.sender, msg.value);
}
// Function to withdraw Ether to a specific address
function withdraw(address payable recipient, uint amount) public {
// Ensure that the contract has enough balance
require(address(this).balance >= amount, "Insufficient balance");
// Send Ether to the recipient
(bool success,) = recipient.call{value: amount}("");
require(success, "Transfer failed");
emit Withdrawn(recipient, amount);
}
// Function to get the contract's balance
function getBalance() public view returns (uint) {
return address(this).balance;
}
}
STRINGCONCATENATOR AT
Balance: 0 ETH
concatenate
hello,world
0: string: helloworld
Low level interactions
CALLDATA
Transact
WALLET AT 0XD7A...F771B (
Balance: 0 ETH
withdraw
getBalance
Low level interactions
CALLDATA
Transact
WALLET AT 0X7EF...6CB47 ()
Balance: 0 ETH
withdraw
getBalance
0: uint256: 0
Filter with transaction hash or address
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
Wallet.getBalance() 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert


Activities
Firefox Web Browser
Sep 3 12:24
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljs
80%
DEPLOY & RUN TRANSACTIONS
Home
palindrome.sol
distinct.sol
concat.sol
wallet.sol
contract Wallet {
// Event to log deposits
event Deposited(address indexed sender, uint amount);
// Event to log withdrawals
event Withdrawn(address indexed recipient, uint amount);
// Function to receive Ether
receive() external payable {
emit Deposited(msg.sender, msg.value);
STRINGCONCATENATOR AT
Balance: 0 ETH
concatenate
hello,world
0: string: helloworld
Low level interactions
CALLDATA
Transact
Balance: 0 ETH
withdraw
getBalance
Low level interactions
CALLDATA
Transact
WALLET AT 0X7EF...8CB47 ()
Balance: 0 ETH
withdraw
f797943daF4ED8CB47,c
getBalance
0: uint256: 0
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Wallet.withdraw(address,uint256) 0x7EF...8CB47 value: 0 wei data: 0xf3f...00000 logs: 2 hash: 0x395...311db
Debug
call to Wallet.getBalance
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Wallet.getBalance() data: 0x120...65fe0
Debug
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
Wallet.getBalance() 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47
execution cost
312 gas (Cost only applies when called by a contract)
input
0x120...65fe0
output
0x0000000000000000000000000000000000000000000000000000000000000000
decoded input
{}
decoded output
{
"0": "uint256: 0"
}
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
Remix Copilot (enabled)
Scam Alert

Aarnav Malvia
C-32
2103109
Blockchain
Date - 13/8/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment :- 5

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

Theory :-
Ethereum is a decentralised platform that enables the creation and execution of smart contracts and decentralised apps (dApps).

Geth is one of the three original implementations of the ethereum protocol, written in Go language.

Steps to build your private blockchain-
Step 1 - Install Geth
Geth is a UI 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 blockchain starts with a genesis (first) block. The genesis block is configured using genesis.json file for a private network.

Step 3 - Intiate the Private blockchain
We have the configuration ready for the ~~go~~ genisis block. We need to run first geth command to initialize the private blockchain.
'geth -- data dir. | ~~node~~ node1 init genesis.json'

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

~~Step 5~~ port: our node is running on this port.
http.port: This is for HTTP protocol access.

Step 5 - Add more nodes

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

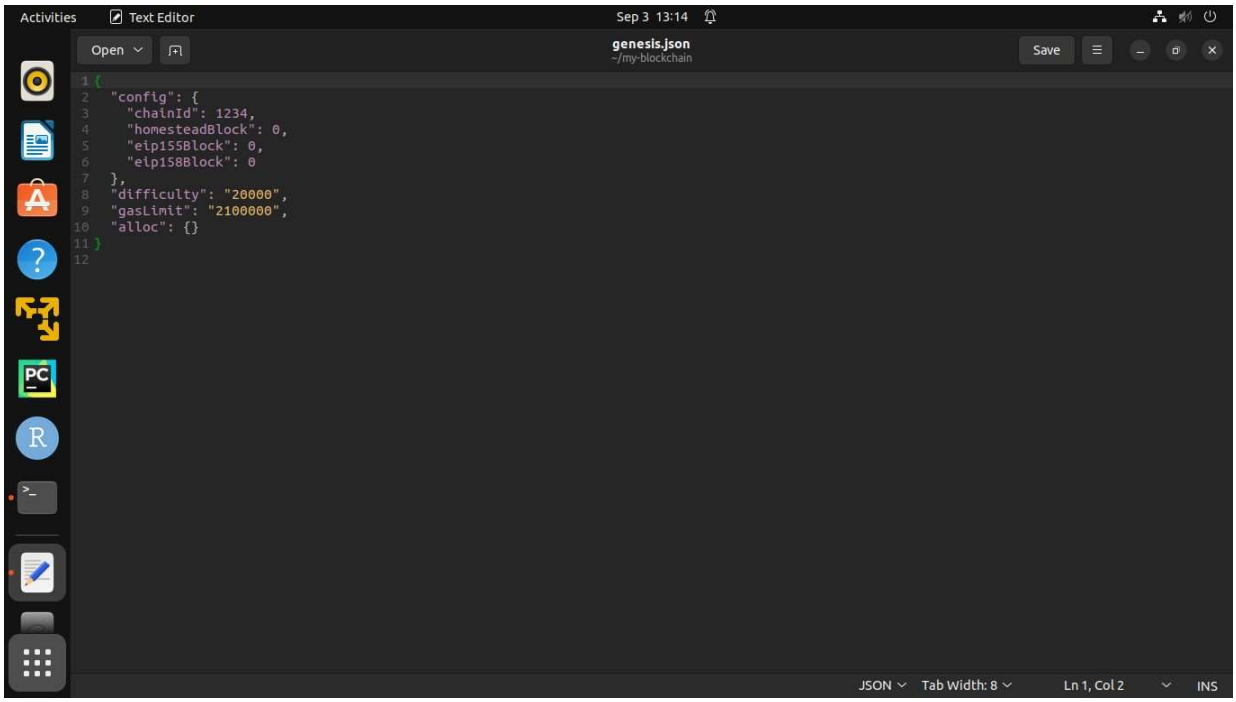
Step 7 - Mining blocks and creating transactions.

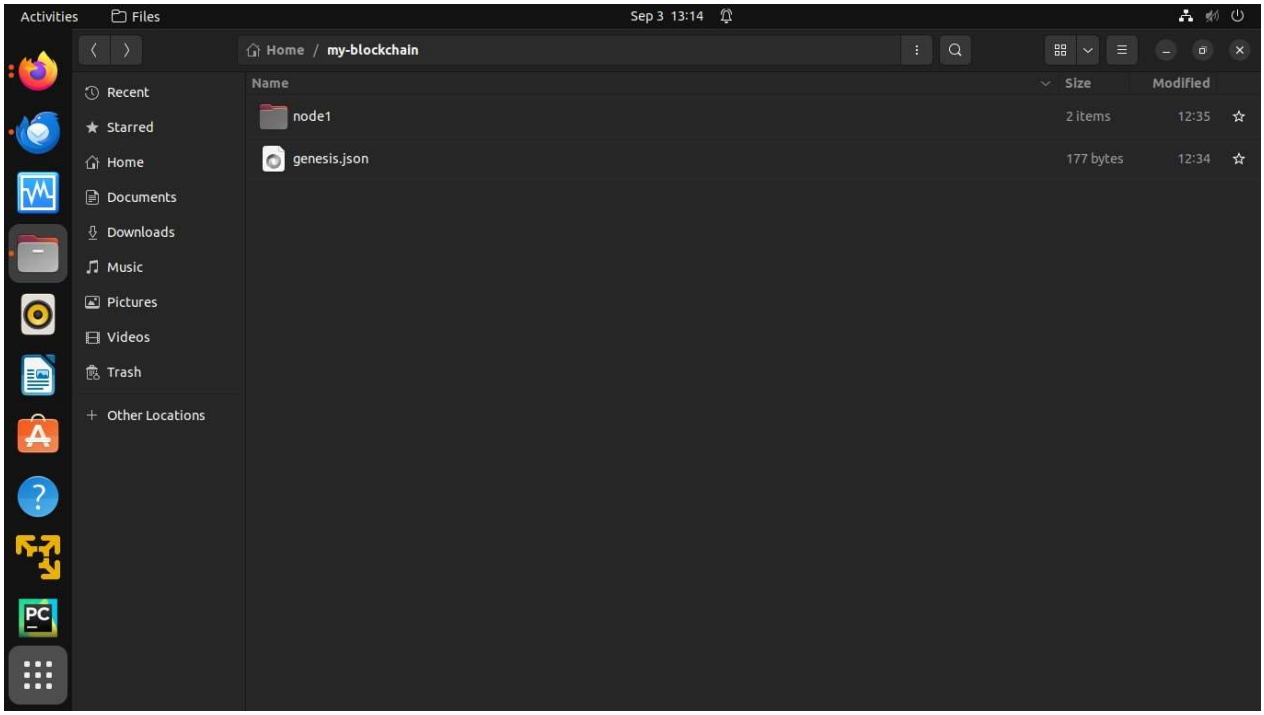
Mining is the process of validating transactions and adding them to blockchain.

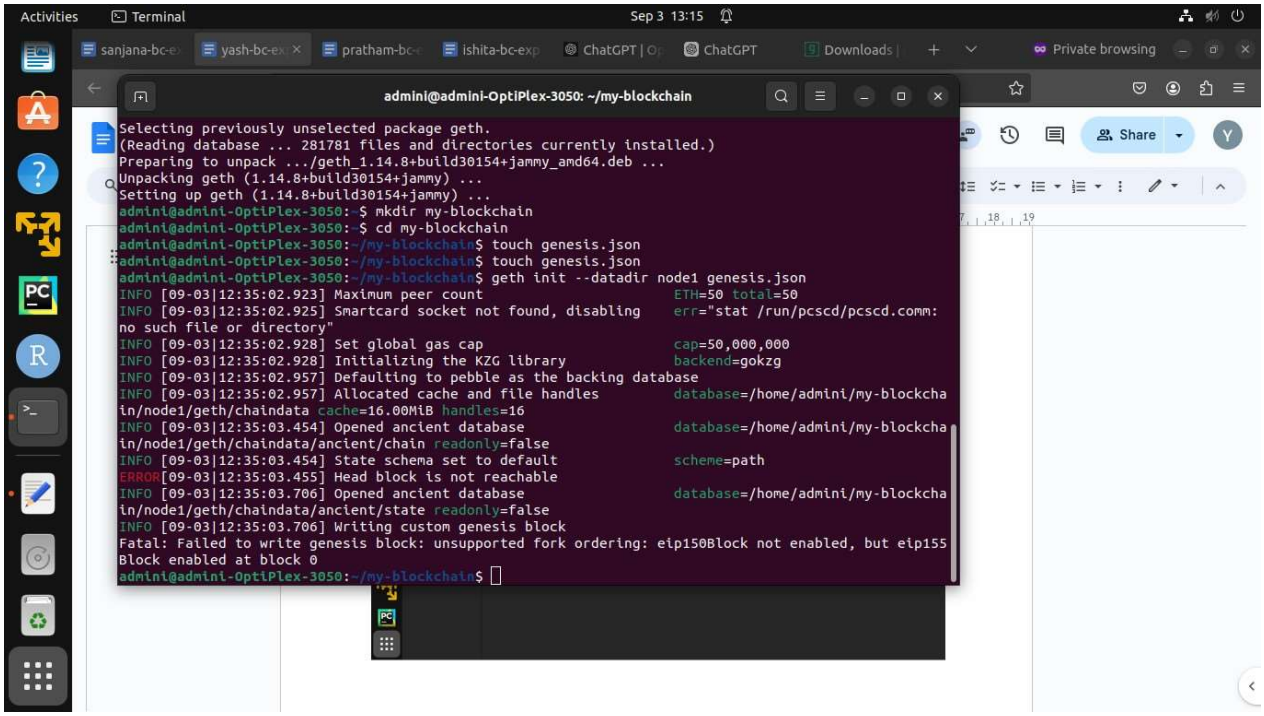
Conclusion:- Thus we have studied and implemented the blockchain platform Ethereum using Geth.

30/9/24

Exp 5


Activities
Text Editor
Sep 3 13:14
Open
genesis.json
~/my-blockchain
Save
{
"config": {
"chainId": 1234,
"homesteadBlock": 0,
"eip155Block": 0,
"eip158Block": 0
},
"difficulty": "20000",
"gasLimit": "2100000",
"alloc": {}
}
JSON
Tab Width: 8
Ln 1, Col 2
INS


Activities
Files
Sep 3 13:14
Home / my-blockchain
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
Name
Size
Modified
node1
2 items
12:35
genesis.json
177 bytes
12:34


Activities
Terminal
Sep 3 13:15
admini@admini-OptiPlex-3050: ~/my-blockchain
Selecting previously unselected package geth.
(Reading database ... 281781 files and directories currently installed.)
Preparing to unpack .../geth_1.14.8+build30154+jammy_amd64.deb ...
Unpacking geth (1.14.8+build30154+jammy) ...
Setting up geth (1.14.8+build30154+jammy) ...
admini@admini-OptiPlex-3050:~$ mkdir my-blockchain
admini@admini-OptiPlex-3050:~$ cd my-blockchain
admini@admini-OptiPlex-3050:~/my-blockchain$ touch genesis.json
admini@admini-OptiPlex-3050:~/my-blockchain$ touch genesis.json
admini@admini-OptiPlex-3050:~/my-blockchain$ geth init --datadir node1 genesis.json
INFO [09-03|12:35:02.923] Maximum peer count ETH=50 total=50
INFO [09-03|12:35:02.925] Smartcard socket not found, disabling err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [09-03|12:35:02.928] Set global gas cap cap=50,000,000
INFO [09-03|12:35:02.928] Initializing the KZG library backend=gokzg
INFO [09-03|12:35:02.957] Defaulting to pebble as the backing database
INFO [09-03|12:35:02.957] Allocated cache and file handles database=/home/admini/my-blockchain/node1/geth/chaindata cache=16.00MiB handles=16
INFO [09-03|12:35:03.454] Opened ancient database database=/home/admini/my-blockchain/node1/geth/chaindata/ancient/chain readonly=false
INFO [09-03|12:35:03.454] State schema set to default scheme=path
ERROR[09-03|12:35:03.455] Head block is not reachable
INFO [09-03|12:35:03.706] Opened ancient database database=/home/admini/my-blockchain/node1/geth/chaindata/ancient/state readonly=false
INFO [09-03|12:35:03.706] Writing custom genesis block
Fatal: Failed to write genesis block: unsupported fork ordering: eip150Block not enabled, but eip155Block enabled at block 0
admini@admini-OptiPlex-3050:~/my-blockchain$
Private browsing
Share

Aarnav Malvia
C-32
2103109
Blockchain
Date - 20/8/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment :- 6

Aim :- Show implementation of blockchain platform Ganache.

Theory :-
Ganache is a personal blockchain for ethereum development than ~~can~~ 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 count its Ganache. ~~after~~
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 purposes.

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

```
pragma solidity 0.8.0;
contraction Addition;
    function add(int a, int b)
        public pure returns (int) {
            return a+b;
}
```

Step 5- Connect Remid IDE with Ganache link
Remix IDE, deploy the smart contract to the 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.

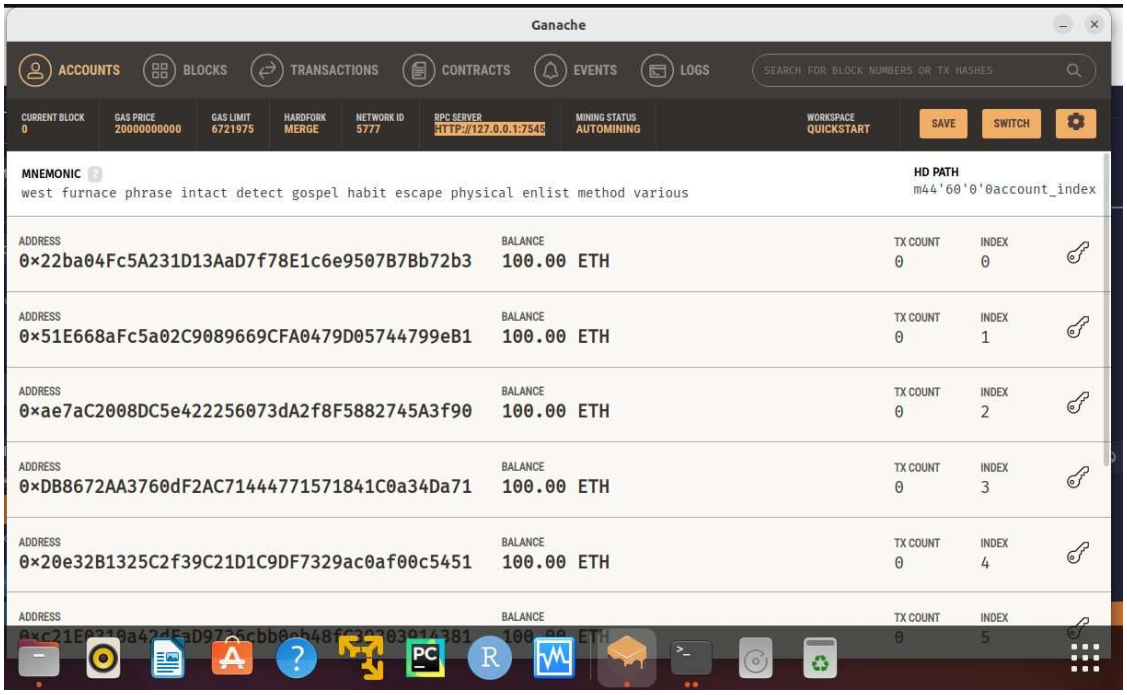
Step 7 - Observe ether transactions
Monitor the effects and contract deployment on account balances.

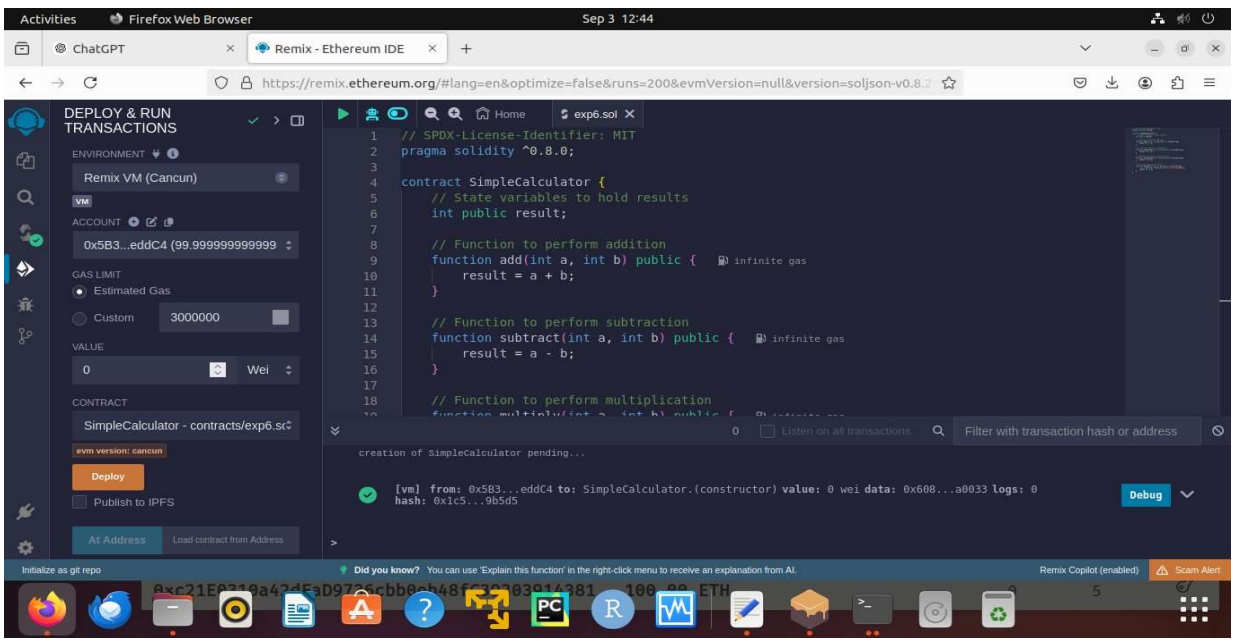
Conclusion:- Thus we have studied and implemented blockchain platform Ganache.

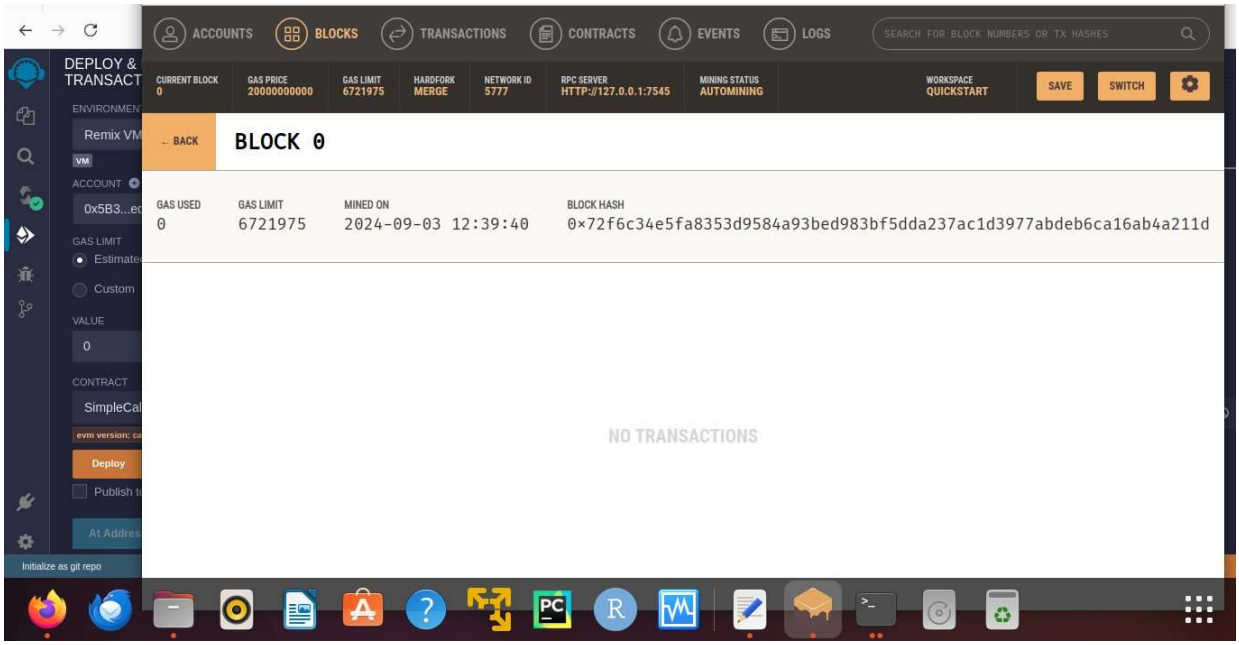
30/9/24

Exp 6 ganache

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

contract SimpleCalculator {
	// State variables to hold results
	int public result;

	// Function to perform addition
	function add(int a, int b) public {
	result = a + b;
	}

	// Function to perform subtraction
	function subtract(int a, int b) public {
	result = a - b;
	}

	// Function to perform multiplication
	function multiply(int a, int b) public {
	result = a * b;
	}

	// Function to perform division
	function divide(int a, int b) public {
	require(b != 0, "Division by zero is not allowed");
	result = a / b;
	}
}
```


Ganache
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 QUICKSTART
SAVE
SWITCH
MNEMONIC
west furnace phrase intact detect gospel habit escape physical enlist method various
HD PATH
m44'60'0'0account_index
ADDRESS 0x22ba04Fc5A231D13AaD7f78E1c6e9507B7Bb72b3 BALANCE 100.00 ETH TX COUNT 0 INDEX 0
ADDRESS 0x51E668aFc5a02C9089669CFA0479D05744799eB1 BALANCE 100.00 ETH TX COUNT 0 INDEX 1
ADDRESS 0xae7aC2008DC5e422256073dA2f8F5882745A3f90 BALANCE 100.00 ETH TX COUNT 0 INDEX 2
ADDRESS 0xDB8672AA3760dF2AC71444771571841C0a34Da71 BALANCE 100.00 ETH TX COUNT 0 INDEX 3
ADDRESS 0x20e32B1325C2f39C21D1C9DF7329ac0af00c5451 BALANCE 100.00 ETH TX COUNT 0 INDEX 4
ADDRESS BALANCE TX COUNT 0 INDEX 5


Activities
Firefox Web Browser
Sep 3 12:44
ChatGPT
Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.2
DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (99.999999999999
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
SimpleCalculator - contracts/exp6.sc
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Home
exp6.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleCalculator {
// State variables to hold results
int public result;
// Function to perform addition
function add(int a, int b) public {
infinite gas
result = a + b;
}
// Function to perform subtraction
function subtract(int a, int b) public {
infinite gas
result = a - b;
}
// Function to perform multiplication
Listen on all transactions
Filter with transaction hash or address
creation of SimpleCalculator pending...
[vm] from: 0x5B3...eddC4 to: SimpleCalculator.(constructor) value: 0 wei data: 0x608...a0033 logs: 0
hash: 0x1c5...9b5d5
Debug
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert


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 QUICKSTART
SAVE
SWITCH
BACK
BLOCK 0
GAS USED 0
GAS LIMIT 6721975
MINED ON 2024-09-03 12:39:40
BLOCK HASH 0x72f6c34e5fa8353d9584a93bed983bf5dda237ac1d3977abdeb6ca16ab4a211d
NO TRANSACTIONS
DEPLOY &
TRANSACT
ENVIRONMEN
Remix VM
VM
ACCOUNT
0x5B3...ec
GAS LIMIT
Estimate
Custom
VALUE
0
CONTRACT
SimpleCal
evm version: ca
Deploy
Publish t
At Addres
Initialize as git repo

Aarnav Malvia
2103109-C-32
Blockchain
Date - 27/8/24

Experiment - 07

Case Study: Walmart's Use of Hyperledger Fabric for Food Supply Chain Management

Introduction

Walmart, one of the largest retail companies in the world, faced a significant challenge in ensuring the safety, traceability, and quality of the food products in its vast supply chain. Traditional methods of tracking the origins and journey of food were inefficient, time-consuming, and prone to error, especially in cases of contamination or foodborne illness outbreaks.

To tackle these challenges, Walmart collaborated with **IBM** to implement **Hyperledger Fabric**, a permissioned blockchain framework. This initiative aimed to enhance transparency, traceability, and accountability across Walmart's global food supply chain.

Problem Statement

Walmart's food supply chain includes multiple stakeholders, such as farmers, suppliers, distributors, and retailers, creating a complex network. In the event of contamination, it would take Walmart days, or even weeks, to trace the source of a specific product, leading to:

- **Public health risks**: Delays in identifying the source of contamination could allow unsafe food to remain on store shelves, risking consumer safety.
- **Financial loss**: Inefficient recalls led to wasted resources and damaged reputation.
- **Lack of transparency**: Consumers were increasingly demanding to know the origin and journey of the food products they purchase.

- **Fraud and inefficiencies**: A lack of standardized processes and manual record-keeping contributed to errors, fraud, and delays in the supply chain.

Solution: Implementing Hyperledger Fabric

Collaboration with IBM and Hyperledger Fabric

Walmart collaborated with **IBM** to build a blockchain-based solution using **Hyperledger Fabric**. The blockchain was specifically designed to create an immutable, transparent, and efficient tracking system for food items. This permissioned blockchain allowed only verified and trusted participants to access the ledger.

Key Features of the Solution:

1. **End-to-End Traceability**: From the farm to the retail shelf, each stakeholder records the status of the product in real-time, ensuring that the product's journey is fully traceable.
2. **Immutability**: Once data is entered into the blockchain, it is permanent and tamper-proof, ensuring the integrity of information across the supply chain.
3. **Smart Contracts (Chaincode)**: Automated business processes were encoded into smart contracts, allowing the system to enforce pre-agreed terms and trigger actions such as shipments, payments, or alerts based on the current state of the product.
4. **Permissioned Access**: Hyperledger Fabric's permissioned framework ensured that only authorized stakeholders could view and add data, protecting sensitive business information while ensuring accountability.

Implementation Steps:

1. **Pilot Projects**: Walmart initially piloted the blockchain solution for two products—**pork** in China and **mangoes** in the U.S. These two

products were chosen for their global supply chains and the complexities associated with tracking them.

2. **Data Entry**: Every point in the supply chain, from farmers to Walmart's stores, was required to input data about the product's journey into the blockchain. For example, a mango supplier would input data about the harvest date, batch number, and shipping details onto the blockchain.
3. **Real-Time Tracking**: As the product moved through the supply chain, each entity—distributors, storage facilities, and transporters—updated the blockchain with real-time data, creating a comprehensive history of each product.
4. **Customer Transparency**: Once the system was implemented, Walmart provided consumers with the ability to scan QR codes on certain food items. This allowed customers to view the entire journey of the food product, from the farm to the store shelf.

Results and Benefits

1. Drastically Reduced Traceability Time

Before implementing blockchain, it would take Walmart **7 days** to trace the source of a contaminated food item. With Hyperledger Fabric, Walmart can now trace the source in **2.2 seconds**. This rapid traceability has significantly reduced the risks associated with foodborne illnesses and contamination.

2. Increased Efficiency

By automating processes with smart contracts and reducing reliance on manual documentation, Walmart streamlined its supply chain operations. Disputes regarding shipments, payments, or product quality could be resolved much faster, leading to improved operational efficiency.

3. Enhanced Consumer Trust

Consumers today are more interested in knowing where their food comes from. By offering transparency through blockchain, Walmart strengthened consumer confidence. Shoppers could now verify the origin, journey, and quality of the food they were buying.

4. Reduced Fraud and Waste

The immutable nature of blockchain helped to reduce fraud in the supply chain. Because all stakeholders were required to enter data that could not be altered, instances of falsified records or product substitution were minimized. This increased transparency also helped reduce food waste, as mislabeling or mishandling could be quickly identified and corrected.

5. Improved Supply Chain Collaboration

The blockchain solution encouraged better collaboration between Walmart and its suppliers, as all parties had access to the same data. This real-time information sharing allowed for more accurate forecasts, inventory management, and quality control.

Conclusion

Walmart's implementation of Hyperledger Fabric demonstrated how blockchain technology can transform the supply chain, especially in industries where transparency and traceability are critical. By significantly improving the speed of tracking products through the supply chain, reducing fraud, and increasing consumer trust, Walmart successfully showcased the potential of blockchain in improving food safety and supply chain efficiency.

This case study is a benchmark for how enterprises can leverage permissioned blockchain solutions like Hyperledger Fabric to solve real-world challenges, increase operational efficiency, and foster trust in the marketplace.

18/10/24

Arnav Maluia
C-32
2103109
Blockchain
Date - 3/9/24

Experiment - 08

Case Study: Corda Blockchain Platform

Introduction

Corda is an open-source blockchain platform developed by R3, specifically designed for businesses. Unlike traditional blockchain platforms, which are often public and generalized, Corda focuses on enabling secure and private transactions among known parties. This makes it particularly suited for industries such as finance, insurance, and healthcare, where privacy and compliance are crucial.

Key Features of Corda

- **Privacy**: Corda ensures that only the parties involved in a transaction can see the details, which contrasts with the transparency often found in other blockchains.
- **Smart Contracts**: Corda supports the use of smart contracts, allowing businesses to automate workflows and ensure compliance with business rules.
- **Interoperability**: Corda is designed to interact with other Corda networks and can integrate with existing business processes.
- **Notary Service**: Corda uses a notary service to prevent double-spending and ensure transaction integrity without requiring global consensus on the entire network.

Use Cases

1. Financial Services

Company: HSBC

Implementation: HSBC has leveraged Corda to facilitate cross-border payments through a platform called "Corda Settler." This solution allows for

real-time settlements between parties using both digital currencies and traditional fiat currencies.

Outcome: The use of Corda has significantly reduced settlement times, enhancing liquidity and efficiency in cross-border transactions.

2. Trade Finance

Company: ING

Implementation: ING has utilized Corda for a trade finance solution that automates the issuance of letters of credit. The platform allows multiple parties, including banks and suppliers, to interact securely and efficiently.

Outcome: This implementation has reduced processing times from days to hours, lowering costs and improving transparency in the trade finance process.

3. Healthcare

Company: Mayo Clinic

Implementation: Mayo Clinic has explored using Corda to create a secure and interoperable platform for managing patient data and clinical trials. By leveraging the privacy features of Corda, healthcare providers can share sensitive patient information while maintaining compliance with regulations.

Outcome: Enhanced data sharing capabilities among healthcare providers, leading to improved patient outcomes and streamlined clinical trials.

Technical Overview

- **Architecture**: Corda's architecture is designed around a shared ledger that is accessible only to relevant parties. Each node in the network maintains its own copy of the ledger, which promotes efficiency and scalability.

- **Consensus Mechanism**: Corda uses a unique consensus model that does not require all nodes to reach an agreement on every transaction, allowing for faster processing times.
- **Smart Contract Language**: Corda supports smart contracts written in Kotlin, Java, and other JVM languages, making it accessible for developers familiar with these technologies.

Challenges

- **Adoption Barrier**: Corda's focus on private networks may limit its appeal for projects requiring broader public participation, potentially hindering adoption.
- **Complexity in Integration**: Integrating Corda with existing legacy systems can be complex and may require substantial investment in infrastructure.
- **Limited Public Awareness**: Compared to more popular platforms like Ethereum, Corda may not have the same level of recognition among developers and businesses, which could slow down its adoption.

Conclusion

Corda offers a compelling solution for businesses that require privacy and efficiency in their transactions. Its unique focus on enterprise use cases, along with features like smart contracts and a robust notary service, positions it as a valuable tool in the blockchain ecosystem. The successful implementations in financial services, trade finance, and healthcare demonstrate Corda's potential to transform traditional processes and drive innovation in various sectors.

As industries increasingly seek solutions that enhance security, compliance, and efficiency, Corda's continued development and support from R3 could lead to broader adoption and more diverse use cases in the future.

18/10/24

Arnav Mahria
C-32
2103109
Blockchain
Date- 24/9/24

Mini Project
PayBC: Blockchain-Based Payment System

Aim:

The primary objective of this project, PayBC, is to develop a secure and efficient digital payment system that leverages blockchain technology for transaction verification and security. By implementing Ethereum blockchain and Aadhar verification, the system aims to provide users with a transparent, immutable, and decentralized platform for conducting financial transactions while ensuring user authenticity and seamless payment processing through QR codes.

Theory and Technology Implementation

PayBC utilizes a comprehensive tech stack and follows a specific workflow to ensure secure and efficient transactions:

Core Technologies

1. Blockchain Infrastructure
   - Ethereum blockchain (Sepolia testnet)
   - Smart Contracts developed in Solidity
   - Web3.js for blockchain interaction
   - MetaMask for wallet management and user authentication

2. Identity Verification
   - Aadhar verification using anonymous packages
   - Cryptographic techniques for secure identity validation

3. Transaction Processing
   - QR code generation for payment facilitation
   - Ethereum addresses for transaction routing
   - Sepolia network listeners for real-time transaction monitoring

System Workflow

1. User Authentication

- Users connect their MetaMask wallet to the application
- System generates a unique session for the user

2. Identity Verification Process
   - User submits Aadhar details
   - Anonymous packages process the verification
   - System confirms validity of Aadhar information
   - Result stored securely in the system

3. Payment Flow
   - Post verification, users can initiate transactions
   - System generates QR codes containing:
     * Ethereum address
     * Transaction details
   - Recipient can scan QR code to receive payment

4. Transaction Monitoring
   - Sepolia network listeners actively monitor the specified Ethereum address
   - Real-time updates on transaction status
   - Smart contracts validate and process transactions

Technical Implementation Details

1. Frontend Development
   - React.js for user interface
   - Web3 integration for MetaMask connectivity
   - QR code generation library

2. Backend Infrastructure
   - Node.js server
   - Express.js framework

- MongoDB for user data storage
- Sepolia testnet node connection

Security Measures

1. Blockchain Security
   - Immutable transaction records
   - Cryptographic verification of transactions
   - Decentralized validation through Ethereum network

2. Data Privacy
   - Hashing of Aadhar information
   - Anonymous verification packages
   - Secure storage of user data

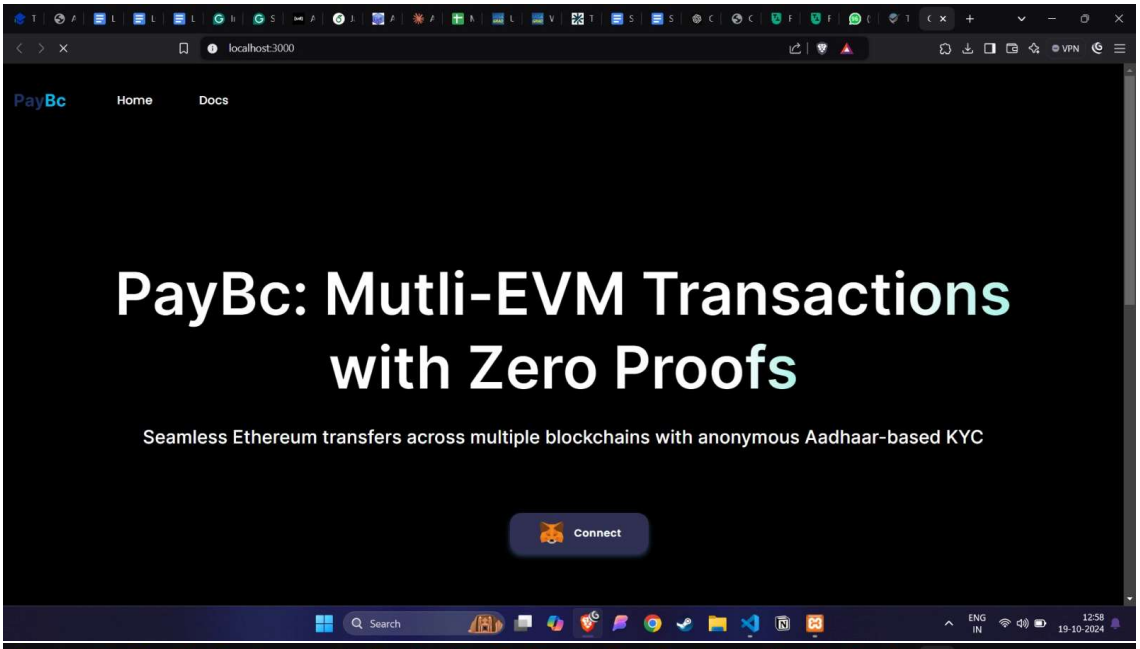
3. Transaction Security
   - MetaMask's built-in security features
   - QR code encryption
   - Real-time transaction monitoring

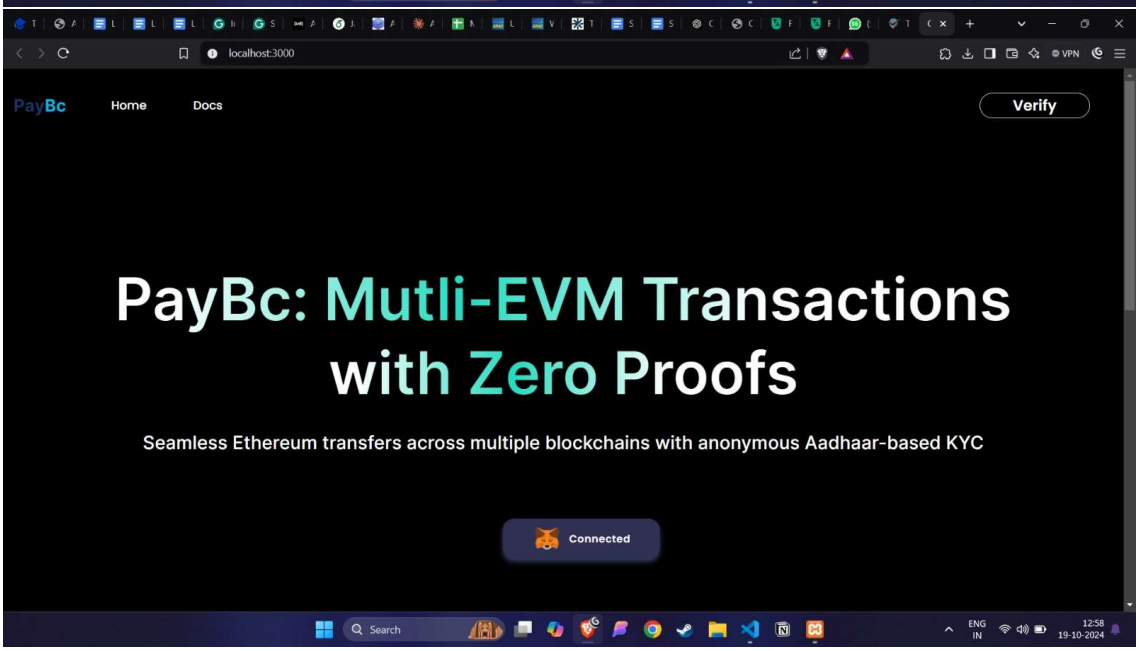
Conclusion:

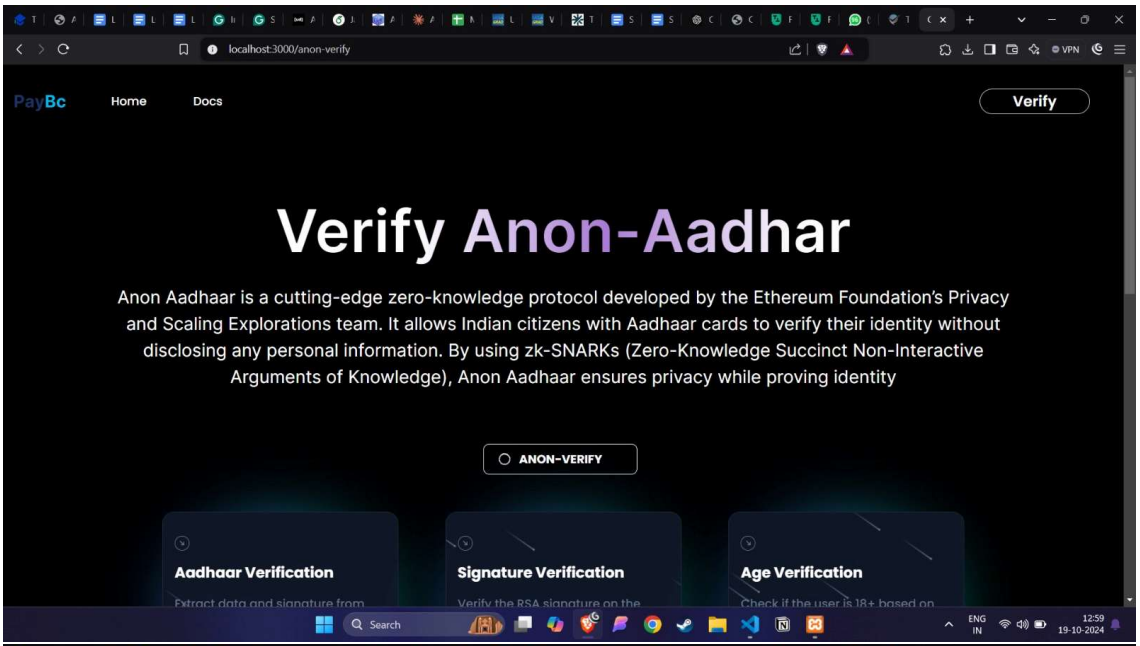
PayBC successfully demonstrates the practical implementation of blockchain technology in everyday payment systems. By utilizing the Ethereum network (Sepolia testnet) and incorporating Aadhar verification, the project achieves a balance between security, usability, and reliability. The integration of MetaMask for user authentication, coupled with QR code generation for payments, provides a user-friendly interface while maintaining the robust security features of blockchain technology. The system's ability to monitor transactions in real-time on the Sepolia network ensures transparency and reliability. This project not only showcases the potential of blockchain in financial applications but also provides a working model for secure, verified, and efficient digital transactions. The successful development and implementation of PayBC paves the way for future advancements in blockchain-based payment solutions, particularly in environments where both transaction security and user verification are paramount.

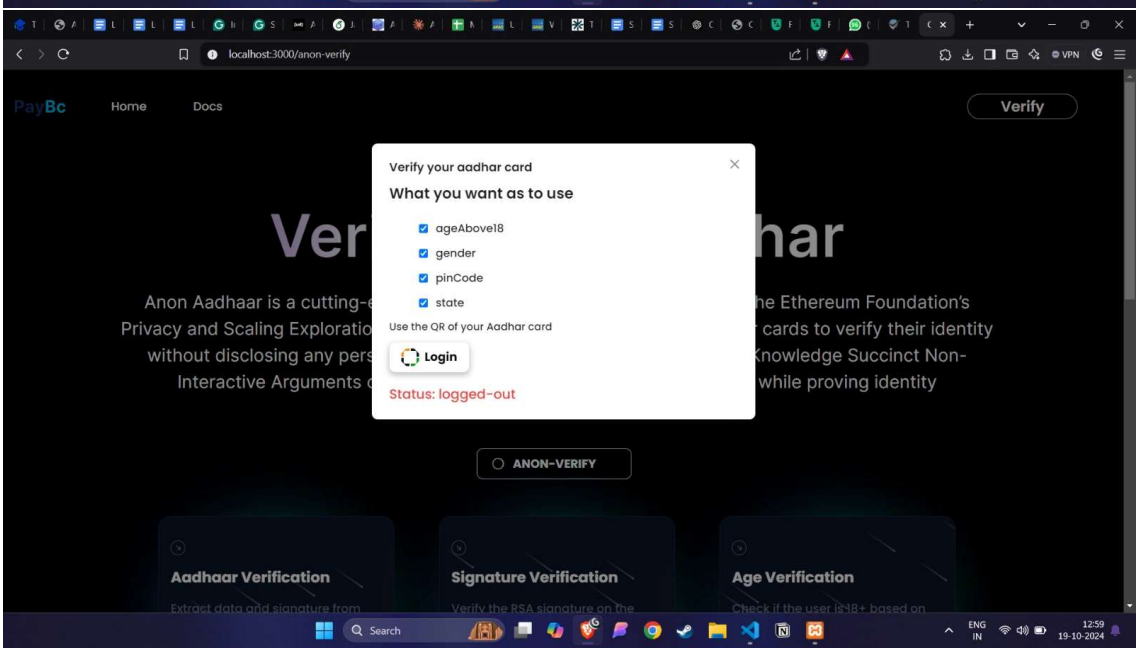
18/10/24

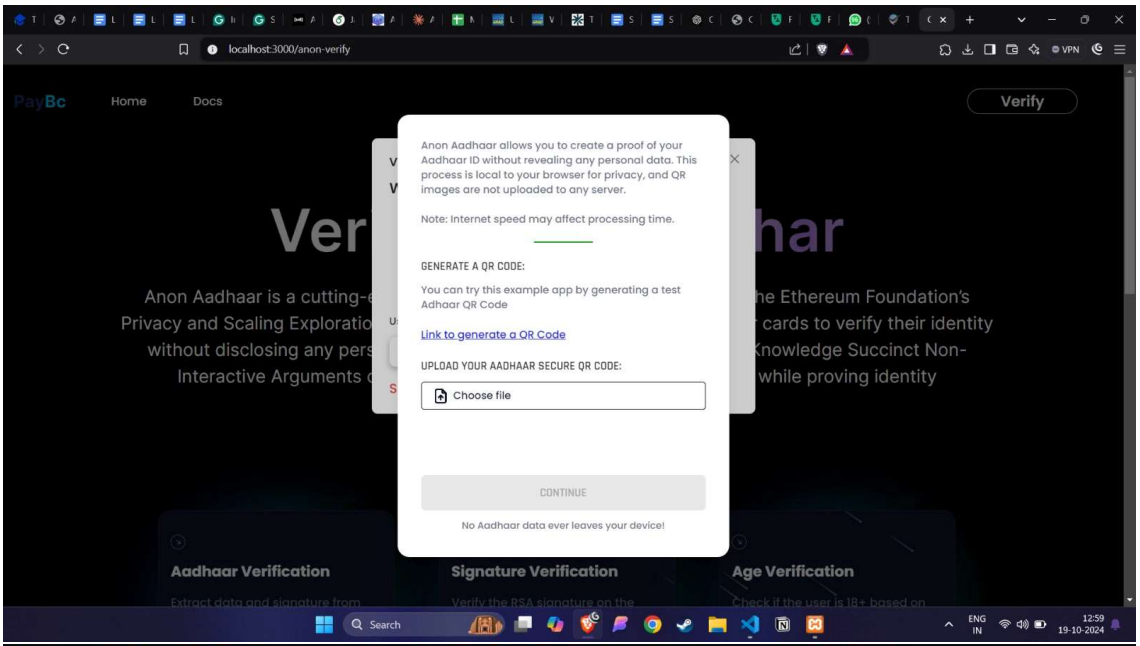
Output :-

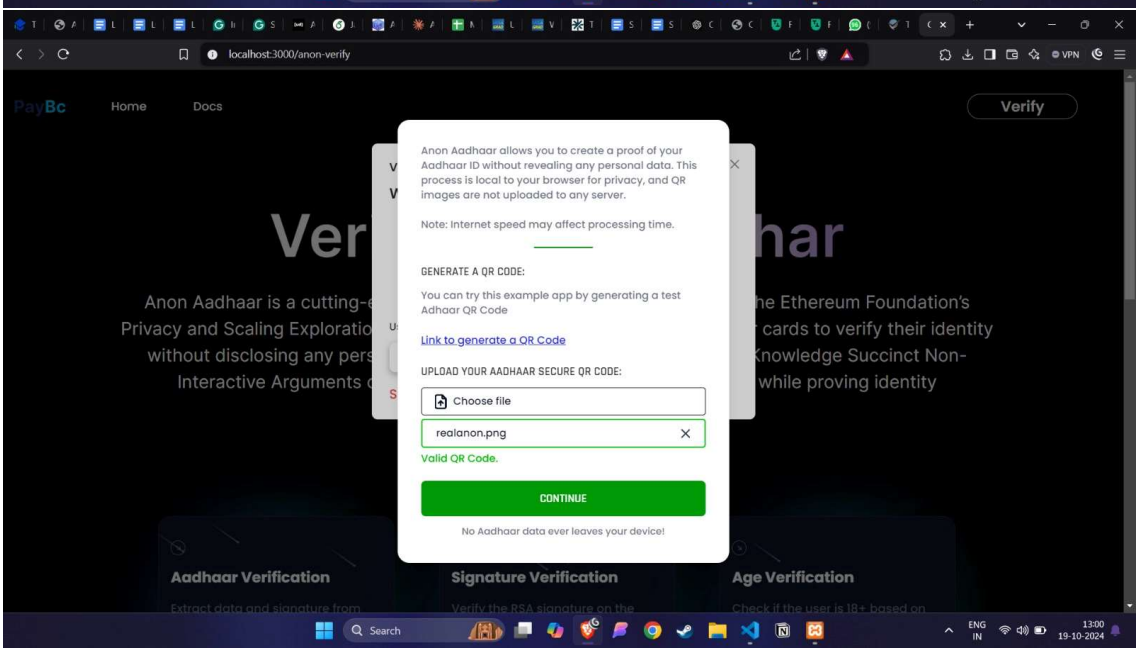

localhost:3000
PayBc
Home
Docs
PayBc: Mutli-EVM Transactions with Zero Proofs
Seamless Ethereum transfers across multiple blockchains with anonymous Aadhaar-based KYC
Connect

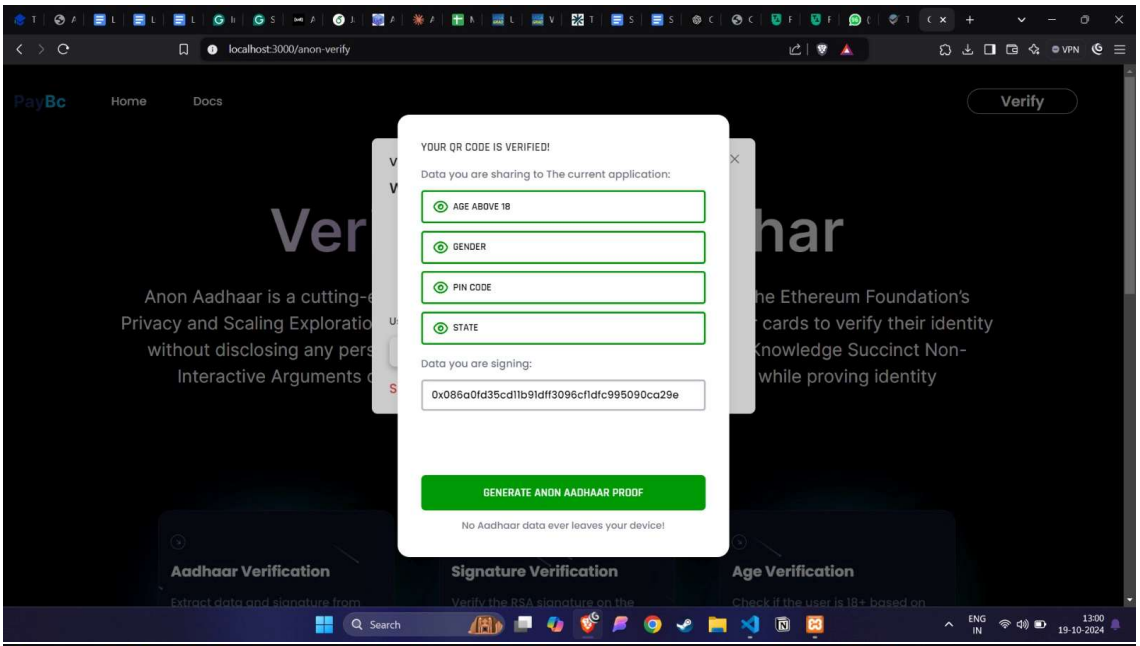

localhost:3000
PayBc
Home
Docs
Verify
PayBc: Mutli-EVM Transactions with Zero Proofs
Seamless Ethereum transfers across multiple blockchains with anonymous Aadhaar-based KYC
Connected

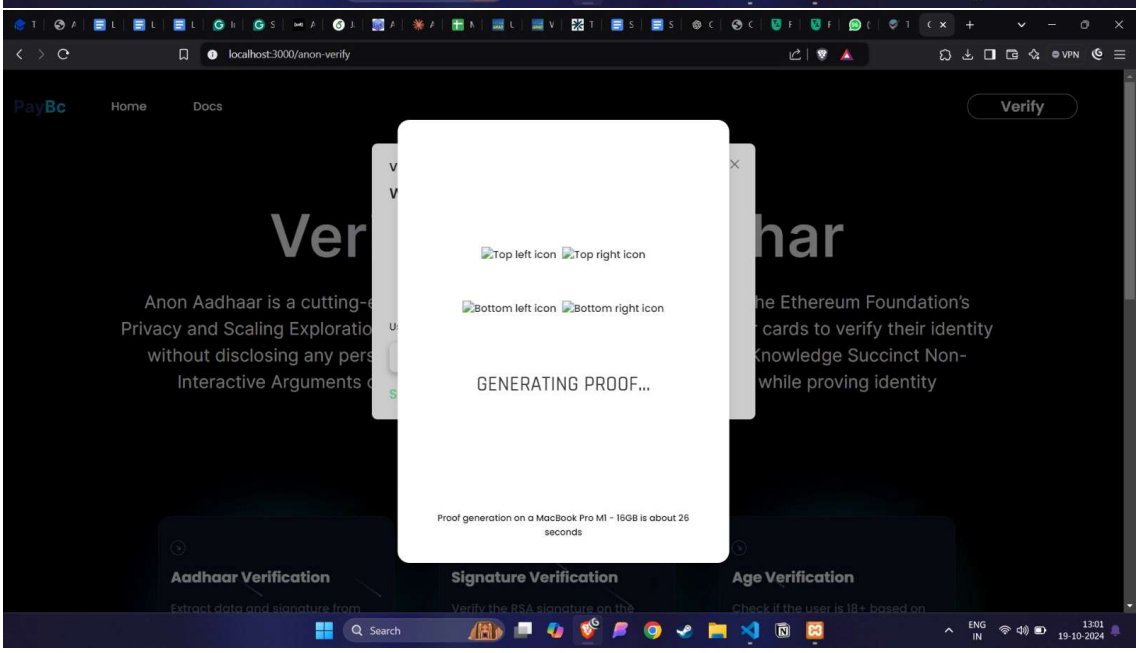

localhost:3000/anon-verify
PayBc
Home
Docs
Verify
Verify Anon-Aadhar
Anon Aadhaar is a cutting-edge zero-knowledge protocol developed by the Ethereum Foundation's Privacy and Scaling Explorations team. It allows Indian citizens with Aadhaar cards to verify their identity without disclosing any personal information. By using zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge), Anon Aadhaar ensures privacy while proving identity
ANON-VERIFY
Aadhaar Verification
Signature Verification
Age Verification

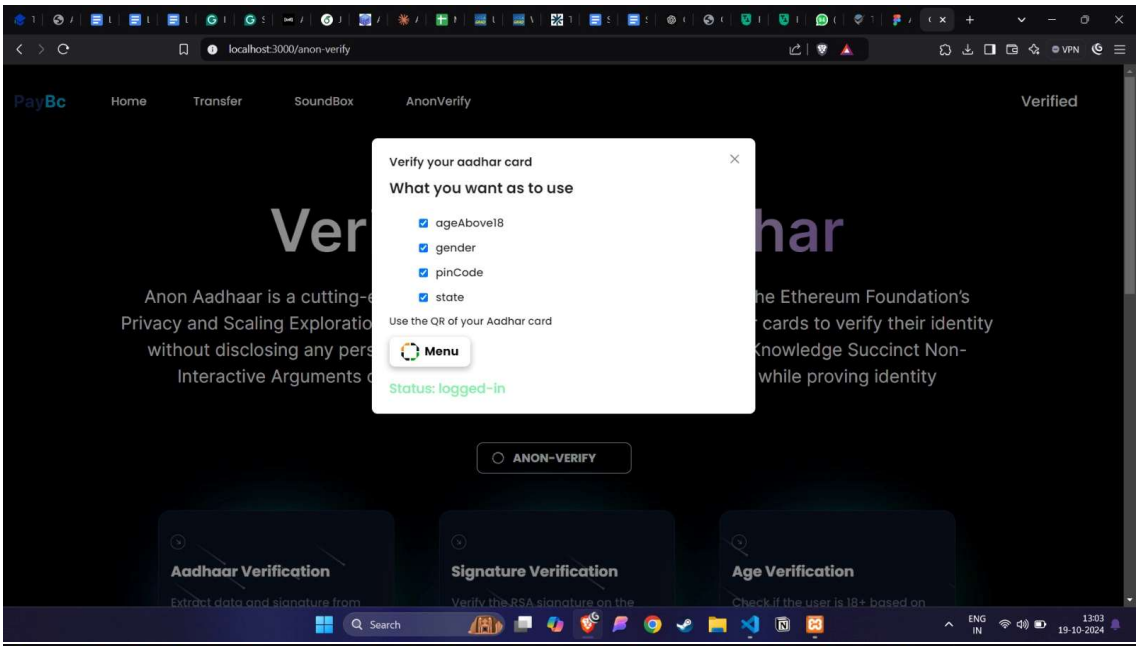

localhost:3000/anon-verify
PayBc
Home
Docs
Verify
Verify your aadhar card
What you want as to use
ageAbove18
gender
pinCode
state
Use the QR of your Aadhar card
Login
Status: logged-out
ANON-VERIFY
Aadhaar Verification
Signature Verification
Age Verification

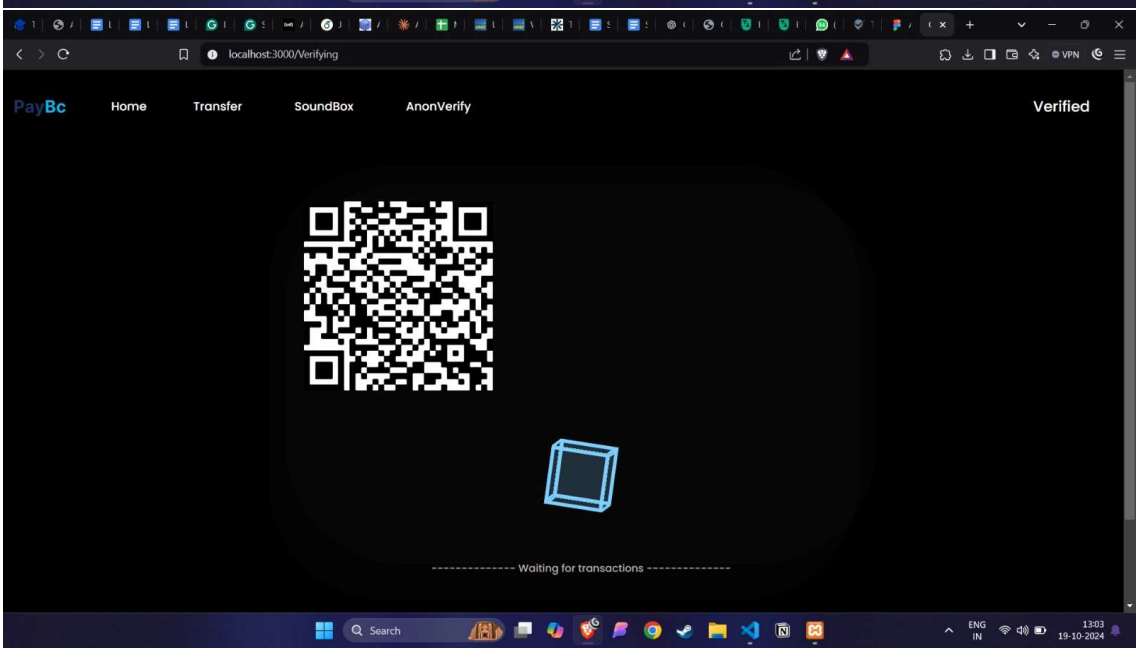

localhost:3000/anon-verify
PayBc
Home
Docs
Verify
Anon Aadhaar allows you to create a proof of your Aadhaar ID without revealing any personal data. This process is local to your browser for privacy, and QR images are not uploaded to any server.
Note: Internet speed may affect processing time.
GENERATE A QR CODE:
You can try this example app by generating a test Adhaar QR Code
Link to generate a QR Code
UPLOAD YOUR AADHAAR SECURE QR CODE:
Choose file
CONTINUE
No Aadhaar data ever leaves your device!
Aadhaar Verification
Signature Verification
Age Verification

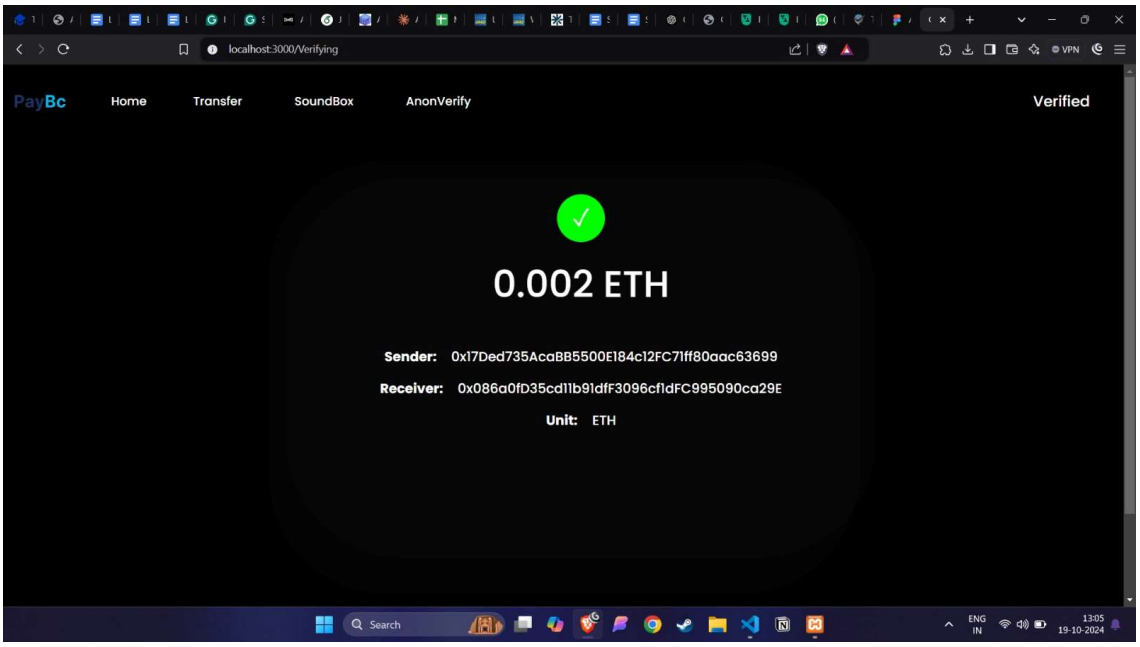

localhost:3000/anon-verify
PayBc
Home
Docs
Verify
Anon Aadhaar allows you to create a proof of your Aadhaar ID without revealing any personal data. This process is local to your browser for privacy, and QR images are not uploaded to any server.
Note: Internet speed may affect processing time.
GENERATE A QR CODE:
You can try this example app by generating a test Adhaar QR Code
Link to generate a QR Code
UPLOAD YOUR AADHAAR SECURE QR CODE:
Choose file
realanon.png
Valid QR Code.
CONTINUE
No Aadhaar data ever leaves your device!
Aadhaar Verification
Signature Verification
Age Verification


localhost:3000/anon-verify
PayBc
Home
Docs
Verify
YOUR QR CODE IS VERIFIED!
Data you are sharing to The current application:
AGE ABOVE 18
GENDER
PIN CODE
STATE
Data you are signing:
0x086a0fd35cd11b91dff3096cf1dfc995090ca29e
GENERATE ANON AADHAAR PROOF
No Aadhaar data ever leaves your device!
Aadhaar Verification
Signature Verification
Age Verification


localhost:3000/anon-verify
PayBc
Home
Docs
Verify
Top left icon
Top right icon
Bottom left icon
Bottom right icon
GENERATING PROOF...
Proof generation on a MacBook Pro M1 - 16GB is about 26 seconds
Aadhaar Verification
Signature Verification
Age Verification


localhost:3000/anon-verify
PayBc
Home
Transfer
SoundBox
AnonVerify
Verified
Verify your aadhar card
What you want as to use
ageAbove18
gender
pinCode
state
Use the QR of your Aadhar card
Menu
Status: logged-in
ANON-VERIFY
Aadhaar Verification
Signature Verification
Age Verification


localhost:3000/Verifying
PayBc
Home
Transfer
SoundBox
AnonVerify
Verified
---------- Waiting for transactions ----------


localhost:3000/Verifying
PayBc
Home
Transfer
SoundBox
AnonVerify
Verified
0.002 ETH
Sender: 0x17Ded735AcaBB5500E184c12FC71ff80aac63699
Receiver: 0x086a0fD35cd11b91dfF3096cf1dFC995090ca29E
Unit: ETH
Search
ENG
IN
13:05
19-10-2024

Arnav Malvia
C-32
2103109
BlockChain
Date :- 13/7/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment :- 1

Q1 What are the problems associated with a centralized system?

Ans1 Centralised System-
In a centralized system, a single entity or a small group of entities controls the entire network. This central authority is responsible for validating transactions, maintaing the ledger and ensuring the security of the network. Traditional financial systems and mainly online service operate in a centralized manner.

Problems associated with centralised systems-

1. Single Point of failure:
In a centralized system, all operations and data are managed by a single point of control. If this central server or authority fails or is compromised, the entire system can collapse.
example- If a bank's main server goes down due to the technical issue or a cyber attack all the services dependent on the server would be disrupted, potentially causing significant financial and operational damage.

2. Lack of Transparency-
Centralised systems often do not provide users with full visibility into the operations and data handling processes. This opacity can lead to mistrust among the users and stakeholders.
example - In traditional banking, customers cannot see how their

transaction are processed internally. They must trust the bank to handle their money correctly.

3. Security Risks -
Centralised systems are attractive targets for hackers because compromising the central authority can grant access to the entire network.
example - A successful hack on a centralised cryptocurrency exchange could result in the theft of all users funds stored on that platform.

4. Censorship -
The central authority has the power to control and censor transactions or information on the network.
example - A government controlled financial system might censor transactions it sees politially undesirable, restricting citizens financial freedom.

5. High Costs -
Centralised systems often require significant investments in infrastructure security and maintainence to ensure reliability and performance.
example - Banks spend large amounts ~~of~~ on secure data centers, cybersecurity measures and compliance with regulatory requirements which can drive up operational costs.

6. Limited Access and Control -
Users have limited control over their data and transactions as the central authority retains ultimate control.
example - Users of a social media platform must adhere

to the terms of service set by the platfoorm and the platfoorm can unilaterally deide to suspend or delete accounts.

7. Inefficiency-
As centralised systems scale they can become bottlenecks due to the concentration of processing power and decision making in one place.
example- A centralised payment processing system may experience delays during peak transaction periods, affecting the user experience.

8. Trust issues-
Users must trust the central authority to act in their best interests which can be challenging if there are concerns about the authority's integrity or competence.
example- If a centralised exchange is sucessful suspected of manipulating market prices, users may lose trust and migrate to other platforms.

Q2 Difference between Centralized, Decentralised and Distributed Systems.

Ans 2

Aspect	Centralised Systems	Decentralised Systems	Distributed Systems
1) Define	Single central server controls and manages all operations	Multiple nodes with independent control no central authority.	Multiple interconnected nodes working together as a single system.
2) Control	Centralised control with a single point of management.	Distributed control, each node operates independently.	Shared control, nodes collaborate to achieve common goals.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

3) Single point of failure	High risk if central server fails, the whole system fails.	Redured risk, failure of one node does not impact entire system.	Reduced risk, designed for fault tolerance and redundancy.
4) Scalability	Limited Scalability, can become bottle neck.	More scalable, can add nodes independently	Highly scalable, can add more nodes to distribute the load.
5) Resource Utilization	Control central server resources are heavily utilized.	Resources are spread across multiple nodes	Efficient resource sharing across nodes.
6) Performance	Can be high initially but may degrade with increased load.	Generally good, performance improves with more nodes.	High performance due to parallel processing and resource sharing
7) Management	Easier to manage centrally.	More complex, requires managing multiple nodes.	Complex, requires coordination and management of many nodes.
8) Latency	Lower latency as operations are managed centrally.	Can vary, depends on the distance between the nodes	Potentially higher latency due to network communication.

16/7/24 A+

Aarav Malvia
C-32
2103109
Blockchain
Date :- 8/8/24

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment :- 2

Q1 Explain transactions in blockchain and UTXO and Double spending.

Ans 1 Transactions in blockchain:-

1) They represent the movement of cryptocurrency from one address to another address.
2) A transaction typically includes the amount being sent, the sender's ~~addre~~ digital signature to prove the ownership.

UTXO -

1) Unspent Transaction Output
2) It is a concept used in certain blockchain architectures like bitcoin.
3) It represents the "change" for "balance" of cryptocurrency owned by an address that has not been spent yet.
4) This system increases privacy and the efficiency.

Double Spending:-

1) It is a potential issue in digital currencies where a user attempts to spend the same cryptocurrency more than once.
2) It occurs when someone tries to send the same cryptocurrency to two different ~~rece~~ recipients simultaneously.
3) Blockchain technology prevents the double spending by maintaining a distributed ledger of all the transactions, making it difficult.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Q2 Explain in detail CORDA, Ripple and Quorum.

Ans 2 CORDA :-

CORDA is an open source blockchain platform developed by the R3, and designed for use in enterprise environments, especially within the financially centred industry.

Primary focus is on enabling the private and the secure transactions and the contracts between different parties while ensuring regulatory compliance.

Ripple :-

It is both a blockchain platform and a digital payment protocol. Its designed for the real time, cross border payments and settlement.

Ripple's focus is on enabling the fast, cost efficient, the international money transfers.

Quorum :-

It is an enterprise focused, permissioned blockchain platform which is developed by JP Morgan Chase and Co.

It is built on top of ethereum but includes the modifications and the features tailored for the private permissioned use case.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Q3 Explain 0, 1 and 6 confirmation transaction.

Ans 3 0 transaction, 1 transaction and 6 transaction refer to the number of total blocks in the blockchain that have been added after a transaction is broadcasted.

~~0 transaction~~

0-confirmation transaction :-
This means the transactions have been broadcasted to the network but has not yet been included in any block.

1-confirmation transaction :-
When a transaction either receives a 1-confirmation transaction it means that it has been included in the latest block added to the blockchain.
It provides a basic level of the security as it indicates that at least one of the miner has validated and added the transaction to the blockchain.

6-confirmation transaction :-
It is often considered as the gold standard for the transaction security in the crypto. When a transaction has received six confirmations, it means that it has been included in the six consecutive blocks added to the blockchain.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q4 What is smart contract? How crowd funding platforms can be managed using smart contracts?

Ans 4 A smart contract is a self executing contract with the terms of the agreement between the buyer and the seller directly into the lines of code. They automatically execute and enforce themselves when predefined conditions or triggers are met, without the need for any third party intermediaries.

Crowdfunding platforms can be managed using the smart contracts:-

1) Fund raising campaigns - They can be used to create the fund raising campaigns on crowd funding platforms.
2) Contributions - Contributers can send their funds directly to the smart contract address.
3) Transparency
4) Trust
5) Automatic Execution

10/9/24

A+