A1 – 1 : Program to display the Fibonacci series up to n-th term without recursion

---

Code :

```
print("Program to display the Fibonacci series up to n-th term without recursion")

nterms = int(input("Enter number of terms : "))

n1, n2 = 0, 1

count = 0

if nterms <= 0:

   print("Please enter a positive integer")

elif nterms == 1:

   print("Fibonacci series upto", nterms, ":")

   print(n1)

else:

   print("Fibonacci series:")

   while count < nterms:

      print(n1)

      nth = n1 + n2

     # update values

      n1 = n2

      n2 = nth

      count += 1
```

Output :

Program to display the Fibonacci series up to n-th term without recursion

Enter number of terms : 5

Fibonacci series:

0

1

1

2

3

Program to display the Fibonacci series up to n-th term without recursion

Enter number of terms : 7

Fibonacci series:

0

1

1

2

3

5

8

A1 – 2 : Program to display the Fibonacci series up to n-th term using recursion

---

Code :

```
print("Program to display the Fibonacci series up to n-th term with recursion")

def fibonacci(n):

   if(n <= 1):

      return n

   else:

      return(fibonacci(n-1) + fibonacci(n-2))

n = int(input("Enter number of terms : "))

print("Fibonacci series : ")

for i in range(n):

   print(fibonacci(i))
```

Output :

```
Program to display the Fibonacci series up to n-th term with recursion

Enter number of terms : 4

Fibonacci series :

0

1

1

2
```

Program to display the Fibonacci series up to n-th term with recursion

Enter number of terms : 9

Fibonacci series :

0

1

1

2

3

5

8

13

21

A2 : Program to implement Huffman Encoding using a greedy strategy.

Code :

```
import heapq

class node:
	def __init__(self, freq, symbol, left=None, right=None):
		# frequency of symbol
		self.freq = freq
		# symbol name (character)
		self.symbol = symbol
		# node left of current node
		self.left = left
		# node right of current node
		self.right = right
		# tree direction (0/1)
		self.huff = ''
	def __lt__(self, nxt):
		return self.freq < nxt.freq
# utility function to print huffman
# codes for all symbols in the newly
# created Huffman tree
def printNodes(node, val=''):
	# huffman code for current node
```

```
        newVal = val + str(node.huff)
        # if node is not an edge node
        # then traverse inside it
        if(node.left):
                printNodes(node.left, newVal)
        if(node.right):
                printNodes(node.right, newVal)
                # if node is edge node then
                # display its huffman code
        if(not node.left and not node.right):
                print(f"{node.symbol} -> {newVal}")
# characters for huffman tree
chars = ['a', 'b', 'c', 'd', 'e', 'f']
# frequency of characters
freq = [ 5, 9, 12, 13, 16, 45]
# list containing unused nodes
nodes = []
# converting characters and frequencies
# into huffman tree nodes
for x in range(len(chars)):
        heapq.heappush(nodes, node(freq[x], chars[x]))
while len(nodes) > 1:
        # sort all the nodes in ascending order
```

```
        # based on their frequency
        left = heapq.heappop(nodes)
        right = heapq.heappop(nodes)
        # assign directional value to these nodes
        left.huff = 0
        right.huff = 1
        # combine the 2 smallest nodes to create
        # new node as their parent
        newNode = node(left.freq+right.freq, left.symbol+right.symbol, left, right)
        heapq.heappush(nodes, newNode)
# Huffman Tree is ready!
print("Characters :", chars)
print("Frequency  :", freq, "\n\nHuffman Encoding:")
printNodes(nodes[0])
```

Output :

```
Characters : ['a', 'b', 'c', 'd', 'e', 'f']
Frequency  : [5, 9, 12, 13, 16, 45]

Huffman Encoding:
f -> 0
c -> 100
```

d -> 101

a -> 1100

b -> 1101

e -> 111

A3 : Program to solve a fractional Knapsack problem using a greedy method.

Code :

```
class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight
def fractionalKnapsack(W, arr):
    # Sorting Item on basis of ratio
    arr.sort(key=lambda x: (x.value/x.weight), reverse=True)
    # Result(value in Knapsack)
    finalvalue = 0.0
    # Looping through all Items
    for item in arr:
        # If adding Item won't overflow,
        # add it completely
        if item.weight <= W:
            W -= item.weight
            finalvalue += item.value
            # If we can't add current Item,
            # add fractional part of it
        else:
            finalvalue += item.value * W / item.weight
```

```
            break

    # Returning final value
    return finalvalue


if __name__ == "__main__":
    W = 50
    arr = [Item(60, 10), Item(100, 20), Item(120, 30)]
    # Function call
    max_val = fractionalKnapsack(W, arr)
    print("Maximum value in Knapsack =",max_val)
```

Output :

```
Maximum value in Knapsack = 240.0
```

A4 : Program to solve a 0-1 Knapsack problem using dynamic programming.

Code :

```
def knapsack_dp(W, wt, val, n):
    """A Dynamic Programming based solution for 0-1 Knapsack problem
    Returns the maximum value that can"""
    K = [[0 for x in range(W + 1)] for x in range(n + 1)]
    # Build table K[][] in bottom up manner
    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif wt[i - 1] <= w:
                K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w])
            else:
                K[i][w] = K[i - 1][w]
    return K[n][W]
val = [60, 100, 120]
wt = [10, 20, 30]
W = 50
n = len(val)
print("Maximum possible profit =", knapsack_dp(W, wt, val, n))
```

Output :

```
Maximum possible profit = 220
```

A5 : Program to solve N Queen Problem using backtracking

Code :

```
N = 4  # Global variable for board size

def print_solution(board):
    """Print the board solution."""
    for row in board:
        print(' '.join(map(str, row)))
    print()

def is_safe(board, row, col):
    """Check if it's safe to place a queen at board[row][col]."""
    # Check this row on the left side
    if 1 in board[row][:col]:
        return False
    # Check upper diagonal on the left side
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    # Check lower diagonal on the left side
    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    return True
```

```
def solve_nq_util(board, col):
    """Recursive utility function to solve N Queen problem."""
    # Base case: If all queens are placed, return True
    if col >= N:
        return True
    # Consider this column and try placing this queen in all rows one by one
    for i in range(N):
        if is_safe(board, i, col):
            # Place this queen in board[i][col]
            board[i][col] = 1
            # Recur to place rest of the queens
            if solve_nq_util(board, col + 1):
                return True
            # If placing queen in board[i][col] doesn't lead to a solution,
            # then remove queen from board[i][col]
            board[i][col] = 0
    # If the queen can't be placed in any row in this column col, return False
    return False
def solve_nq():
    """
    Solve the N Queen problem using Backtracking.
    This function solves the N Queen problem using Backtracking.
    It returns False if queens cannot be placed, otherwise returns True
```

```
    and prints the placement of queens in the form of 1s.

    Note that there may be more than one solution, this function prints one of the feasible
solutions.

    """

    board = [[0 for _ in range(N)] for _ in range(N)]

    if not solve_nq_util(board, 0):

        print("Solution does not exist")

        return False

    print_solution(board)

    return True

if __name__ == "__main__":

    print("Placed Queens where N =",N)

    solve_nq()
```

Output :

```
Placed Queens where N = 4

0 0 1 0

1 0 0 0

0 0 0 1

0 1 0 0
```

C3 : Write a smart contract on a test network for Bank account of a customer for following operations.

- Deposit Money
- Withdraw Money
- Show Balance

Code :

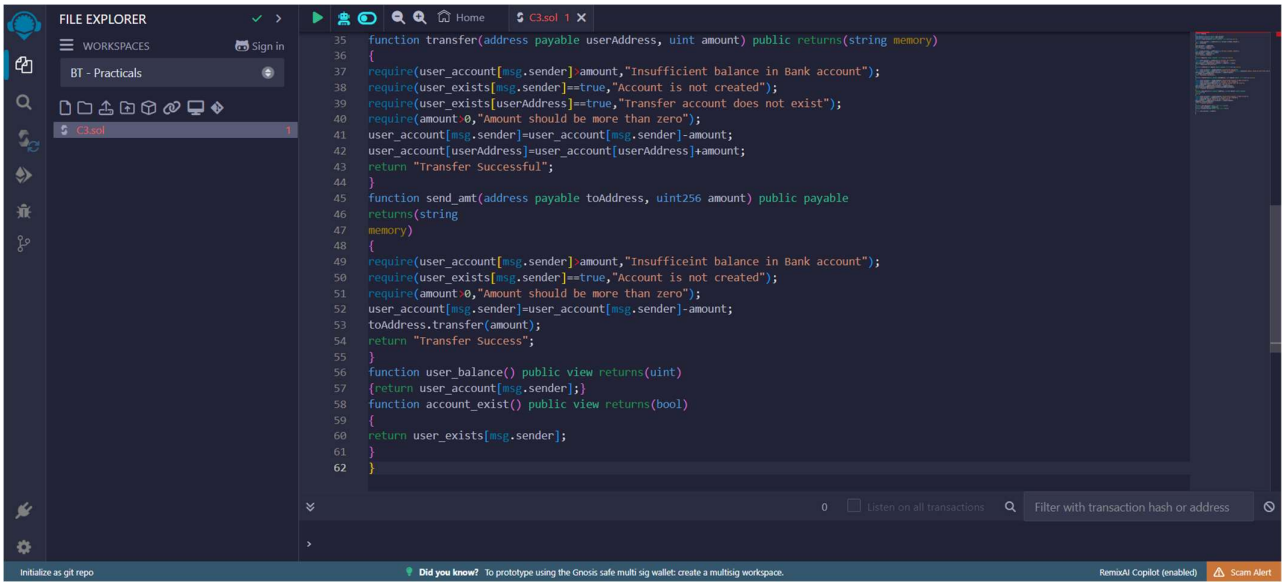
```
pragma solidity ^0.6;

contract banking

{

mapping(address=>uint) public user_account;

mapping(address=>bool) public user_exists;

function create_account() public payable returns(string memory)

{

require(user_exists[msg.sender]==false,'Account already created');

if(msg.value==0)

{

user_account[msg.sender]=0;

user_exists[msg.sender]=true;

return "Account created";

}

require(user_exists[msg.sender]==false,"Account already created");

user_account[msg.sender]=msg.value;
```

```
user_exists[msg.sender]=true;
return "Account created";
}
function deposit() public payable returns(string memory)
{
require(user_exists[msg.sender]==true,"Account not created");
require(msg.value>0,"Value for deposit is Zero");
user_account[msg.sender]=user_account[msg.sender]+msg.value;
return "Deposited Successfully";
}
function withdraw(uint amount) public payable returns(string memory)
{
require(user_account[msg.sender]>amount,"Insufficient Balance");
require(user_exists[msg.sender]==true,"Account not created");require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
msg.sender.transfer(amount);
return "Withdrawl Successful";
}
function transfer(address payable userAddress, uint amount) public returns(string memory)
{
require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(user_exists[userAddress]==true,"Transfer account does not exist");
```

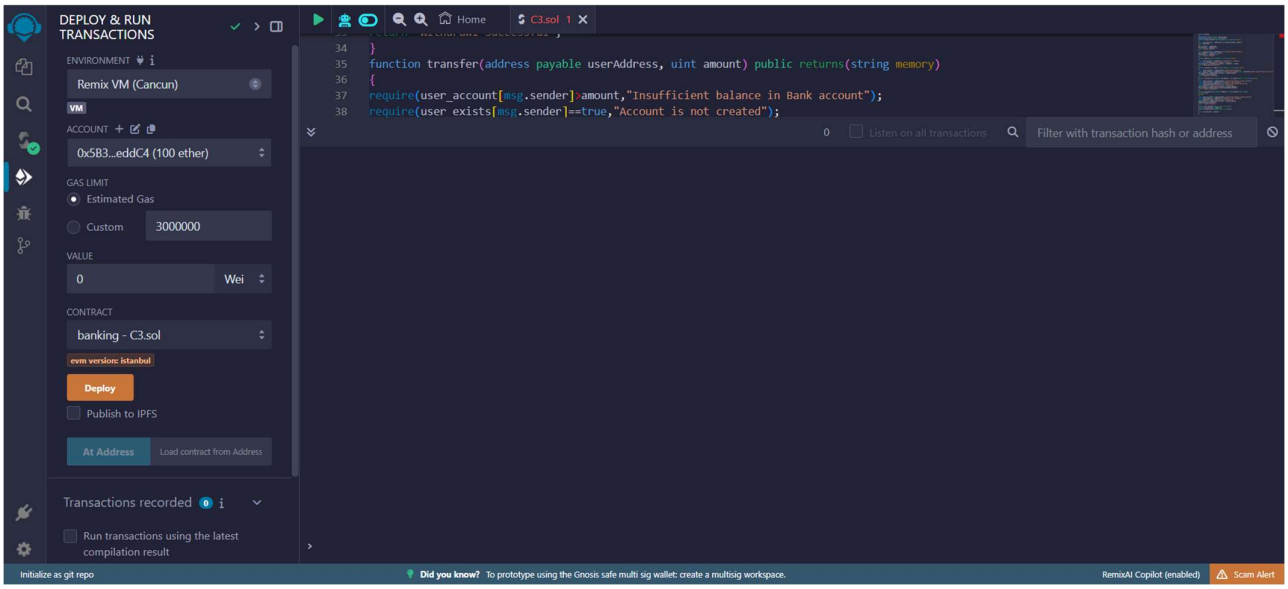
```
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
user_account[userAddress]=user_account[userAddress]+amount;
return "Transfer Successful";
}
function send_amt(address payable toAddress, uint256 amount) public payable
returns(string
memory)
{
require(user_account[msg.sender]>amount,"Insufficeint balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
toAddress.transfer(amount);
return "Transfer Success";
}
function user_balance() public view returns(uint)
{return user_account[msg.sender];}
function account_exist() public view returns(bool)
{
return user_exists[msg.sender];
}
}
```

Output :

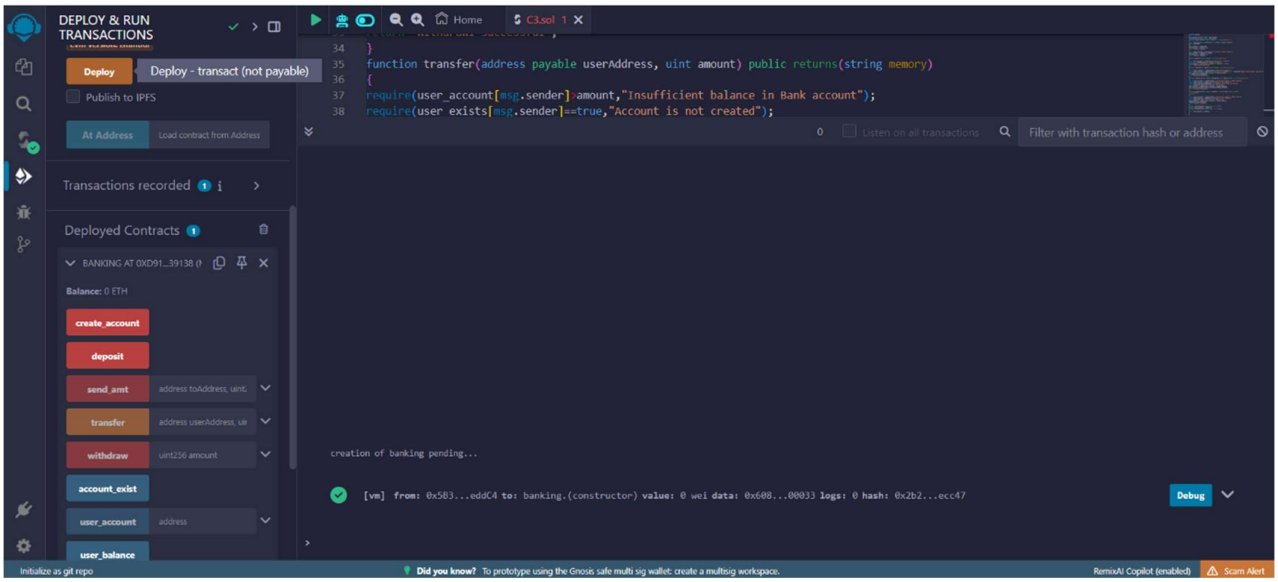
1 >


FILE EXPLORER
WORKSPACES
Sign in
BT - Practicals
C3.sol
Home
C3.sol 1
function transfer(address payable userAddress, uint amount) public returns(string memory)
{
require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(user_exists[userAddress]==true,"Transfer account does not exist");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
user_account[userAddress]=user_account[userAddress]+amount;
return "Transfer Successful";
}
function send_amt(address payable toAddress, uint256 amount) public payable
returns(string
memory)
{
require(user_account[msg.sender]>amount,"Insufficeint balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
toAddress.transfer(amount);
return "Transfer Success";
}
function user_balance() public view returns(uint)
{return user_account[msg.sender];}
function account_exist() public view returns(bool)
{
return user_exists[msg.sender];
}
}
0
Listen on all transactions
Filter with transaction hash or address
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

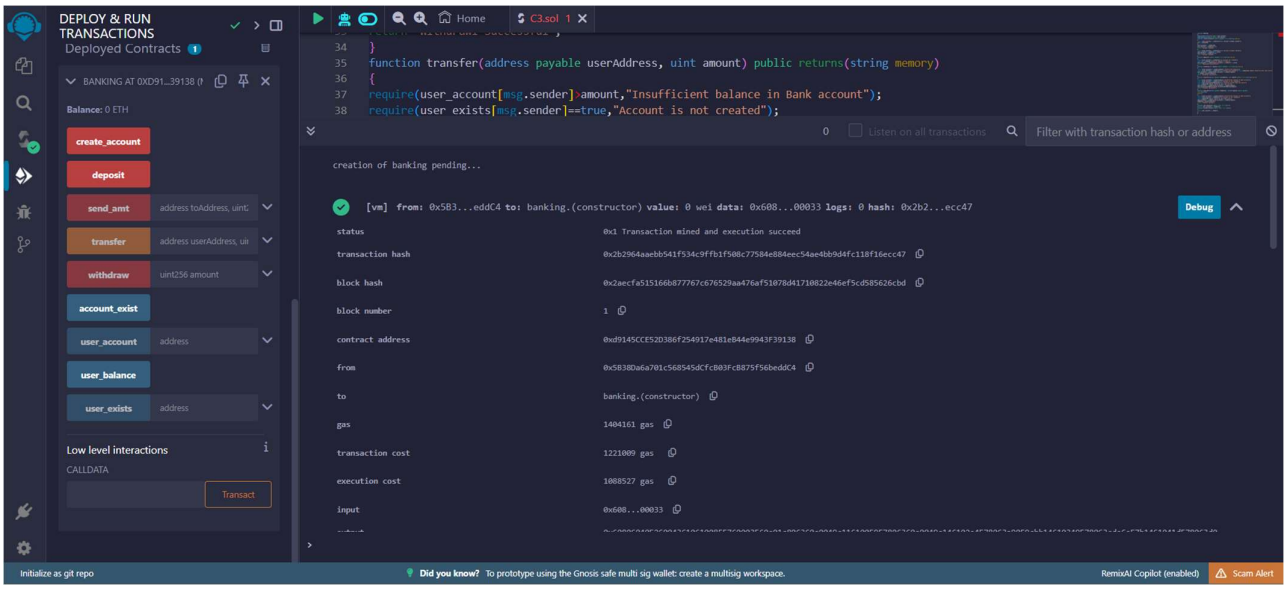
2 >


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (100 ether)
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
banking - C3.sol
evm version: istanbul
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Run transactions using the latest compilation result
Home
C3.sol 1
}
function transfer(address payable userAddress, uint amount) public returns(string memory)
{
require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
require(user exists[msg.sender]==true,"Account is not created");
0
Listen on all transactions
Filter with transaction hash or address
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

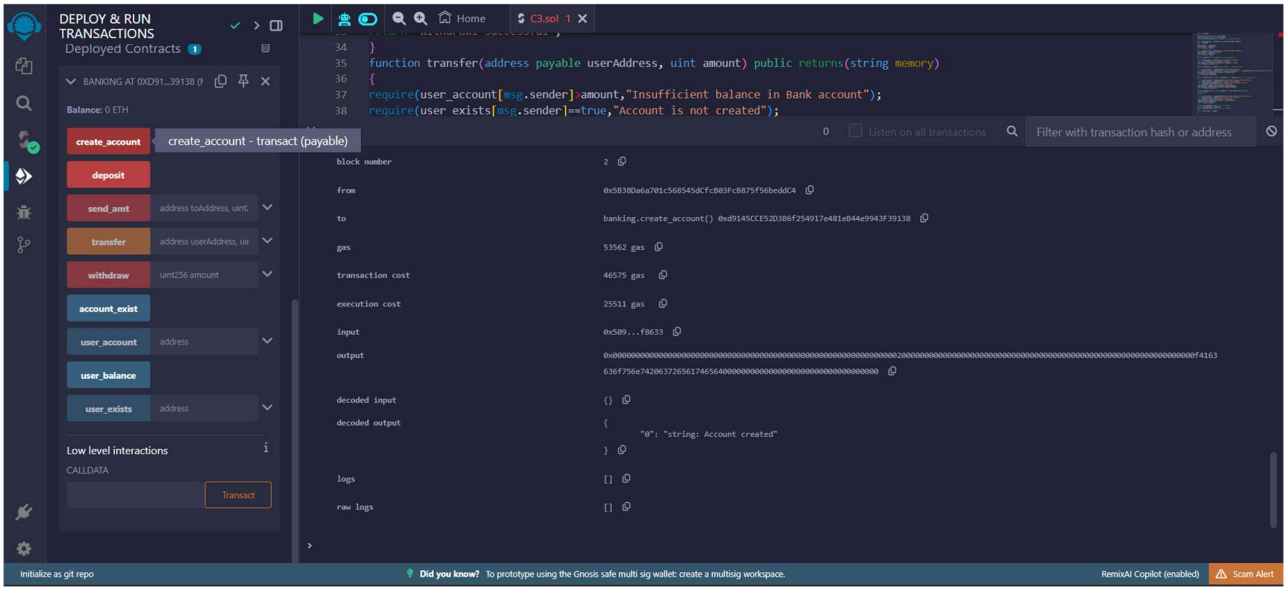
3 >


DEPLOY & RUN TRANSACTIONS
Deploy
Deploy - transact (not payable)
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
BANKING AT 0XD91...39138 (M
Balance: 0 ETH
create_account
deposit
send_amt
address toAddress, uint
transfer
address userAddress, ui
withdraw
uint256 amount
account_exist
user_account
address
user_balance
Home
C3.sol
34 }
35 function transfer(address payable userAddress, uint amount) public returns(string memory)
36 {
37 require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
38 require(user exists[msg.sender]==true,"Account is not created");
0
Listen on all transactions
Filter with transaction hash or address
creation of banking pending...
[vm] from: 0x5B3...eddC4 to: banking.(constructor) value: 0 wei data: 0x608...00033 logs: 0 hash: 0x2b2...ecc47
Debug
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

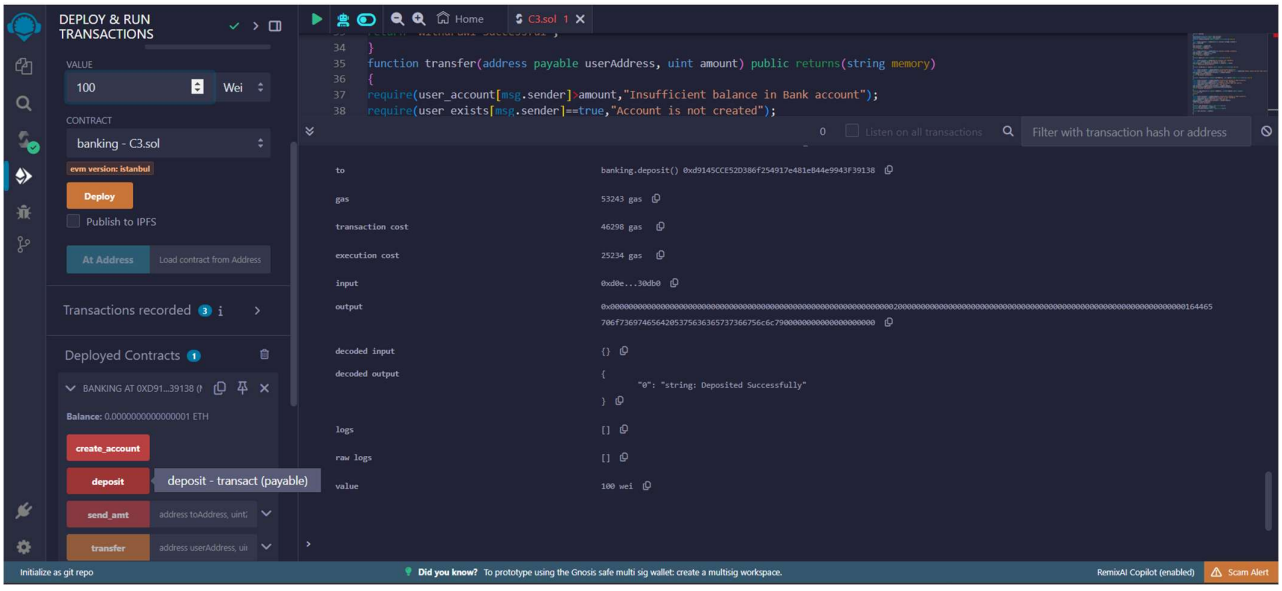
4 >


DEPLOY & RUN TRANSACTIONS
Deployed Contracts
BANKING AT 0XD91...39138 (M
Balance: 0 ETH
create_account
deposit
send_amt
address toAddress, uint
transfer
address userAddress, ui
withdraw
uint256 amount
account_exist
user_account
address
user_balance
user_exists
address
Low level interactions
CALLDATA
Transact
Home
C3.sol
34 }
35 function transfer(address payable userAddress, uint amount) public returns(string memory)
36 {
37 require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
38 require(user exists[msg.sender]==true,"Account is not created");
0
Listen on all transactions
Filter with transaction hash or address
creation of banking pending...
[vm] from: 0x5B3...eddC4 to: banking.(constructor) value: 0 wei data: 0x608...00033 logs: 0 hash: 0x2b2...ecc47
Debug
status
0x1 Transaction mined and execution succeed
transaction hash
0x2b2964aaebb541f534c9ffb1f508c77584e884eec54ae4bb9d4fc118f16ecc47
block hash
0x2aecfa515166b877767c676529aa476af51078d41710822e46ef5cd585626cbd
block number
1
contract address
0xd9145CCE52D386f254917e481eB44e9943F39138
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
banking.(constructor)
gas
1404161 gas
transaction cost
1221009 gas
execution cost
1088527 gas
input
0x608...00033
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

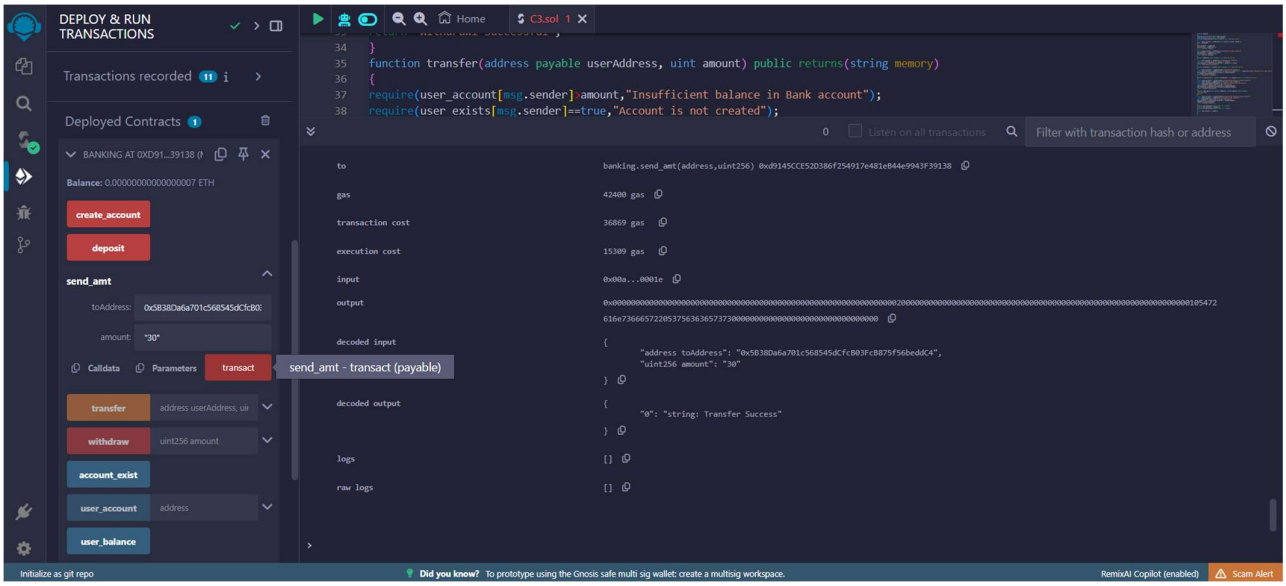
5 >


DEPLOY & RUN TRANSACTIONS
Deployed Contracts
BANKING AT 0XD91...39138
Balance: 0 ETH
create_account
create_account - transact (payable)
deposit
send_amt
transfer
withdraw
account_exist
user_account
user_balance
user_exists
Low level interactions
CALLDATA
Transact
Home
C3.sol
Filter with transaction hash or address
block number 2
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to banking.create_account() 0xd9145CCE52D386f254917e481eB44e9943F39138
gas 53562 gas
transaction cost 46575 gas
execution cost 25511 gas
input 0x509...f8633
decoded input {}
decoded output { "0": "string: Account created" }
logs []
raw logs []
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

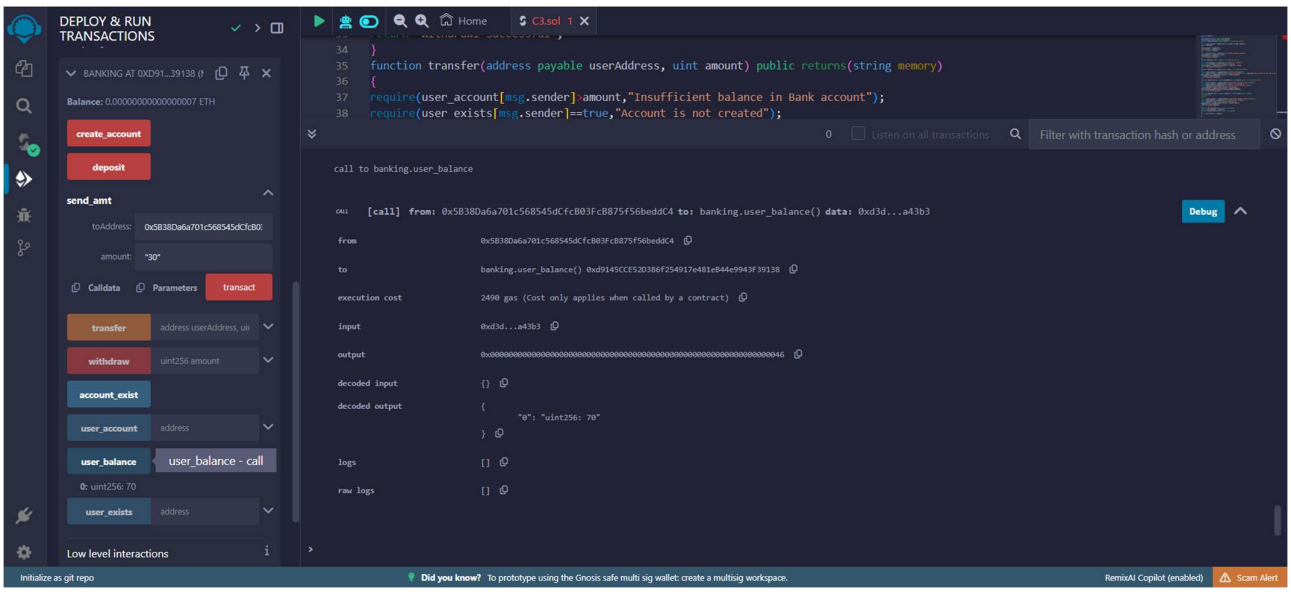
6 >


DEPLOY & RUN TRANSACTIONS
VALUE
100
Wei
CONTRACT
banking - C3.sol
evm version: istanbul
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 3
Deployed Contracts
BANKING AT 0XD91...39138
Balance: 0.000000000000001 ETH
create_account
deposit
deposit - transact (payable)
send_amt
transfer
Home
C3.sol
Filter with transaction hash or address
to banking.deposit() 0xd9145CCE52D386f254917e481eB44e9943F39138
gas 53243 gas
transaction cost 46298 gas
execution cost 25234 gas
input 0xd0e...30db0
decoded input {}
decoded output { "0": "string: Deposited Successfully" }
logs []
raw logs []
value 100 wei
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

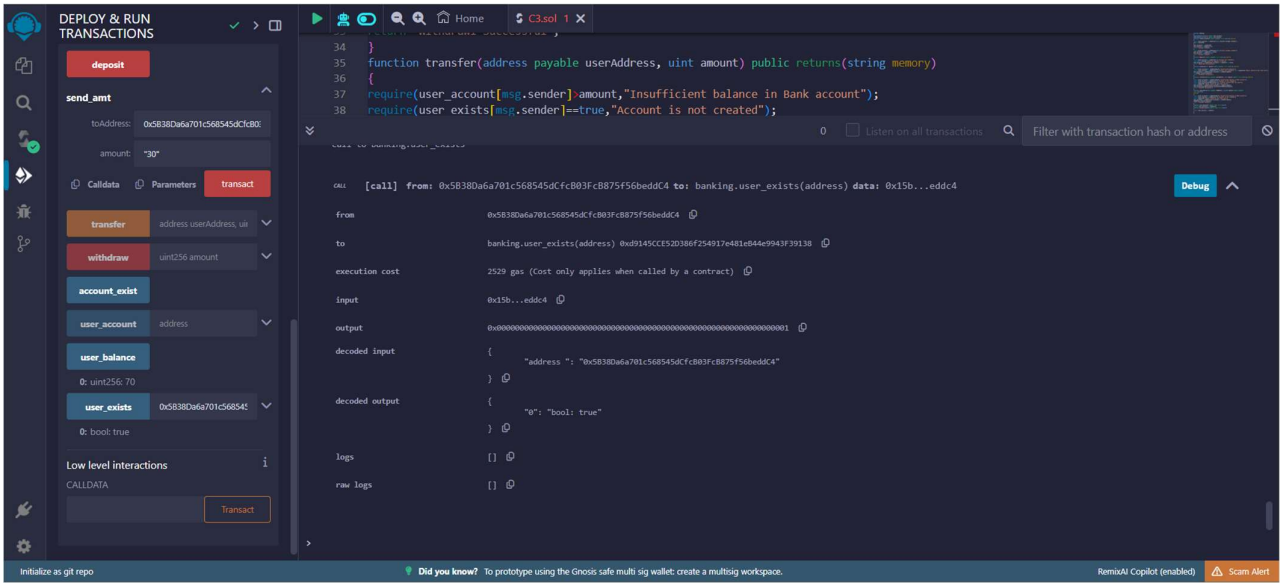
7 >


DEPLOY & RUN TRANSACTIONS
Transactions recorded 11
Deployed Contracts 1
BANKING AT 0XD91...39138
Balance: 0.000000000000000007 ETH
create_account
deposit
send_amt
toAddress: 0x5B38Da6a701c568545dCfcB03
amount: "30"
Calldata
Parameters
transact
send_amt - transact (payable)
transfer
address userAddress, uin
withdraw
uint256 amount
account_exist
user_account
address
user_balance
Home
C3.sol 1
34 }
35 function transfer(address payable userAddress, uint amount) public returns(string memory)
36 {
37 require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
38 require(user exists[msg.sender]==true,"Account is not created");
0
Listen on all transactions
Filter with transaction hash or address
to
banking.send_amt(address,uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas
42400 gas
transaction cost
36869 gas
execution cost
15309 gas
input
0x00a...0001e
output
0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000105472
616e73666572205375636365737300000000000000000000000000000000
decoded input
{
"address toAddress": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
"uint256 amount": "30"
}
decoded output
{
"0": "string: Transfer Success"
}
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

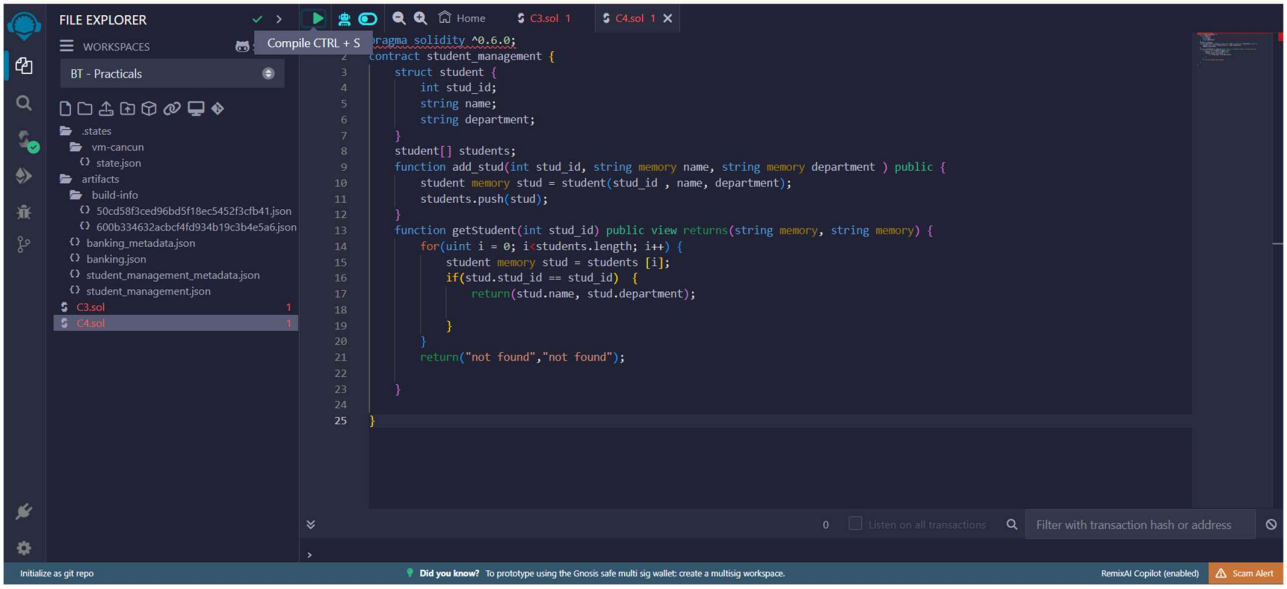
8 >


DEPLOY & RUN TRANSACTIONS
BANKING AT 0XD91...39138
Balance: 0.000000000000000007 ETH
create_account
deposit
send_amt
toAddress: 0x5B38Da6a701c568545dCfcB03
amount: "30"
Calldata
Parameters
transact
transfer
address userAddress, uin
withdraw
uint256 amount
account_exist
user_account
address
user_balance
user_balance - call
0: uint256: 70
user_exists
address
Low level interactions
Home
C3.sol 1
34 }
35 function transfer(address payable userAddress, uint amount) public returns(string memory)
36 {
37 require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
38 require(user exists[msg.sender]==true,"Account is not created");
0
Listen on all transactions
Filter with transaction hash or address
call to banking.user_balance
call [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: banking.user_balance() data: 0xd3d...a43b3
Debug
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
banking.user_balance() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost
2490 gas (Cost only applies when called by a contract)
input
0xd3d...a43b3
output
0x0000000000000000000000000000000000000000000000000000000000000046
decoded input
{}
decoded output
{
"0": "uint256: 70"
}
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

9 >


DEPLOY & RUN TRANSACTIONS
deposit
send_amt
toAddress: 0x5B38Da6a701c568545dCfcB0
amount: "30"
Calldata
Parameters
transact
transfer
address userAddress, uir
withdraw
uint256 amount
account_exist
user_account
address
user_balance
0: uint256: 70
user_exists
0x5B38Da6a701c56854
0: bool: true
Low level interactions
CALLDATA
Transact
Home
C3.sol 1
34 }
35 function transfer(address payable userAddress, uint amount) public returns(string memory)
36 {
37 require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
38 require(user exists[msg.sender]==true,"Account is not created");
0
Listen on all transactions
Filter with transaction hash or address
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: banking.user_exists(address) data: 0x15b...eddc4
Debug
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to banking.user_exists(address) 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost 2529 gas (Cost only applies when called by a contract)
input 0x15b...eddc4
output 0x0000000000000000000000000000000000000000000000000000000000000001
decoded input { "address ": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4" }
decoded output { "0": "bool: true" }
logs []
raw logs []
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

10 >


FILE EXPLORER
WORKSPACES
Compile CTRL + S
BT - Practicals
.states
vm-cancun
state.json
artifacts
build-info
50cd58f3ced96bd5f18ec5452f3cfb41.json
600b334632acbcf4fd934b19c3b4e5a6.json
banking_metadata.json
banking.json
student_management_metadata.json
student_management.json
C3.sol 1
C4.sol 1
Home
C3.sol 1
C4.sol 1
pragma solidity ^0.6.0;
contract student_management {
struct student {
int stud_id;
string name;
string department;
}
student[] students;
function add_stud(int stud_id, string memory name, string memory department) public {
student memory stud = student(stud_id , name, department);
students.push(stud);
}
function getStudent(int stud_id) public view returns(string memory, string memory) {
for(uint i = 0; i<students.length; i++) {
student memory stud = students [i];
if(stud.stud_id == stud_id) {
return(stud.name, stud.department);
}
}
return("not found","not found");
}
}
0
Listen on all transactions
Filter with transaction hash or address
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

C4 : Write a program in solidity to create Student data. Use the following constructs:

- Structures
- Arrays
- Fallback

Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas value.

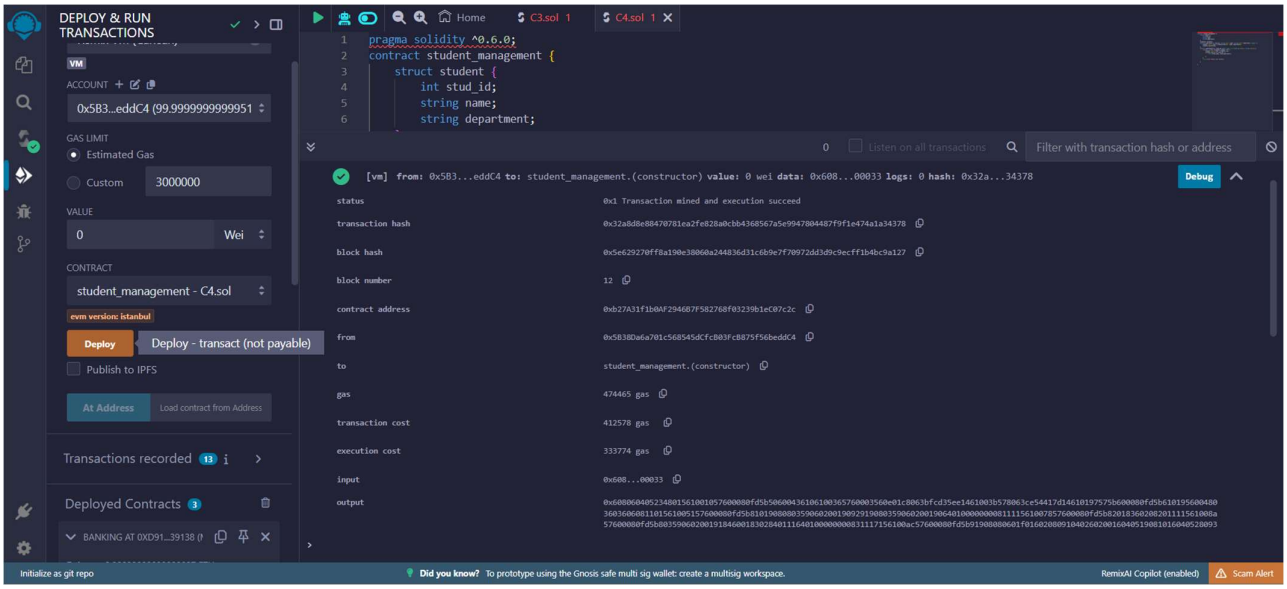
Code :

```
pragma solidity ^0.6.0;
contract student_management {
   struct student {
      int stud_id;
      string name;
      string department;
   }
   student[] students;
   function add_stud(int stud_id, string memory name, string memory department ) public {
      student memory stud = student(stud_id , name, department);
      students.push(stud);
   }
   function getStudent(int stud_id) public view returns(string memory, string memory) {
      for(uint i = 0; i<students.length; i++) {
         student memory stud = students [i];
         if(stud.stud_id == stud_id)  {
            return(stud.name, stud.department);
         }
```

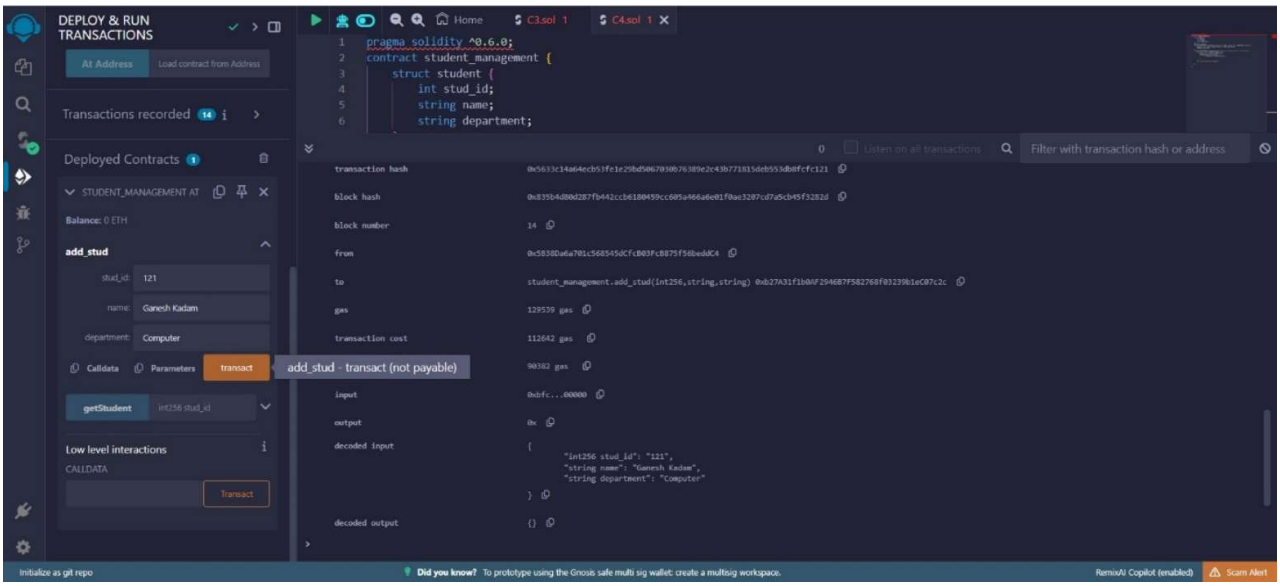
```
    }
    return("not found","not found");
  }
}
```

Output :

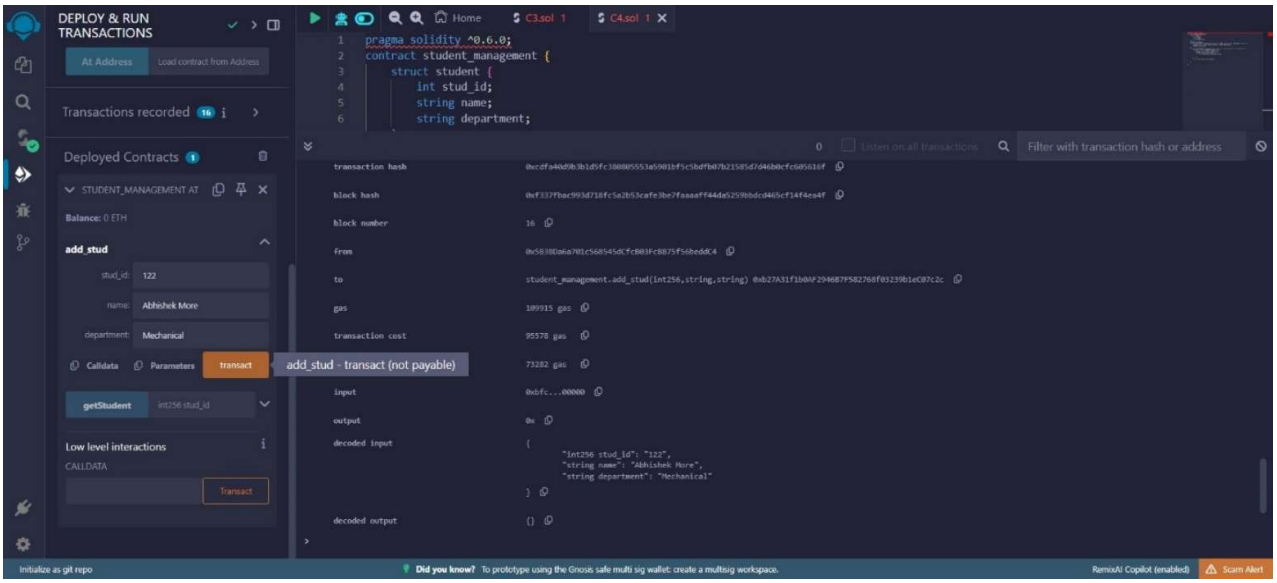
11 >



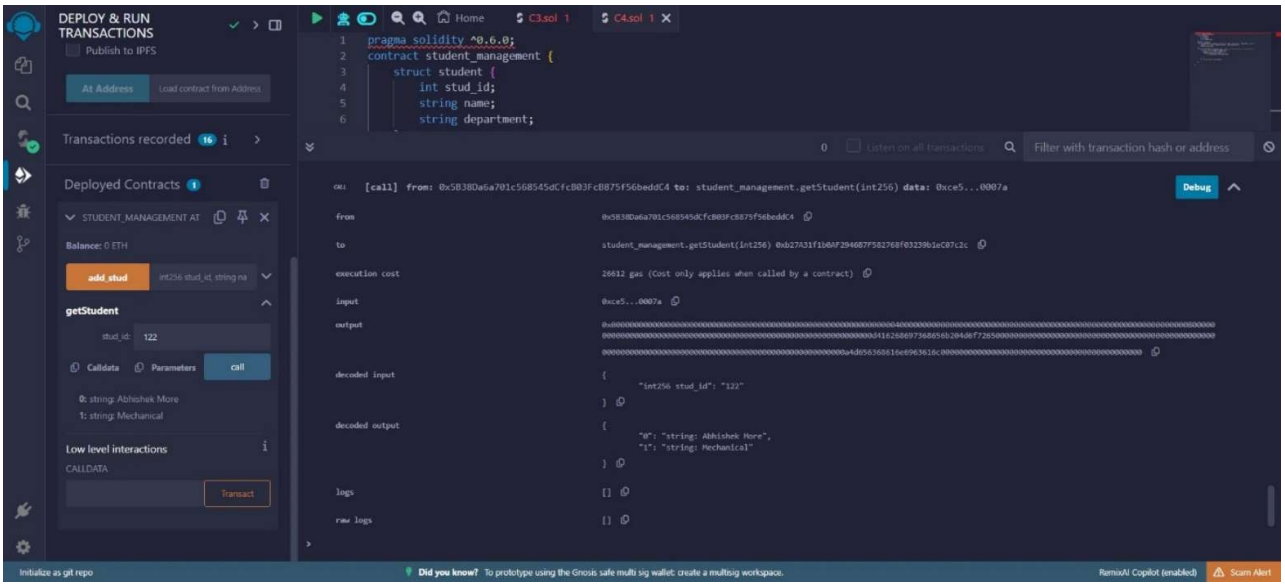
12>


DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 14
Deployed Contracts
STUDENT_MANAGEMENT AT
Balance: 0 ETH
add_stud
stud_id: 121
name: Ganesh Kadam
department: Computer
Calldata
Parameters
transact
add_stud - transact (not payable)
getStudent
int256 stud_id
Low level interactions
CALLDATA
Transact
Home
C3.sol 1
C4.sol 1
pragma solidity ^0.6.0;
contract student_management {
struct student {
int stud_id;
string name;
string department;
Filter with transaction hash or address
transaction hash
block hash
block number
from
to
gas
transaction cost
input
output
decoded input
decoded output
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

13 >


DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 16
Deployed Contracts
STUDENT_MANAGEMENT AT
Balance: 0 ETH
add_stud
stud_id: 122
name: Abhishek More
department: Mechanical
Calldata
Parameters
transact
add_stud - transact (not payable)
getStudent
int256 stud_id
Low level interactions
CALLDATA
Transact
Home
C3.sol 1
C4.sol 1
pragma solidity ^0.6.0;
contract student_management {
struct student {
int stud_id;
string name;
string department;
Filter with transaction hash or address
transaction hash
block hash
block number
from
to
gas
transaction cost
input
output
decoded input
decoded output
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

14 >


DEPLOY & RUN TRANSACTIONS
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 16
Deployed Contracts
STUDENT_MANAGEMENT AT
Balance: 0 ETH
add_stud
int256 stud_id, string na
getStudent
stud_id: 122
Calldata
Parameters
call
0: string: Abhishek More
1: string: Mechanical
Low level interactions
CALLDATA
Transact
Home
C3.sol 1
C4.sol 1
pragma solidity ^0.6.0;
contract student_management {
struct student {
int stud_id;
string name;
string department;
Filter with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: student_management.getStudent(int256) data: 0xce5...0007a
Debug
from
to
execution cost
26812 gas (Cost only applies when called by a contract)
input
0xce5...0007a
output
decoded input
{
"int256 stud_id": "122"
}
decoded output
{
"0": "string: Abhishek More",
"1": "string: Mechanical"
}
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

Mini Project

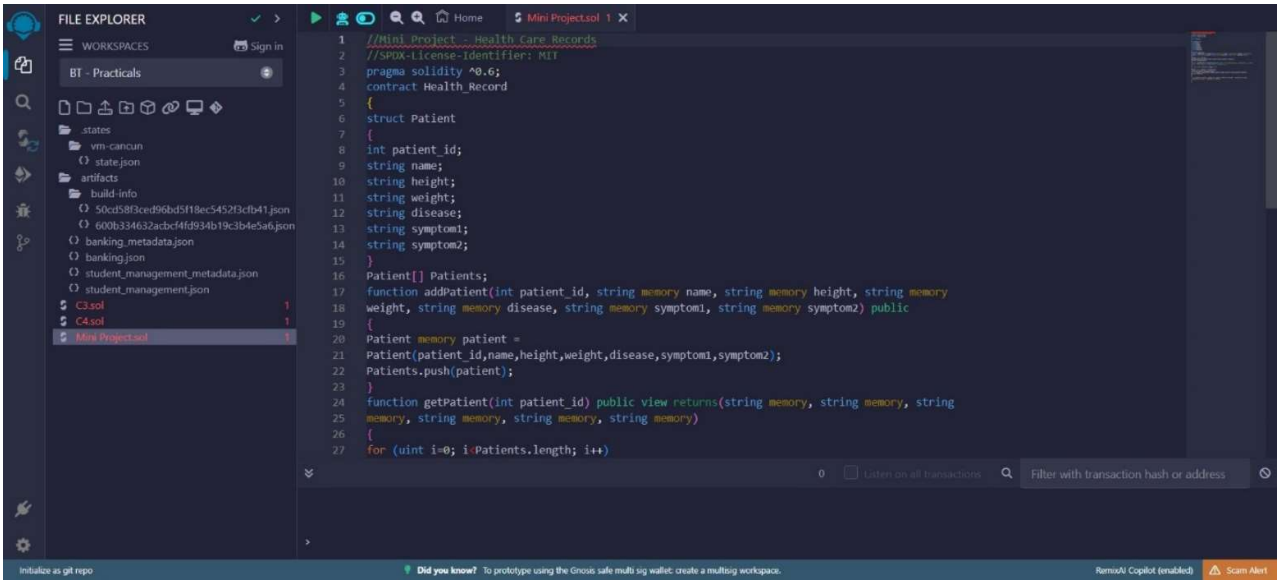
Code :

```
//SPDX-License-Identifier: MIT

pragma solidity ^0.6;

contract Health_Record

{

struct Patient

{

int patient_id;

string name;

string height;

string weight;

string disease;

string symptom1;

string symptom2;

}

Patient[] Patients;

function addPatient(int patient_id, string memory name, string memory height, string
memory

weight, string memory disease, string memory symptom1, string memory symptom2) public

{

Patient memory patient =

Patient(patient_id,name,height,weight,disease,symptom1,symptom2);

Patients.push(patient);

}

function getPatient(int patient_id) public view returns(string memory, string memory, string

memory, string memory, string memory, string memory)

{
```

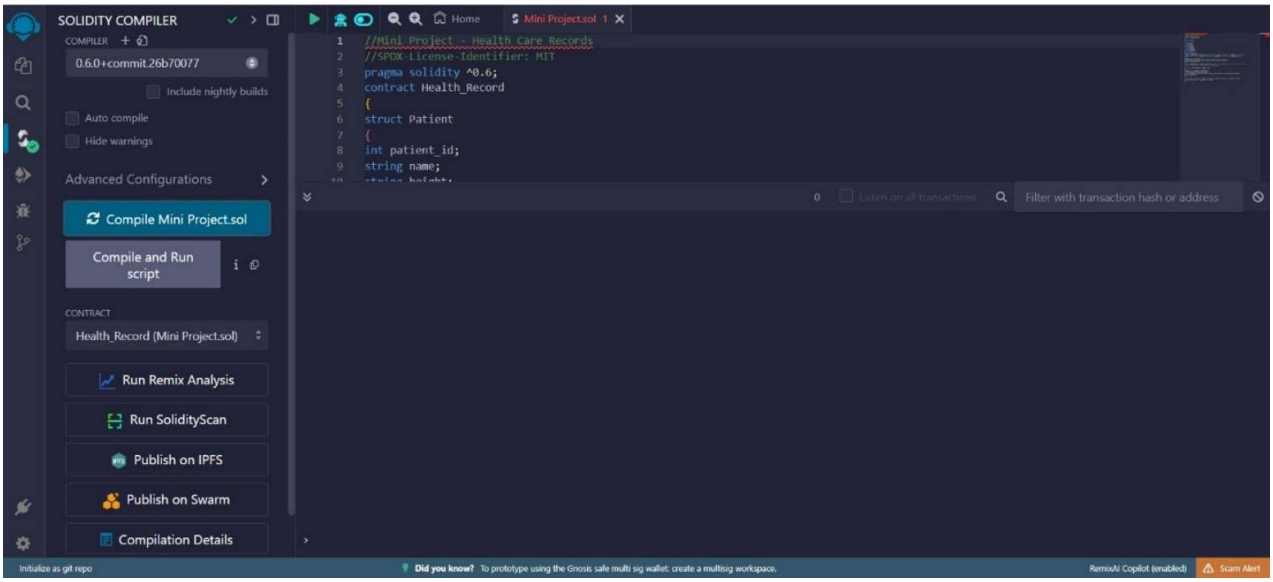
```
for (uint i=0; i<Patients.length; i++)
{
Patient memory patient = Patients[i];
if(patient.patient_id==patient_id)
{return(patient.name,patient.height,patient.weight,patient.disease,patient.symptom1,
patient.symptom2);
}
}
return("Name not Found", "Height not Found", "Weight not Found", "Disease not Found",
"Symptom1 not Found", "Symptom2 not Found");
}
}
```

Output :

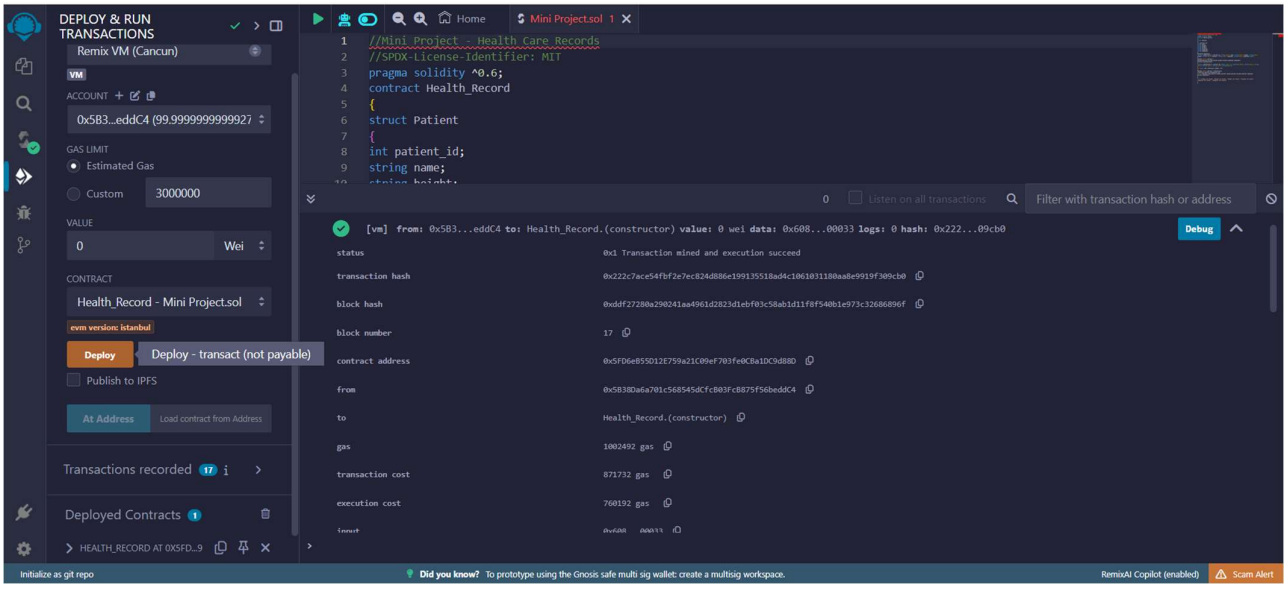
1 >



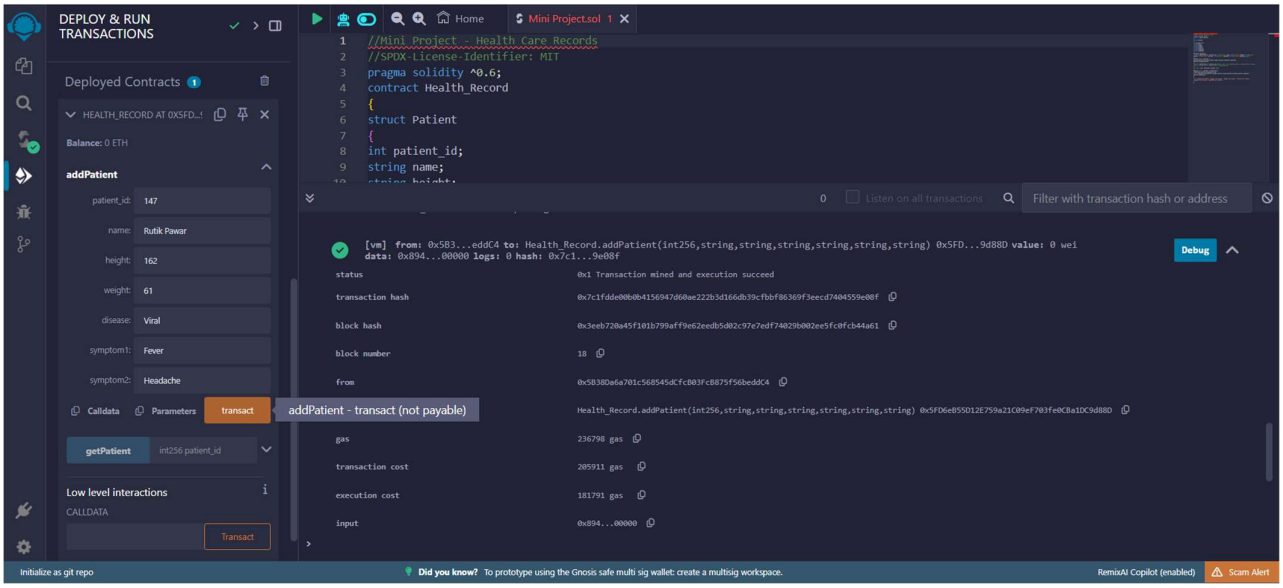
2 >


SOLIDITY COMPILER
COMPILER
0.6.0+commit.26b70077
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile Mini Project.sol
Compile and Run script
CONTRACT
Health_Record (Mini Project.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
Home
Mini Project.sol 1
//Mini Project - Health Care Records
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;
contract Health_Record
{
struct Patient
{
int patient_id;
string name;
Filter with transaction hash or address
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

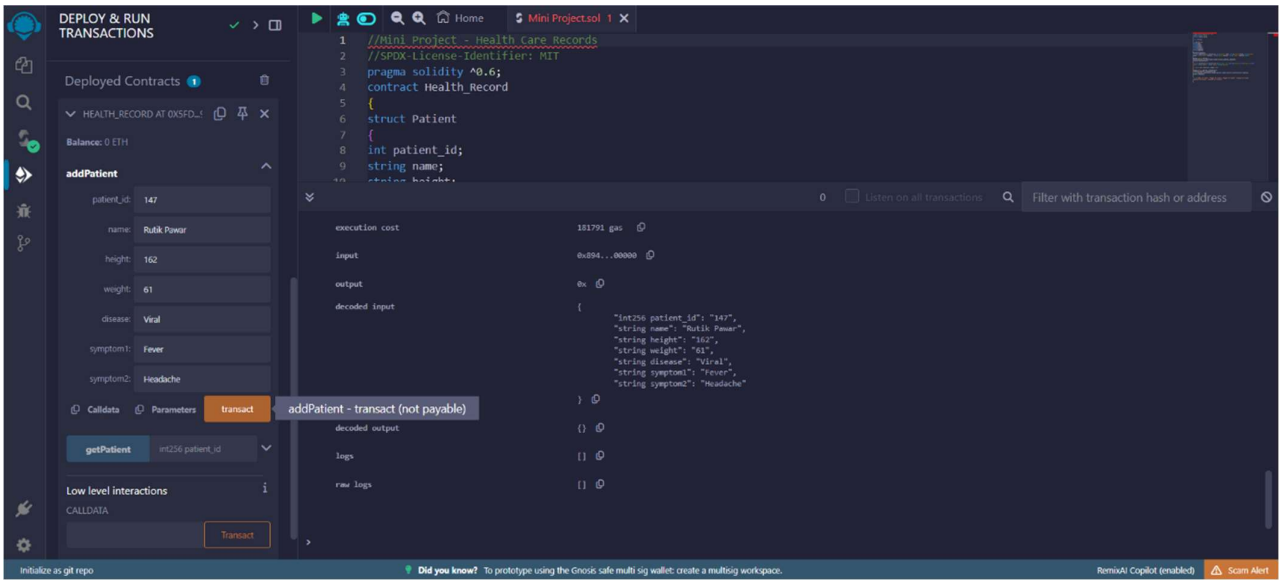
3 >

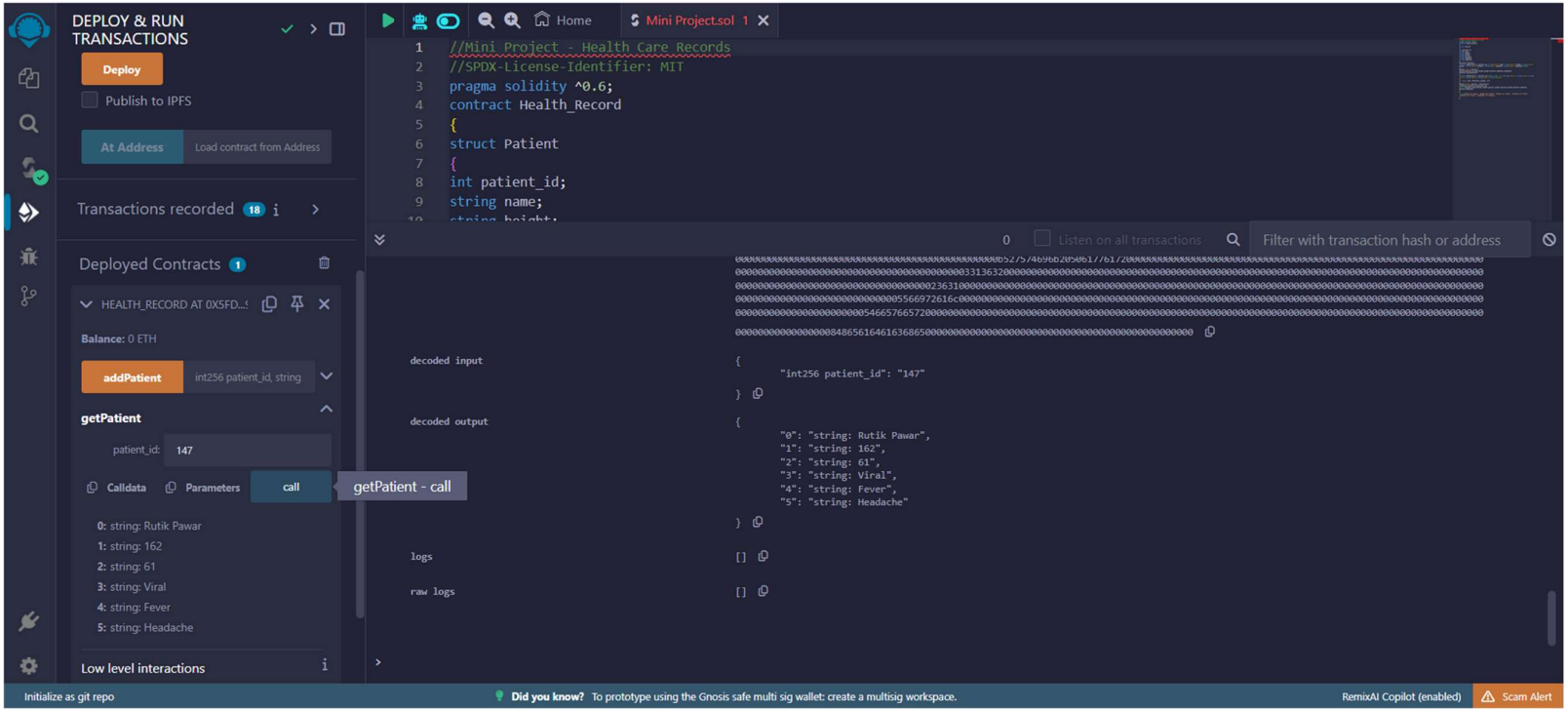

DEPLOY & RUN TRANSACTIONS
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (99.9999999999927
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
Health_Record - Mini Project.sol
evm version: istanbul
Deploy
Deploy - transact (not payable)
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 17
Deployed Contracts 1
HEALTH_RECORD AT 0X5FD...9
Home
Mini Project.sol 1
//Mini Project - Health Care Records
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;
contract Health_Record
{
struct Patient
{
int patient_id;
string name;
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Health_Record.(constructor) value: 0 wei data: 0x608...00033 logs: 0 hash: 0x222...09cb0
Debug
status
0x1 Transaction mined and execution succeed
transaction hash
0x222c7ace54fbf2e7ec824d886e199135518ad4c106103118 0aa8e9919f309cb0
block hash
0xddf27280a290241aa4961d2823d1ebf03c58ab1d11f8f540b1e973c32686896f
block number
17
contract address
0x5FD6eB55D12E759a21C09eF703fe0CBa1DC9d88D
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
Health_Record.(constructor)
gas
1002492 gas
transaction cost
871732 gas
execution cost
760192 gas
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

4 >


DEPLOY & RUN TRANSACTIONS
Deployed Contracts
HEALTH_RECORD AT 0X5FD...
Balance: 0 ETH
addPatient
patient_id: 147
name: Rutik Pawar
height: 162
weight: 61
disease: Viral
symptom1: Fever
symptom2: Headache
Calldata
Parameters
transact
addPatient - transact (not payable)
getPatient
int256 patient_id
Low level interactions
CALLDATA
Transact
Home
Mini Project.sol
//Mini_Project - Health_Care_Records
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;
contract Health_Record
{
struct Patient
{
int patient_id;
string name;
Listen on all transactions
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Health_Record.addPatient(int256,string,string,string,string,string,string) 0x5FD...9d88D value: 0 wei
data: 0x894...00000 logs: 0 hash: 0x7c1...9e08f
Debug
status
0x1 Transaction mined and execution succeed
transaction hash
0x7c1fdde00b0b4156947d60ae222b3d166db39cfbbf86369f3eecd7404559e08f
block hash
0x3eeb720a45f101b799aff9e62eedb5d02c97e7edf74029b002ee5fc0fcb44a61
block number
18
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
Health_Record.addPatient(int256,string,string,string,string,string,string) 0x5FD6eB55D12E759a21C09eF703fe0CBa1DC9d88D
gas
236798 gas
transaction cost
205911 gas
execution cost
181791 gas
input
0x894...00000
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

5 >


DEPLOY & RUN TRANSACTIONS
Deployed Contracts
HEALTH_RECORD AT 0X5FD...
Balance: 0 ETH
addPatient
patient_id: 147
name: Rutik Pawar
height: 162
weight: 61
disease: Viral
symptom1: Fever
symptom2: Headache
Calldata
Parameters
transact
addPatient - transact (not payable)
getPatient
int256 patient_id
Low level interactions
CALLDATA
Transact
Home
Mini Project.sol
//Mini_Project - Health_Care_Records
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;
contract Health_Record
{
struct Patient
{
int patient_id;
string name;
Listen on all transactions
Filter with transaction hash or address
execution cost
181791 gas
input
0x894...00000
output
0x
decoded input
{
"int256 patient_id": "147",
"string name": "Rutik Pawar",
"string height": "162",
"string weight": "61",
"string disease": "Viral",
"string symptom1": "Fever",
"string symptom2": "Headache"
}
decoded output
{}
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert


DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 18
Deployed Contracts 1
HEALTH_RECORD AT 0X5FD...
Balance: 0 ETH
addPatient
int256 patient_id, string
getPatient
patient_id: 147
Calldata
Parameters
call
getPatient - call
0: string: Rutik Pawar
1: string: 162
2: string: 61
3: string: Viral
4: string: Fever
5: string: Headache
Low level interactions
Home
Mini Project.sol 1
//Mini_Project - Health_Care_Records
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;
contract Health_Record
{
struct Patient
{
int patient_id;
string name;
Listen on all transactions
Filter with transaction hash or address
decoded input
"int256 patient_id": "147"
decoded output
"0": "string: Rutik Pawar",
"1": "string: 162",
"2": "string: 61",
"3": "string: Viral",
"4": "string: Fever",
"5": "string: Headache"
logs
[]
raw logs
[]
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert