Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE & COMMERCE

(EMPOWERED AUTONOMOUS COLLEGE)

AFFILIATED TO

UNIVERSITY OF MUMBAI

DEPARTMENT OF INFORMATION TECHNOLOGY

2024-2025

M.SC. (IT) PART I SEM II

PAPER RJSPIT203P – THEORY AND APPLICATIONS OF BLOCKCHAIN

Name :- *Vishwakarma Shivam Suresh Sushila*

Roll No. *6620*

Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Ghatkopar (W), Mumbai-400 086

Certificate

This is to certify that Mr./Ms. Vishwakarma Shivam Suresh Suhila Roll No 6620 of M.Sc.(I.T.) Part-1 class has completed the required number of experiments in the subject of Theory and Applications of Blockchain in the Department of Information Technology during the academic year 2024 - 2025.

Professor In-Charge

Co-ordinator of IT Department
Prof. Bharati Bhole

College Seal & Date

Examiner

INDEX

Sr. No	Title	Date
1	Demonstrate Working with MetaMask a. Install Metamask, Create account, Deposit Ether. b. Transfer ether in between accounts and observe the transaction details.	Nov 25, 2024
2	Implementing cryptography Algorithm in Python: a. DES b. SHA Message Digest	Nov 27, 2024
3	Write the following programs for Blockchain in Python : a. A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it b. A transaction class to send and receive money and test it .	Dec 2, 2024
4	Write the following programs for Blockchain in Python : a. Create multiple transactions and display them . b. Create a blockchain, a genesis block and execute it .	Dec 4, 2024
5	Write the following programs for Blockchain in Python : a. Create a mining function and test it. b. Add blocks to the miner and dump the blockchain.	Dec 11, 2024
6	Write the solidity Code and demonstrate the following: a. Variable b. Operators c. Decision Making - if, if else, if else if d. Loops - for, while, and do while	Dec 18, 2024
7	Write the solidity Code and demonstrate the following: a. Functions b. View Functions c. Pure Functions d. Cryptographic Function	Jan 1, 2025
8	Write the solidity Code and demonstrate the following: a. Contracts b. Inheritance c. Constructors	Jan 8, 2025

	d. Interfaces	
9	Program a Simple contract that can get, increment and decrement the count store in the contract.	Jan 15, 2025
10	Write a simple smart contract to calculate the ‘number of ethers’ for the transaction of gas limit for the given scenario. a. Mint Your Own Coins and Make Simple Transactions With Solidity b. Creating ERC -20 Token	Jan 22, 2025

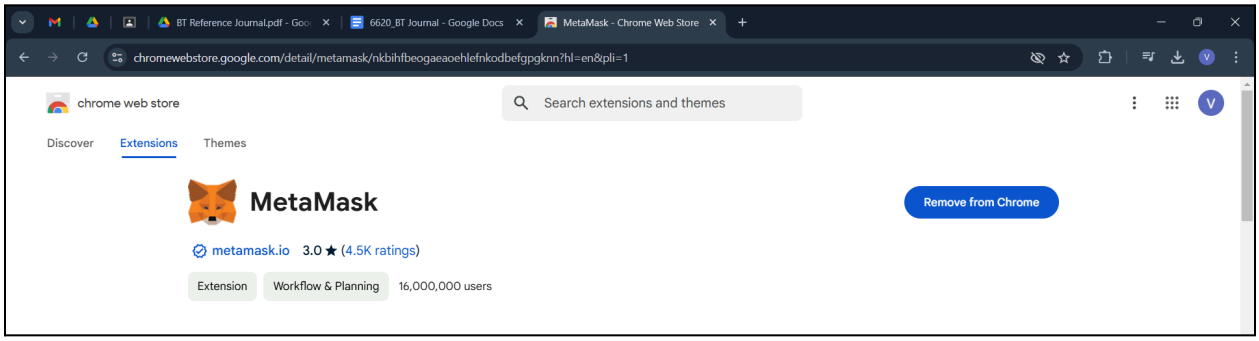
Practical 1: Demonstrate Working with MetaMask

a. Install Metamask ,Create account ,Deposit Ether
b. Transfer ether in between accounts and observe the transaction details

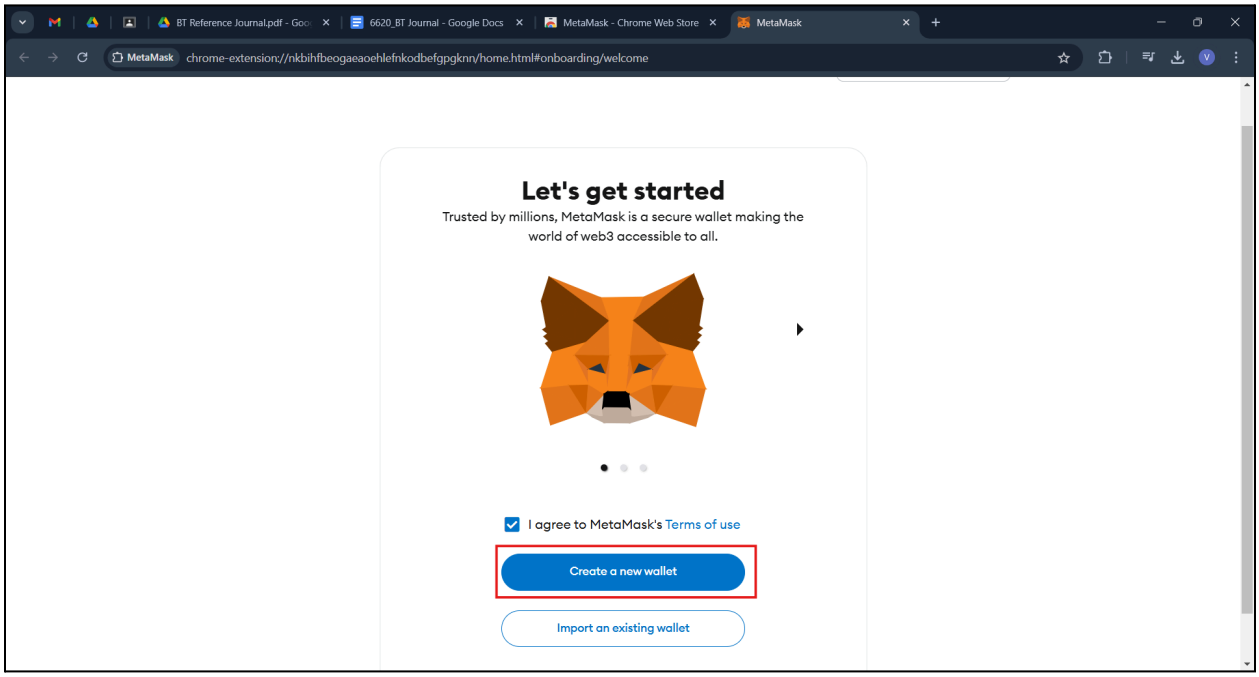
---

A. Install Metamask, Create account, Deposit Ether

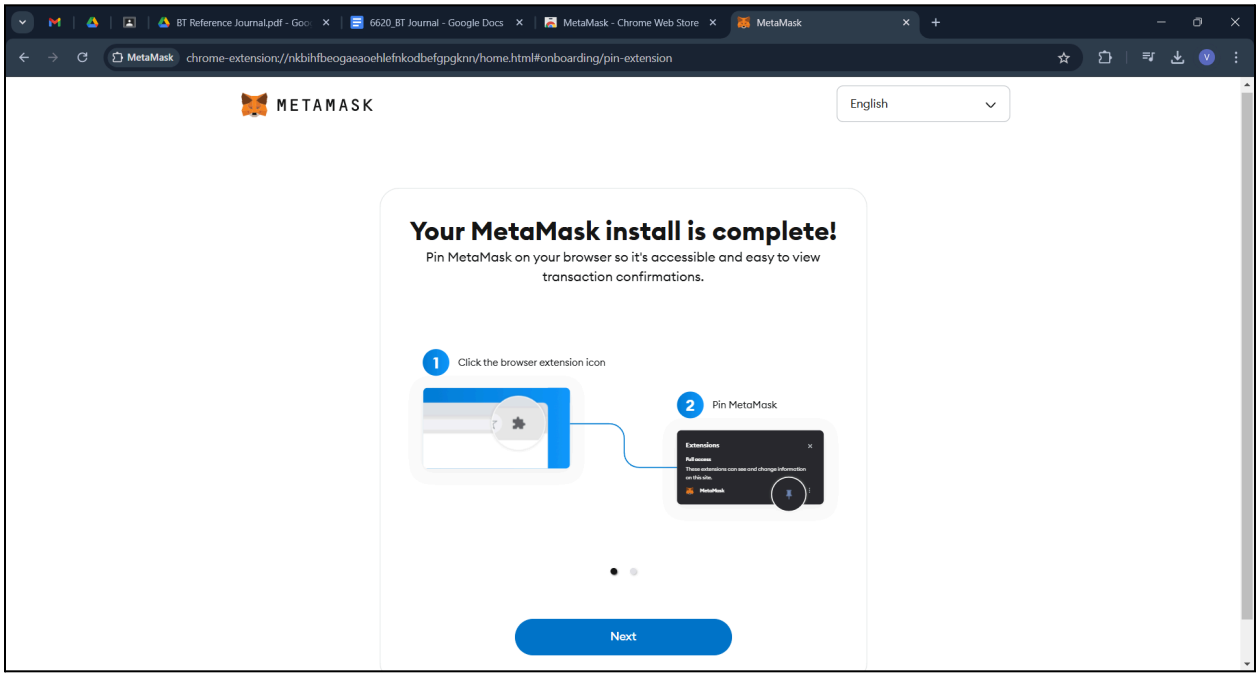
To Install search Metamask extension and add to chrome.

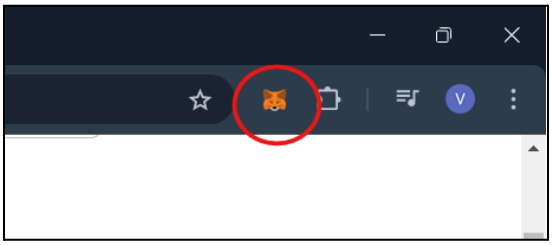


Create a wallet after adding an extension to chrome.

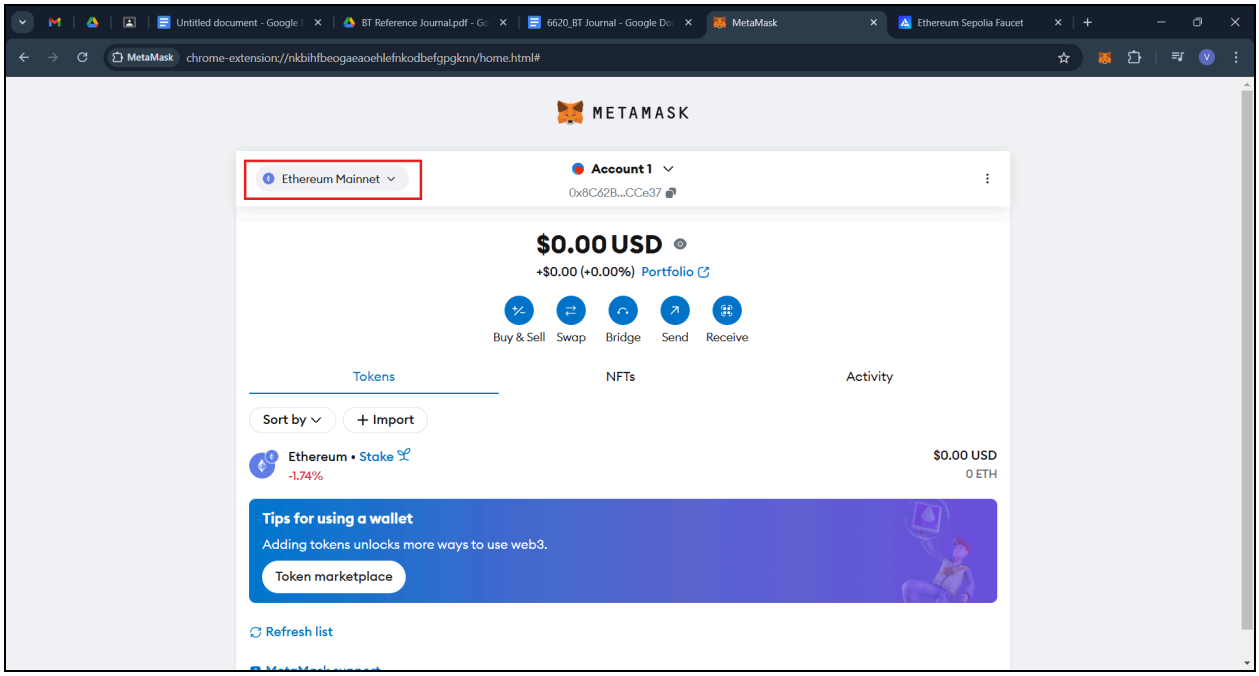


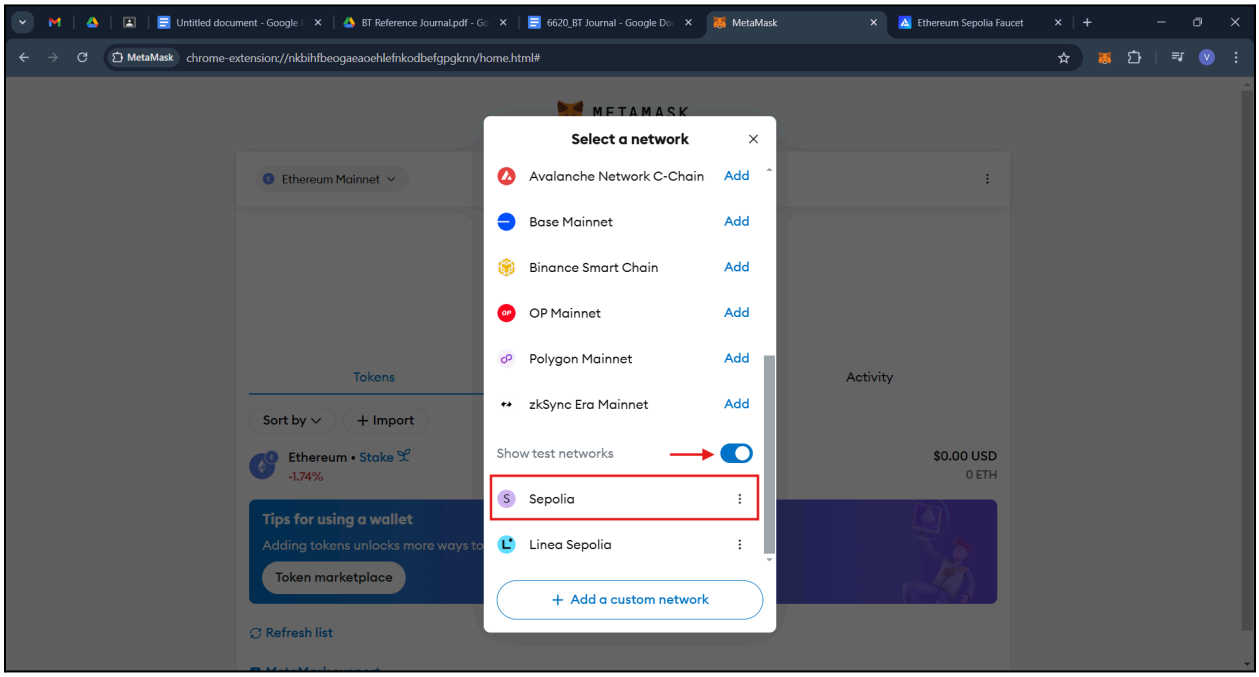
Set password and recovery key and your MetaMask is ready

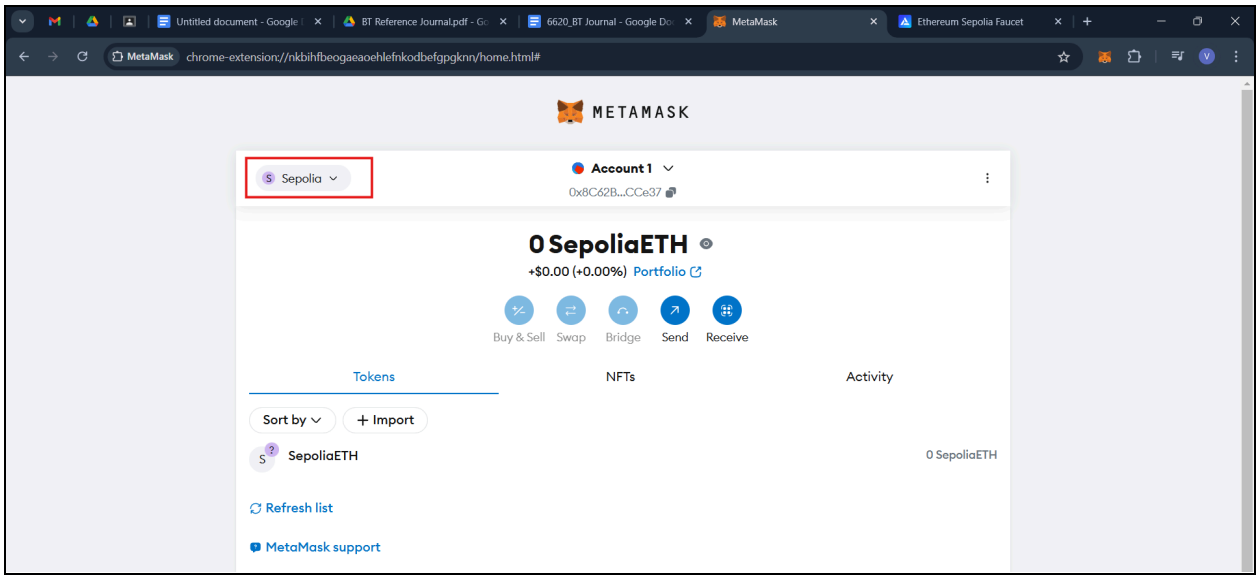




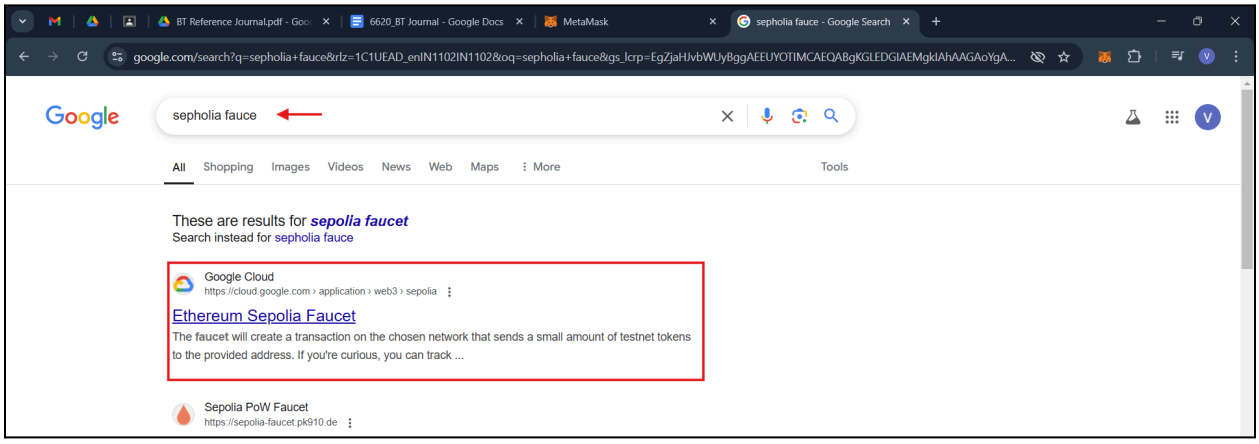
Now use Test Network Sepholia.


METAMASK
Ethereum Mainnet
Account 1
0x8C62B...CCe37
$0.00 USD
+$0.00 (+0.00%) Portfolio
Buy & Sell
Swap
Bridge
Send
Receive
Tokens
NFTs
Activity
Sort by
+ Import
Ethereum • Stake
-1.74%
$0.00 USD
0 ETH
Tips for using a wallet
Adding tokens unlocks more ways to use web3.
Token marketplace
Refresh list

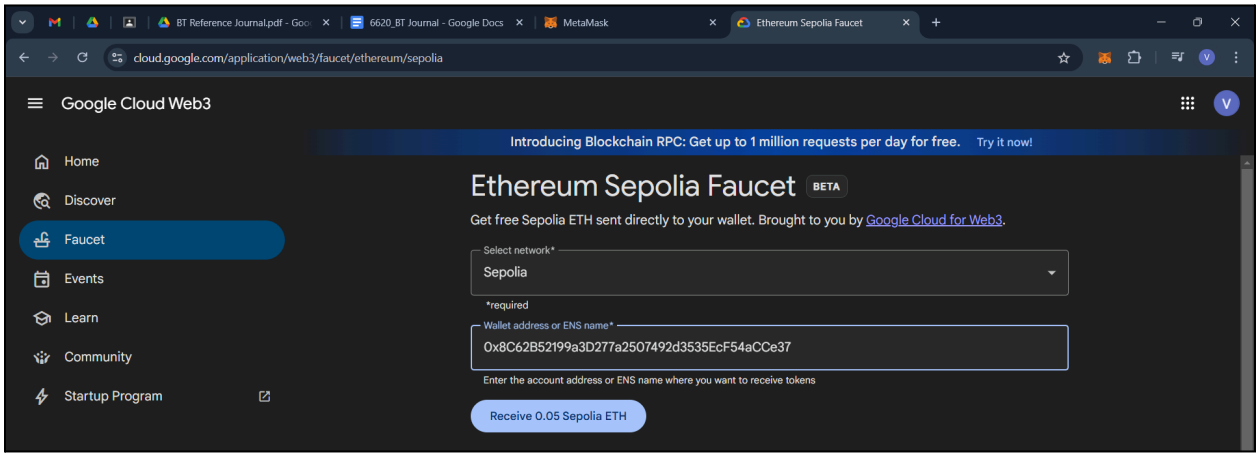

Select a network
Avalanche Network C-Chain Add
Base Mainnet Add
Binance Smart Chain Add
OP Mainnet Add
Polygon Mainnet Add
zkSync Era Mainnet Add
Show test networks
Sepolia
Linea Sepolia
+ Add a custom network

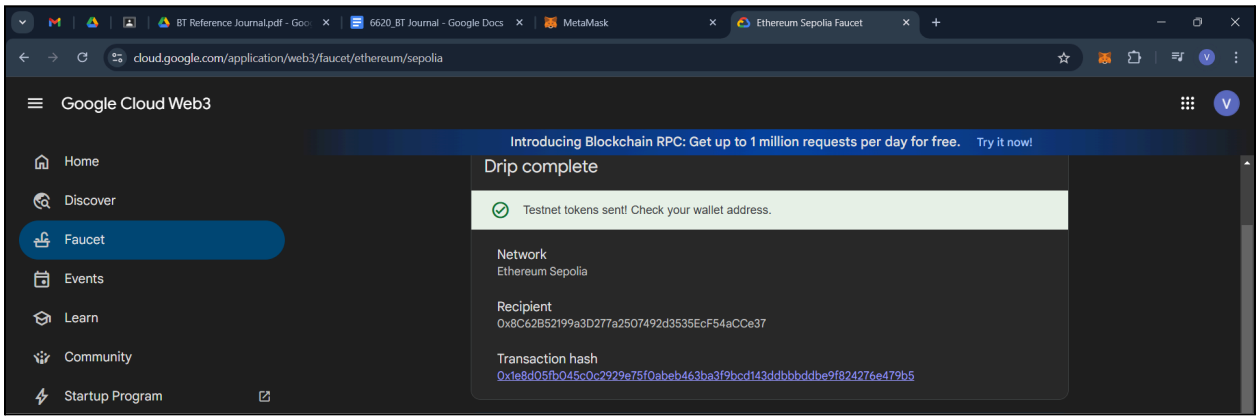


Add ETH into it.
Search Sepholia faucet on browser open google cloud link

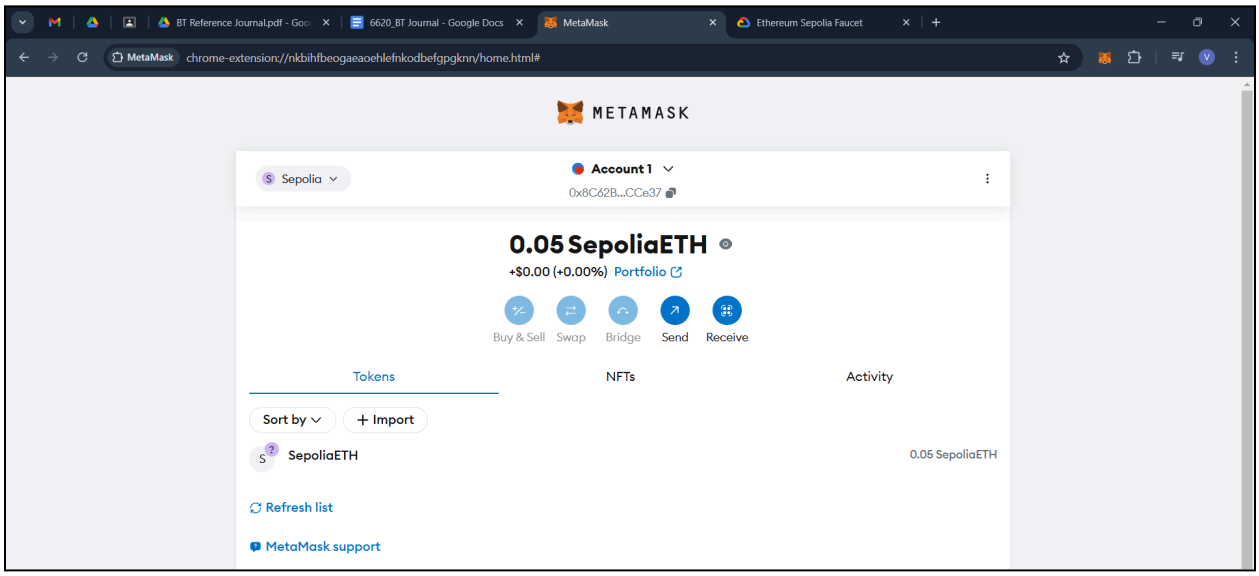


Enter your MetaMask Wallet address an click on receive ETH button





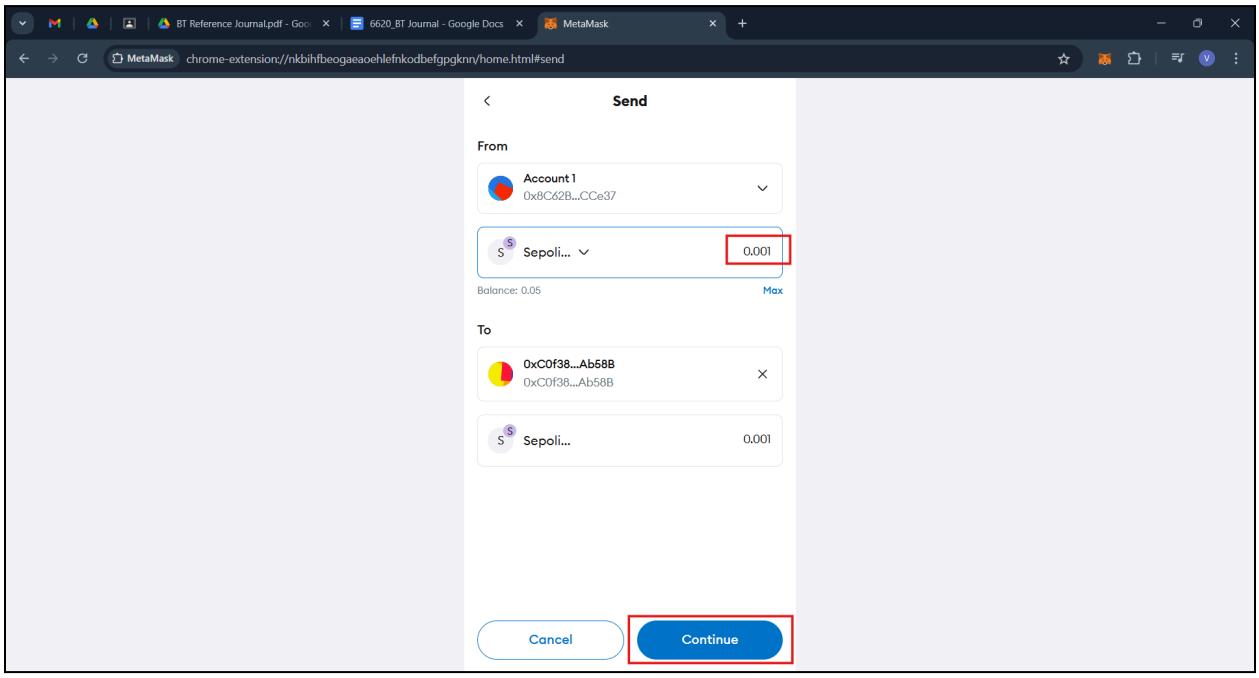
Check your wallet and 0.05 ETh are reflecting

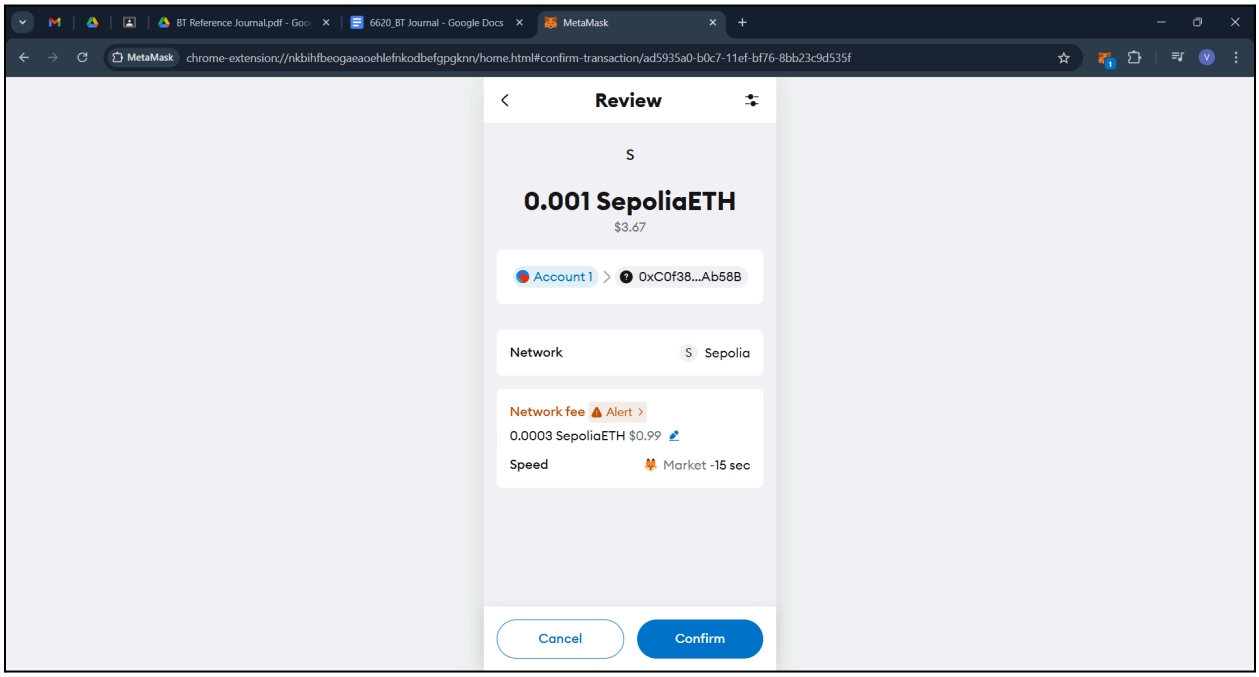


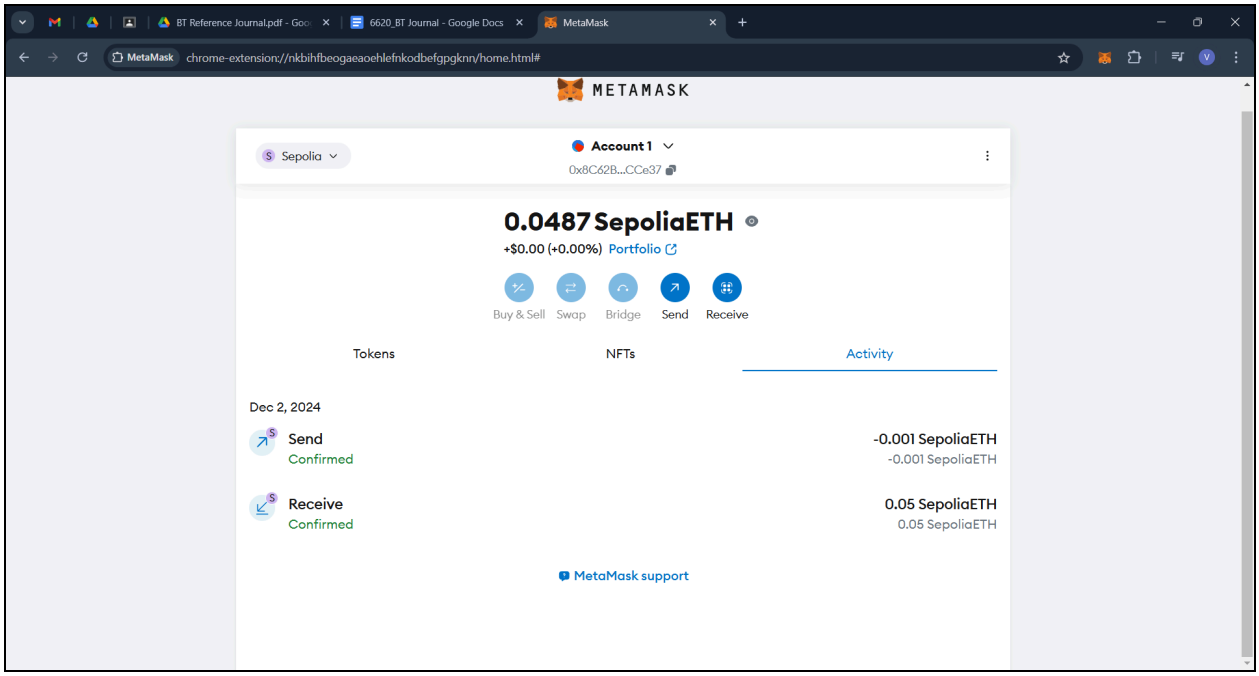
Once it is done now to see whether u can send or receive sepholia token or not now take your token and send it to you college and just sen 0.01 sepholia token with your friend

B. Transfer ether in between accounts and observe the transaction details

Click on send and add the address of the person you want to send






BT Reference Journal.pdf - Goo
6620_BT Journal - Google Docs
MetaMask
MetaMask
chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#
METAMASK
Sepolia
Account 1
0x8C62B...CCe37
0.0487 SepoliaETH
+$0.00 (+0.00%) Portfolio
Buy & Sell
Swap
Bridge
Send
Receive
Tokens
NFTs
Activity
Dec 2, 2024
Send
Confirmed
-0.001 SepoliaETH
-0.001 SepoliaETH
Receive
Confirmed
0.05 SepoliaETH
0.05 SepoliaETH
MetaMask support

Practical 2: Implementing cryptography Algorithm in Python:

a. DES
b. SHA Message Digest

A. DES

Data Encryption Standard (DES) is a symmetric key encryption algorithm that uses a block cipher to transform fixed-length blocks of plaintext into ciphertext. In Python, the Crypto library provides an implementation of DES. The des_encrypt function encrypts a given plaintext using DES with a specified key, and des_decrypt decrypts the ciphertext back to the original plaintext. DES, although historically significant, is considered insecure for modern applications, and more advanced algorithms like Advanced Encryption Standard (AES) are recommended

Install the following library for further practicals
pip install pycryptodome

```
import datetime
from Crypto.Cipher import DES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad

def generate_key():
    return get_random_bytes(8)

def encrypt(message, key):
    cipher = DES.new(key, DES.MODE_ECB)
    encrypted_message = message.encode('utf-8')
    padded_message = pad(encrypted_message, DES.block_size)
    ciphertext = cipher.encrypt(padded_message)
    return ciphertext

def decrypt(ciphertext, key):
    cipher = DES.new(key, DES.MODE_ECB)
    decrypted_message = cipher.decrypt(ciphertext)
    unpadded_message = unpad(decrypted_message, DES.block_size)
    ciphertext = unpadded_message.decode('utf-8')
    return ciphertext
```

```python
if __name__ == "__main__":
    print(datetime.datetime.now())
    key = generate_key()
    message = "6620_Shivam"
    ciphertext = encrypt(message, key)
    print("Encrypted:", ciphertext.hex())
    decrypted_message = decrypt(ciphertext, key)
    print("Decrypted:", decrypted_message)
```

```
PS D:\6620\BT> python -u "d:\6620\BT\des.py"
2024-12-03 07:59:32.780246
Encrypted: ba0e77db8fceca21e4e4c8f082275ac4
Decrypted: 6620_Shivam
PS D:\6620\BT>
```

Explanation:
generate_key(): Generates a random 8-byte key for DES encryption.
encrypt(message, key): Encrypts the given message using DES in ECB mode.
decrypt(ciphertext, key): Decrypts the ciphertext back to the original message.
pad() and unpad(): Ensures the message is padded to a multiple of DES block size (8 bytes).

B. SHA Message Digest

Secure Hash Algorithm (SHA) is a family of cryptographic hash functions designed to produce a fixed-size hash value, typically represented as a hexadecimal number. In Python, the built-in hashlib module provides implementations of various SHA algorithms. The sha_digest function demonstrates the SHA-256 algorithm, computing the hash digest of an input message. SHA is widely used for generating secure message digests in applications like data integrity verification and password hashing.

```python
import datetime
import hashlib

def sha256(message):
    # Create a new SHA-256 hash object
    sha256_hash = hashlib.sha256()

    # Update the hash object with the bytes-like object (message)
    sha256_hash.update(message.encode('utf-8'))

    # Get the hexadecimal representation of the digest
    digest = sha256_hash.hexdigest()

    return digest

print(datetime.datetime.now())
message = "6620_Shivam"
result = sha256(message)
print(f"SHA-256 Digest for '{message}': {result}")
```

```
PS D:\6620\BT> python -u "d:\6620\BT\SHA_digest.py"
2024-12-03 08:02:08.175458
SHA-256 Digest for '6620_Shivam': 93e0ca924d001356b9131f9ce6f0824c579065620072add3efd4ba3808f5274c
PS D:\6620\BT>
```

Practical 3: Write the following programs for Blockchain in Python :

a. A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it
b. A transaction class to send and receive money and test it .

A. A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it

RSA, named after its inventors Rivest, Shamir, and Adleman, is an asymmetric key encryption algorithm widely used for secure communication. It relies on the mathematical challenge of factoring large primes. Each participant has a pair of keys: a public key for encryption and a private key for decryption. RSA's security hinges on the difficulty of factoring large primes, making it a fundamental component in securing
online transactions and communications.

```
import datetime
import binascii
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5

class Client:
    def __init__(self):
        random_gen = Random.new().read
        self._private_key = RSA.generate(1024, random_gen)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return
binascii.hexlify(self._public_key.export_key(format='DER')).decode('ascii'
)

# Create a new client instance
client = Client()
print(datetime.datetime.now())
print("Sender: ", client.identity)
```

```
D:\MSc-IT\SEM 2\BT\Practical 2>python -u "d:\MSc-IT\SEM 2\BT\Practical 2\RSA.py"
2024-12-05 21:21:46.721930
Sender:  30819f300d06092a864886f70d010101050003818d0030818902818100ce3bfacdde6223b57bc9e77ccc5b422
261389344c746f6381197430915df517a44d135bc7e0e9b14355f916bc127b87e49551b01fde3d3934173fad1e1c25ebc2
760064475a872b9b6269eb1e977336233ad98678dfa6948ece11664fef2f5fb4d77955af50f5d09015e6daab366696e371
90ea7473fecfcb0ba5b5af6a044bb0203010001

D:\MSc-IT\SEM 2\BT\Practical 2>
```

Explanation:

Client class:

- Generates a 1024-bit RSA key pair (private and public keys).
- The public key is used to derive the client's identity (a unique identifier).

Identity property: Returns the hexadecimal representation of the public key.

B. A transaction class to send and receive money and test it .

```
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
from Crypto.PublicKey import RSA
import binascii
import datetime
import collections

class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return
binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()

    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity

        return collections.OrderedDict({
            'sender': identity,
            'recipient': self.recipient,
            'value': self.value,
```

```
            'time': self.time
        })

    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf-8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')

def display_transaction(transaction):
    _dict = transaction.to_dict()
    print("sender: " + _dict['sender'])
    print()
    print("recipient: " + _dict['recipient'])
    print()
    print("value: " + str(_dict['value']))
    print()
    print("time: " + str(_dict['time']))
    print()

# Create two clients and a transaction between them
transactions = []
A = Client()
B = Client()

# Create a transaction from A to B
t1 = Transaction(A, B.identity, 15.0)

# Sign the transaction
signature = t1.sign_transaction()

# Display the transaction
display_transaction(t1)
print("Signature:", signature)
```

Explanation:

Client class:

- Generates an RSA key pair (private and public).
- The identity is the public key in hexadecimal format.

Transaction class:

- Represents a transaction with sender, recipient, and value.
- to_dict(): Converts the transaction details into an ordered dictionary.
- sign_transaction(): Signs the transaction using the sender's private key.

display_transaction(): Prints the details of the transaction.

Practical 4: Write the following programs for Blockchain in Python :

a. Create multiple transactions and display them.
b. Create a blockchain, a genesis block and execute it .

A. Create multiple transactions and display them.

Multiple transactions within a blockchain refer to the numerous data exchanges that occur between participants in a blockchain network. Each transaction involves the transfer or modification of digital assets or information, and these transactions are grouped together into blocks. The blockchain, as a decentralized and distributed ledger, ensures transparency and security by recording these transactions in a sequential and immutable manner across all network nodes.

```
import binascii
import datetime
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5
from collections import OrderedDict

class Client:
    def __init__(self):
        random = Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return
binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')

class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
```

```
        self.value = value
        self.time = datetime.datetime.now()

    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
        return OrderedDict({
            'sender': identity,
            'recipient': self.recipient,
            'value': self.value,
            'time': self.time
        })

    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf-8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')

def display_transaction(transaction):
    _dict = transaction.to_dict()
    print("sender: " + _dict['sender'])
    print()
    print("recipient: " + _dict['recipient'])
    print()
    print("value: " + str(_dict['value']))
    print()
    print("time: " + str(_dict['time']))
    print()

# Create clients
alice = Client()
bob = Client()
jhon = Client()

# Create transactions
transactions = []
```

```python
t1 = Transaction(alice, bob.identity, 15.0)
t1.sign_transaction()
transactions.append(t1)

t2 = Transaction(bob, jhon.identity, 25.0)
t2.sign_transaction()
transactions.append(t2)

t3 = Transaction(bob, jhon.identity, 200.0)
t3.sign_transaction()
transactions.append(t3)

# Display all transactions
tn = 1
for t in transactions:
    print("Transaction #", tn)
    display_transaction(t)
    tn += 1
    print()
```

```
D:\MSc-IT\SEM 2\BT\Practical 2>python -u "d:\MSc-IT\SEM 2\BT\Practical 2\prac_4a.py"
Transaction # 1
sender: 30819f300d06092a864886f70d010101050003818d00308189028181008d440c99dfd1d46b7625150f6c8952b12
929b063ac12a78d21c27c2b257194cf35b09cf8cf1f92d25695e78510ff0997a60ad940ad5ab2c8d915609bf8005d313b7e
82a78b75b90d847ede2a15f460855a5a489f37b48d8bee2e1858b1a57a75fb381502745f0d397c0928ff5c7bdcde6bc78b5
7d94b2ae8637285ff60f0678d0203010001

recipient: 30819f300d06092a864886f70d010101050003818d00308189028181008b7c03845b1c302f267a965cd8311fe
e00ea681694aee2ad9ce4ae392139f068f702509d4e504392bdbe73356ff4686686581e012b402943ec2b422e93c60ebcb1
2d341cde23c3b5566a3a2c3752cfca846c40fd8ce1499fccd5f3ae03b5e80a0297b2bba4d115a14d200f8bedb5905965ab5
d50d2a48037a8c2ce38c1c6d2b390203010001

value: 15.0

time: 2024-12-05 21:32:10.064159
```

```
Transaction # 2
sender: 30819f300d06092a864886f70d010101050003818d00308189028181008b7c03845b1c302f267a965cd8311fee00
ea681694aee2ad9ce4ae392139f068f702509d4e504392bdbe73356ff4686686581e012b402943ec2b422e93c60ebcb12d3
41cde23c3b5566a3a2c3752cfca846c40fd8ce1499fccd5f3ae03b5e80a0297b2bba4d115a14d200f8bedb5905965ab5d50
d2a48037a8c2ce38c1c6d2b390203010001

recipient: 30819f300d06092a864886f70d010101050003818d00308189028181009e7077bb1ae152eef67359cfe47efb
b526964266021aae57ad382562eea8247150837cd5a49ef4d24f10424e276dfe6356671daf430070232bcc9d58755c71bf0
f8e9b30a69a296360242c159c8a287dcec8ea63013946f4bd5d5bc68a0fb6285bcc59c6c804752171520266b28c0dca7cc0
1c8ecfa7cb4a97792cd1da6a43490203010001

value: 25.0

time: 2024-12-05 21:32:10.064159
```

```
Transaction # 3
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100b7c03845b1c302f267a965cd8311fee00
ea681694aee2ad9ce4ae392139f068f702509d4e504392bdbe73356ff4686686581e012b402943ec2b422e93c60ebcb12d3
41cde23c3b5566a3a2c3752cfca846c40fd8ce1499fccd5f3ae03b5e80a0297b2bba4d115a14d200f8bedb5905965ab5d50
d2a48037a8c2ce38c1c6d2b390203010001

recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100e7077bb1ae152eef67359cfe47efb
b526964266021aae57ad382562eea8247150837cd5a49ef4d24f10424e276dfe6356671daf430070232bcc9d58755c71bf0
f8e9b30a69a296360242c159c8a287dcec8ea63013946f4bd5d5bc68a0fb6285bcc59c6c804752171520266b28c0dca7cc0
1c8ecfa7cb4a97792cd1da6a43490203010001

value: 200.0

time: 2024-12-05 21:32:10.064159
```

Explanation:

Client class:

- Generates a 1024-bit RSA key pair.
- The identity property returns the public key in hexadecimal form.

Transaction class:

- Represents a transaction with a sender, recipient, and value.
- to_dict(): Converts the transaction into an ordered dictionary.
- sign_transaction(): Signs the transaction using the sender's private key.

display_transaction(): Prints the details of each transaction.

B. Create a blockchain, a genesis block and execute it .

The term "genesis block" refers to the inaugural block in a blockchain. It serves as the starting point of the entire blockchain network and contains no reference to a previous block. The creation of the genesis block marks the initiation of the blockchain, and subsequent blocks build upon it as the network grows. The genesis block often includes certain unique characteristics or messages, and its details are crucial for
verifying the integrity of the entire blockchain.

```
import hashlib
import datetime

class Block:
    def __init__(self, index, previous_hash, timestamp, data, hash):
        self.index = index
        self.previous_hash = previous_hash
        self.timestamp = timestamp
        self.data = data
        self.hash = hash

def calculate_hash(index, previous_hash, timestamp, data):
    block_info = f"{index}{previous_hash}{timestamp}{data}"
    return hashlib.sha256(block_info.encode()).hexdigest()

def create_block(index, previous_hash, data):
    timestamp = datetime.datetime.now()
    hash = calculate_hash(index, previous_hash, timestamp, data)
    return Block(index, previous_hash, timestamp, data, hash)

# Create a blockchain with a genesis block
blockchain = [Block(0, "0" * 64, datetime.datetime.now(), "Genesis Block",
                    calculate_hash(0, "0" * 64, datetime.datetime.now(),
"Genesis Block"))]

# Add a new block to the blockchain
data_for_new_block = "Some data for the new block"
new_block = create_block(len(blockchain), blockchain[-1].hash,
data_for_new_block)
blockchain.append(new_block)
```

```python
# Display the state of the blockchain
for block in blockchain:
    print(f"\nBlock #{block.index} - Timestamp: {block.timestamp}, Data:
{block.data}, Hash: {block.hash}")
```

```
D:\MSc-IT\SEM 2\BT\Practical 2>python -u "d:\MSc-IT\SEM 2\BT\Practical 2\prac_4b.py"

Block #0 - Timestamp: 2024-12-05 21:34:52.369565, Data: Genesis Block, Hash: 38f2f144b7505fb9e5e22a
a5c0e0f32f28ed7c8653e799fc240c366d99264774

Block #1 - Timestamp: 2024-12-05 21:34:52.369565, Data: Some data for the new block, Hash: 7afce965
79d46c200ec864a699c2338c268617431a8cbcf4587e423c427cad0e

D:\MSc-IT\SEM 2\BT\Practical 2>
```

Explanation:

Block class: Represents a block in the blockchain with the following properties: index, previous_hash, timestamp, data, and hash.

calculate_hash function: Computes the SHA-256 hash of a block based on its index, previous_hash, timestamp, and data.

create_block function: Creates a new block with a unique hash.

Genesis block: The first block in the blockchain, which has a fixed hash of 0 for the previous_hash.

Adding a new block: A new block is added to the blockchain after the genesis block, with some sample data.

Display blockchain: Prints out each block's index, timestamp, data, and hash.

Practical 5: Write the following programs for Blockchain in Python :

a. Create a mining function and test it.

A. Create a mining function and test it.

In a blockchain network, mining is the process by which new blocks are added to the blockchain through the completion of complex mathematical computations. The primary purpose of mining is to secure and validate transactions within the network, ensuring the integrity of the decentralized ledger.

```
import hashlib

def sha256(message):
    return hashlib.sha256(message.encode('ascii')).hexdigest()

def mine(message, difficulty=1):
    assert difficulty >= 1
    # Prefix of '1's based on difficulty
    prefix = '1' * difficulty
    print("Prefix:", prefix)

    for i in range(1000):
        # Create a hash using the message and the nonce (i)
        digest = sha256(str(hash(message)) + str(i))
        print("Testing --> " + digest)

        # Check if the hash starts with the desired prefix
        if digest.startswith(prefix):
            print("\nAfter", i, "iterations,\nfound nonce:", digest)
            return i  # Return the nonce value

# Run mining with difficulty level 2
mine("test message", 2)
```

```
D:\MSc-IT\SEM 2\BT\Practical 2>python -u "d:\MSc-IT\SEM 2\BT\Practical 2\prac_5a.py"
Prefix: 11
Testing --> ce21913faf1526a80694692a491fa8a5b47ec65cbd9ff3f2e2f5a79bd4db0aa6
Testing --> 0e12baf757c3b87babf966a9ba12a82649c5628d6707daf9c31450f7ef43f55a
Testing --> 7d3ab115d1a78bceda8ce0aeb9ed77ad5bc3fdc3183aea154c0c3d97cd723bcc
Testing --> 6b0442e5d00348da054feb84ae9d203a023d2098bf787a35dd0b436e9e63001d
Testing --> 236cbe66095bdece65cd73d8d48b108477784e9bc14a0ce43a520d11a448c9d7
Testing --> 0dddc47ca4c4b6a43e45bf020119c7d5e83098bcc6a7f8bed78af297db1a401f
```

.
.
.
.

```
Testing --> 3c7fd3fcf34c6683e19661f41a3b77119c8d261d2855f20d94c3c2f4c852563a
Testing --> bd3d0f38247fb9217b09f5829fe414c53645ebb3bf91597b5da8757728b6eaad
Testing --> eab00e1d9fc4792bd3a3871998e6a03e5e54803c8b25d492aa6c8ca1be4b7303
Testing --> 11597f5efcf4ba7a9074fac0e08ddee0ee94c345318f4d614898a87934ee1d43

After 227 iterations,
found nonce: 11597f5efcf4ba7a9074fac0e08ddee0ee94c345318f4d614898a87934ee1d43

D:\MSc-IT\SEM 2\BT\Practical 2>
```

Explanation:

sha256 function: Takes a message, encodes it into ASCII, and returns the SHA-256 hash of the message.

mine function:

- Attempts to find a nonce (a number) that, when appended to the message and hashed, results in a digest that starts with a specific number of '1's, determined by the difficulty parameter.
- The function runs for up to 1000 iterations, printing each hash it checks.
- If it finds a hash that starts with the specified number of '1's, it prints the nonce and the hash and returns the nonce value.

Prefix: The prefix is a string consisting of '1' repeated for the number of times defined by the difficulty.

Additional Notes:

- The difficulty parameter determines how many '1's the hash needs to start with (e.g., difficulty 1 means the hash must start with '1', difficulty 2 means it must start with '11').
- The hash will be tested up to 1000 iterations, and the first valid nonce will be returned.

Practical 6: Write the solidity Code and demonstrate the following:

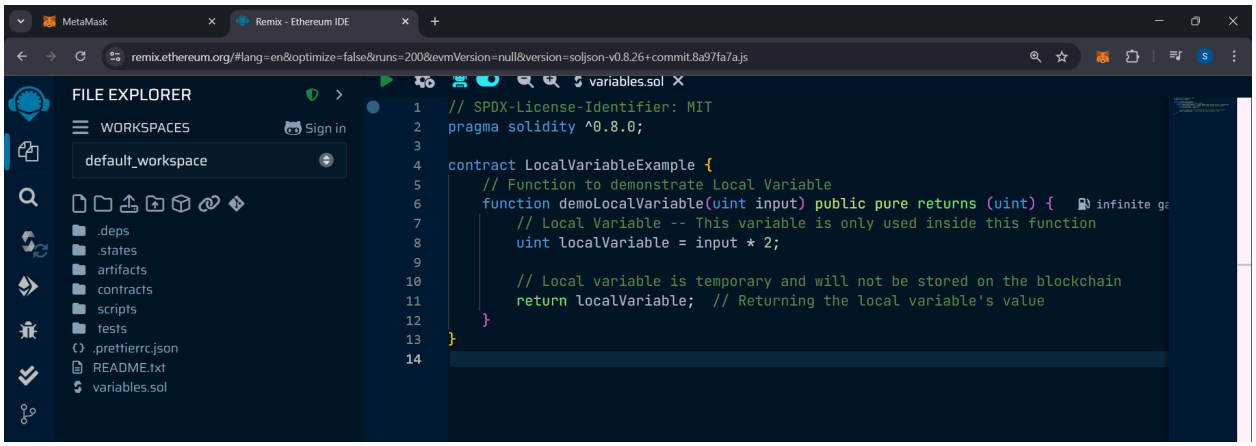
a. Variable
b. Operators
c. Decision Making - if, if else, if else if
d. Loops - for, while, and do while

A. Variable

1. Local Variable

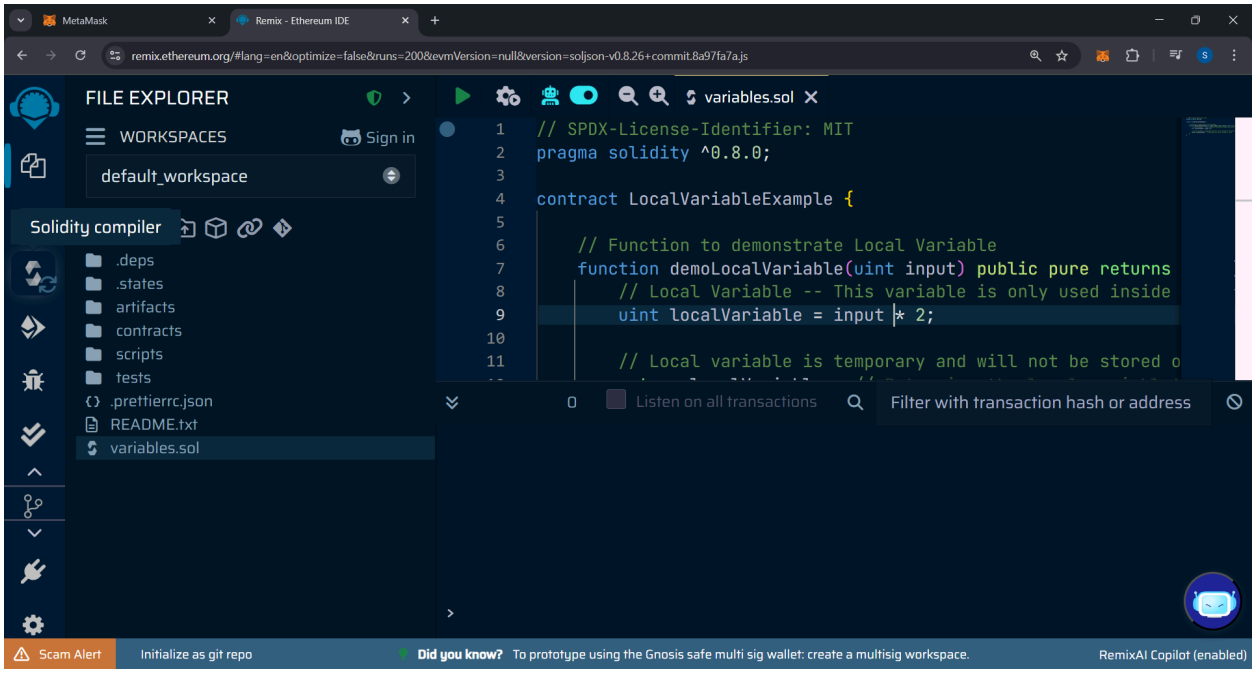
Local variables are declared within a specific function or code block and exist only for the duration of that particular execution. They are used for temporary storage and calculations within the confines of a single function.

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

contract LocalVariableExample {
    // Function to demonstrate Local Variable
    function demoLocalVariable(uint input) public pure returns (uint) {
        // Local Variable -- This variable is only used inside this function
        uint localVariable = input * 2;

        // Local variable is temporary and will not be stored on the blockchain
        return localVariable;  // Returning the local variable's value
    }
}
```

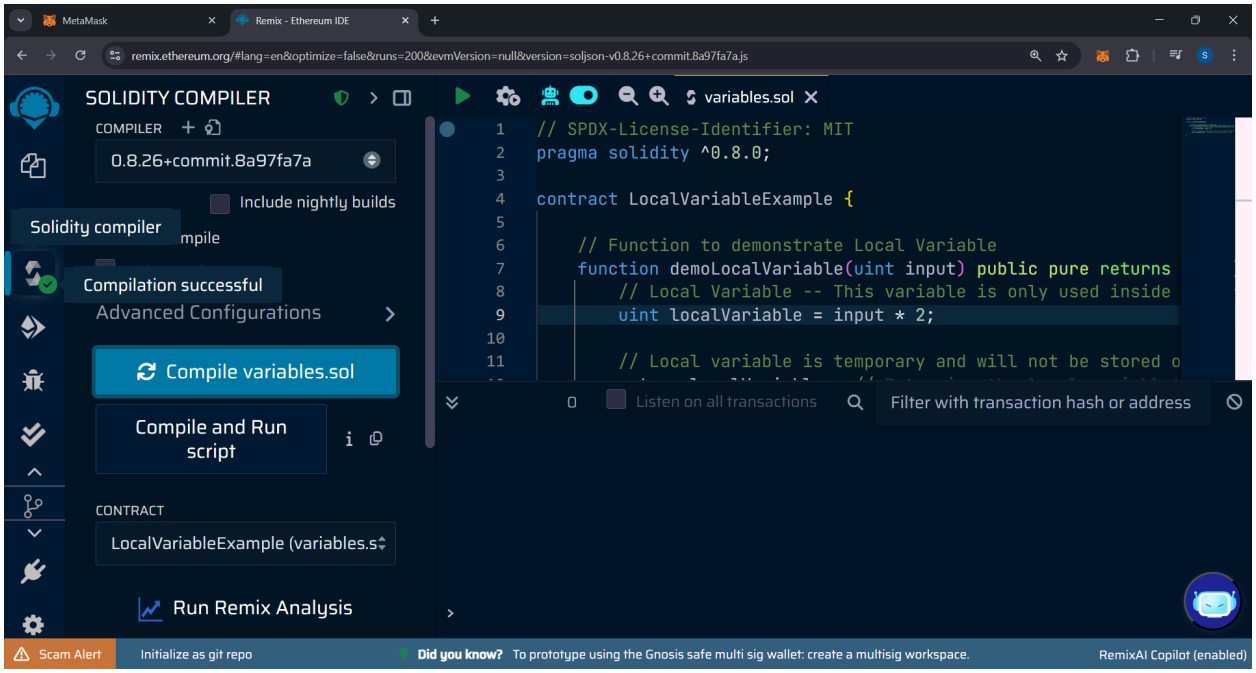


Steps to run

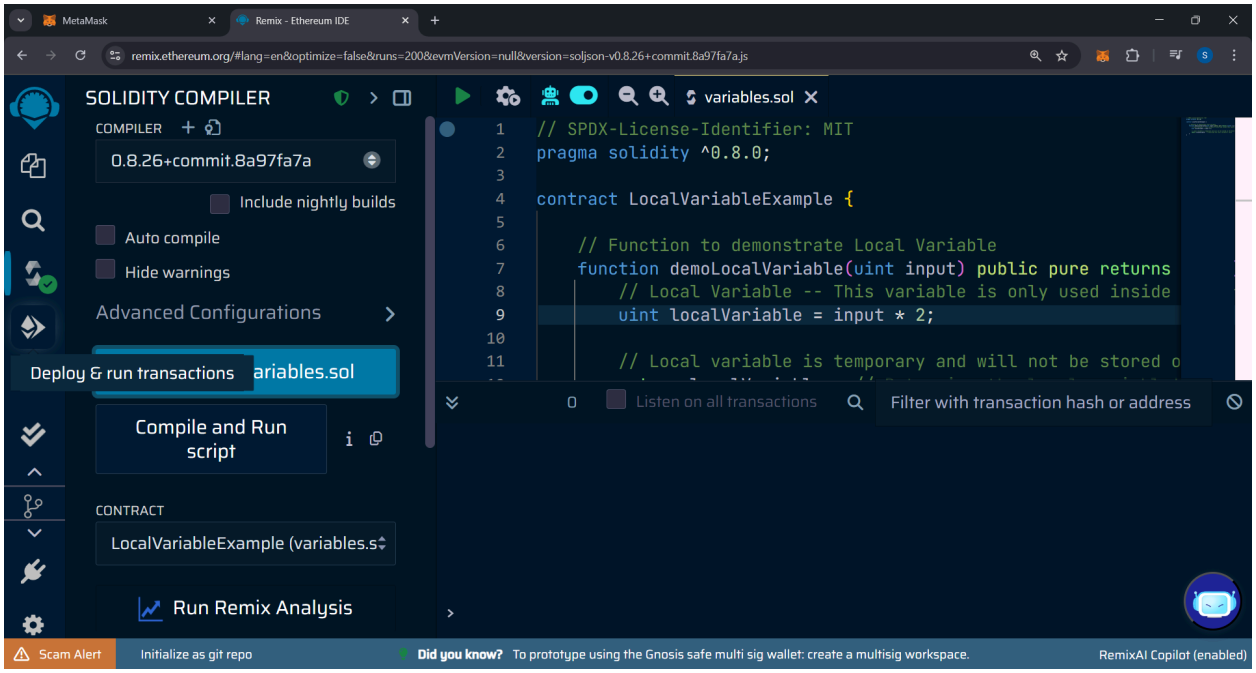
Go to Compiler from Icon menu

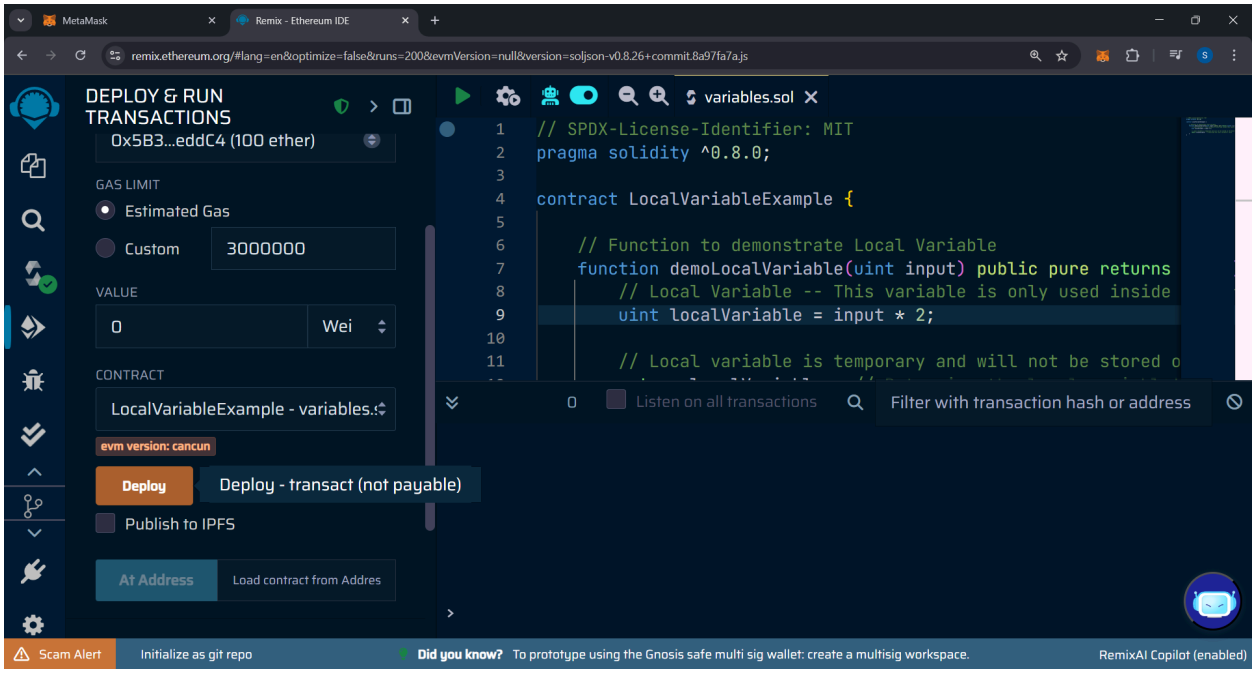


Click on Solidity Compile button

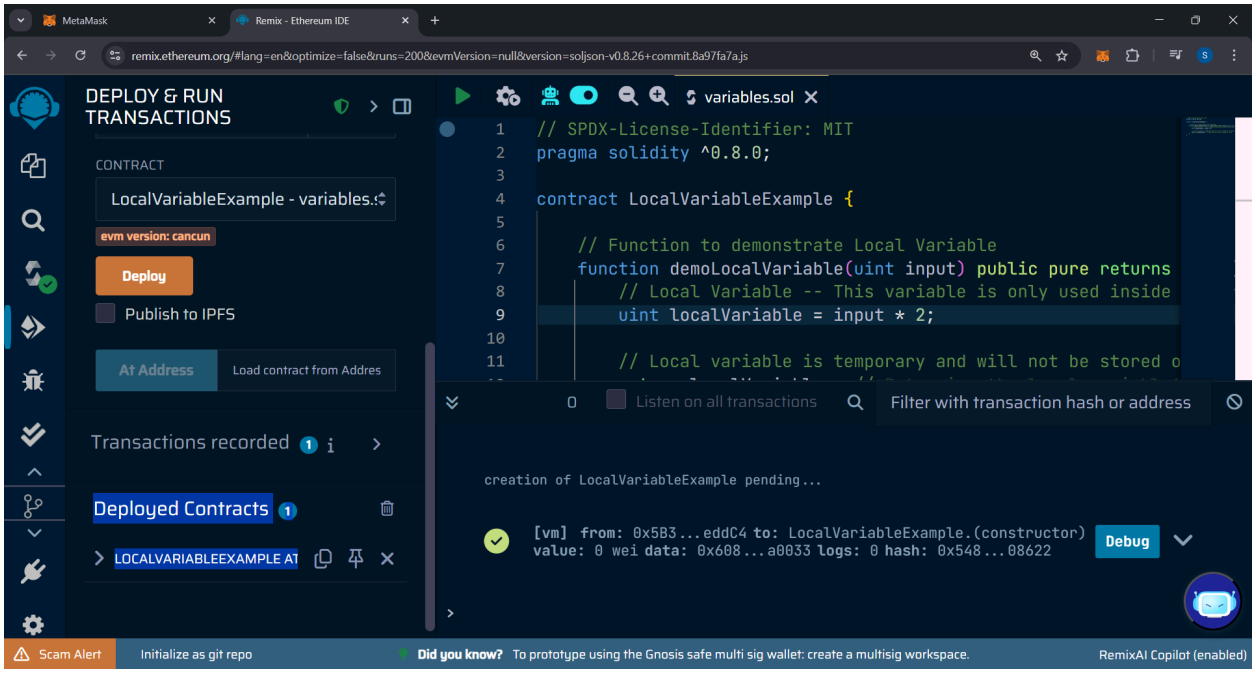


Go to deploy and run transaction tab and click on deploy button

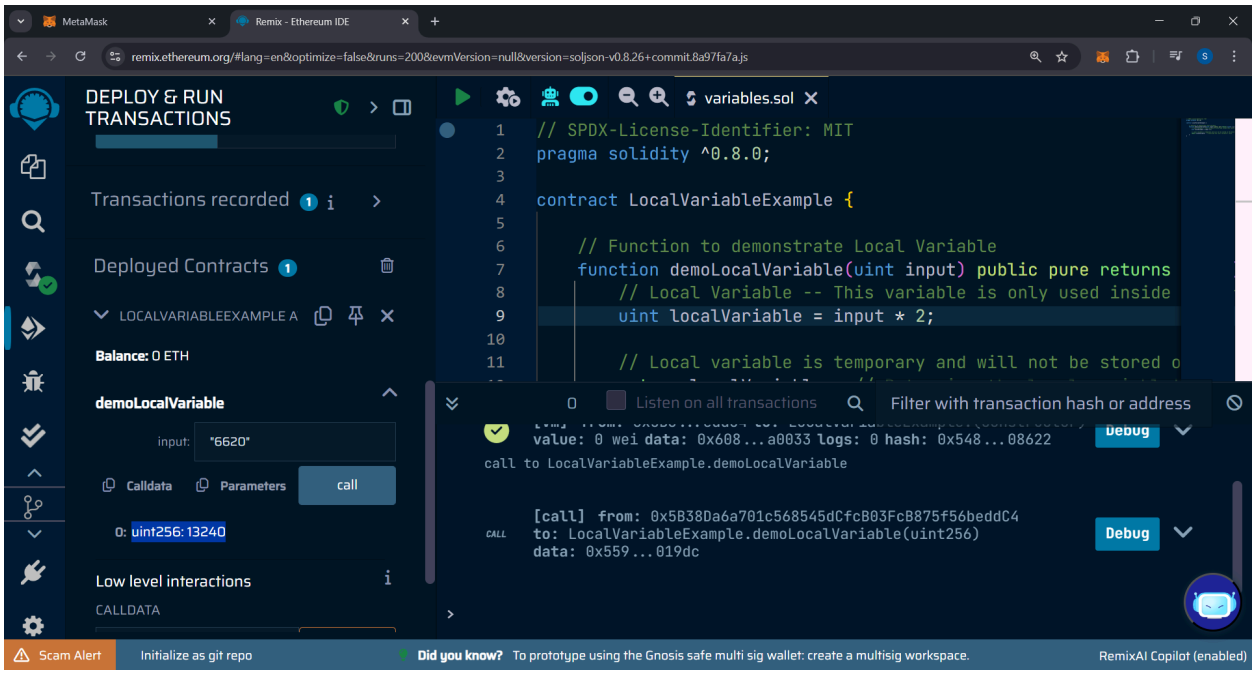




Your contract will be deployed just scroll down and will can see you contract name



Enter input number and check the output

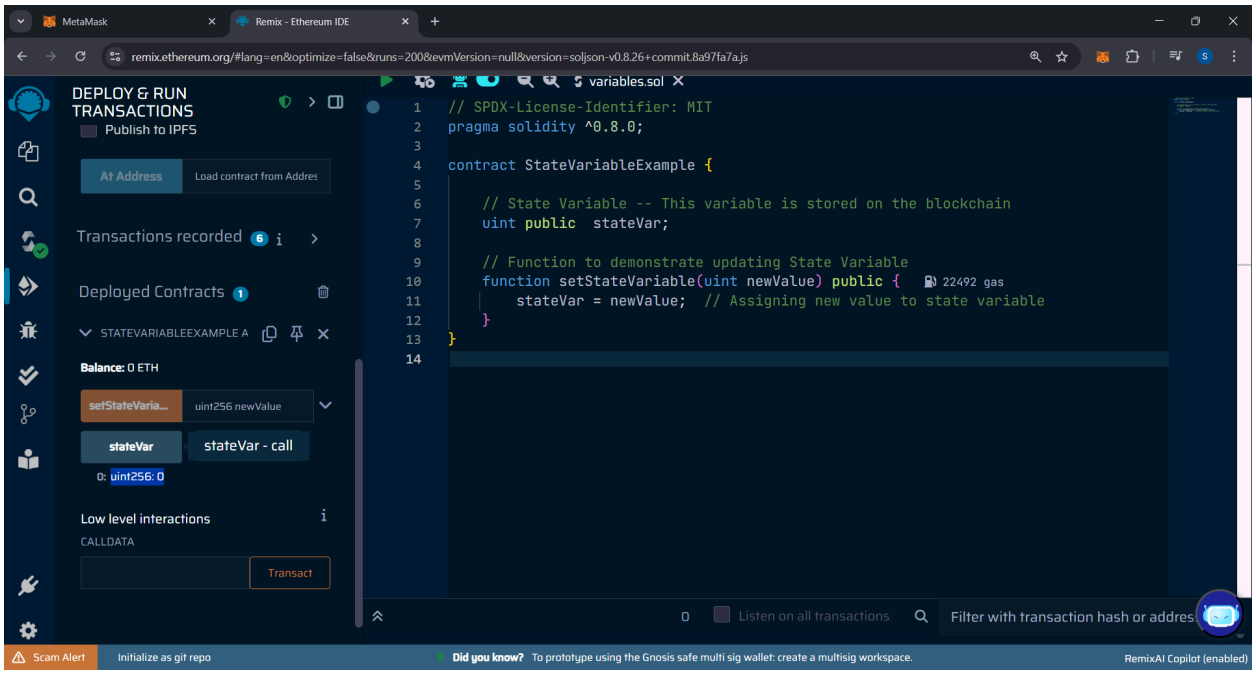


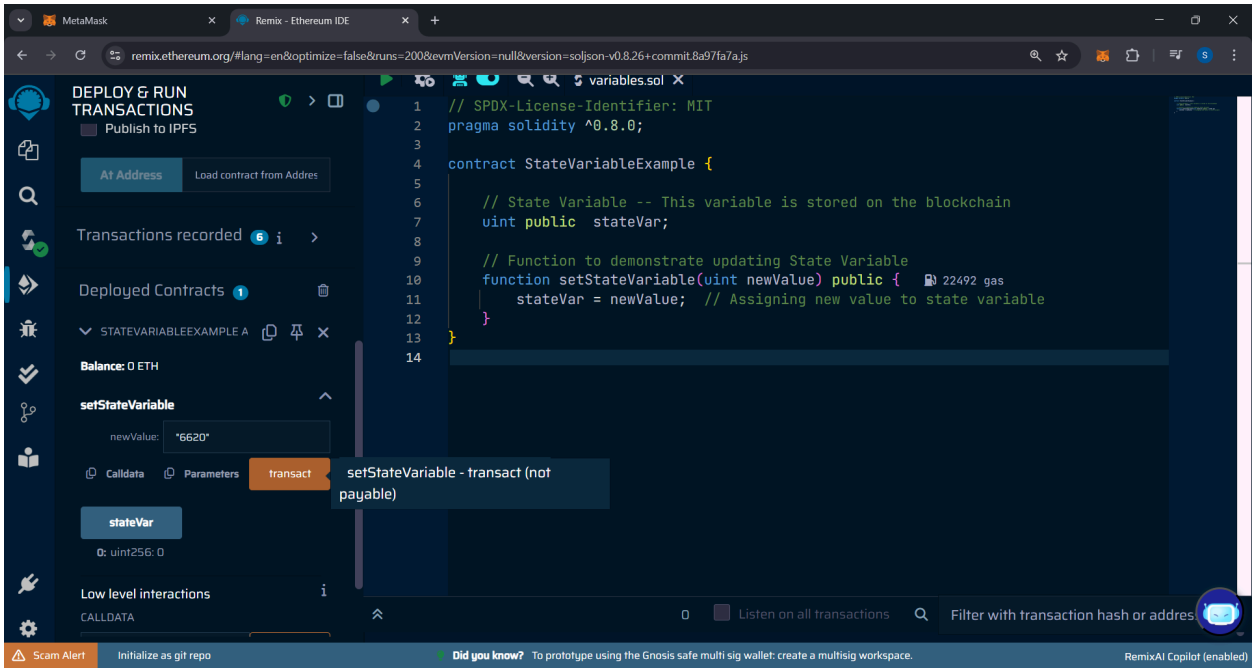
2. State Variable

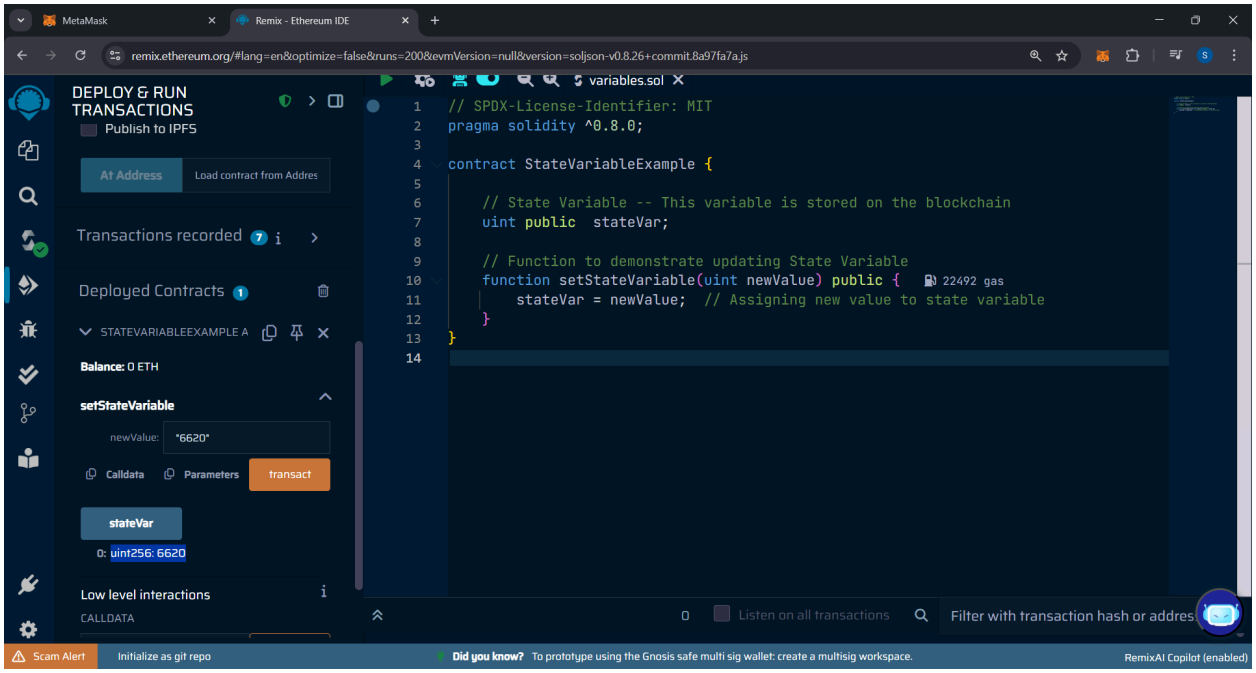
State variables, on the other hand, are declared outside any function within the contract and have a more extended lifetime. They persist across multiple function calls and endure as long as the contract is deployed.

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

contract StateVariableExample {

    // State Variable -- This variable is stored on the blockchain
    uint public  stateVar;

    // Function to demonstrate updating State Variable
    function setStateVariable(uint newValue) public {
        stateVar = newValue;  // Assigning new value to state variable
    }
}
```

Output



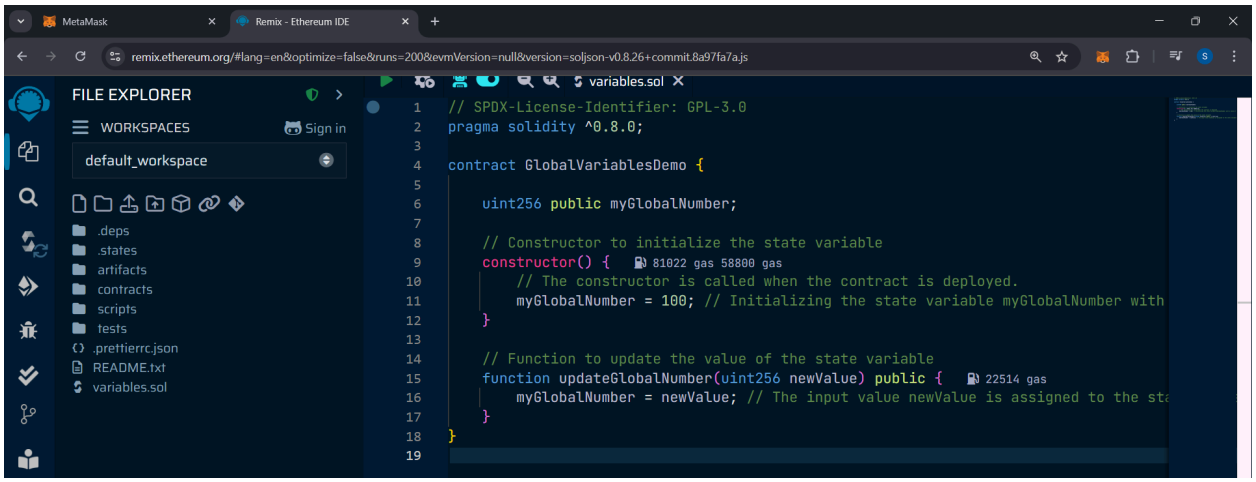

MetaMask
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
DEPLOY & RUN TRANSACTIONS
Publish to IPFS
At Address
Load contract from Addres
Transactions recorded 6
Deployed Contracts 1
STATEVARIABLEEXAMPLE A
Balance: 0 ETH
setStateVariable
newValue: "6620"
Calldata
Parameters
transact
setStateVariable - transact (not payable)
stateVar
0: uint256: 0
Low level interactions
CALLDATA
variables.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract StateVariableExample {
// State Variable -- This variable is stored on the blockchain
uint public stateVar;
// Function to demonstrate updating State Variable
function setStateVariable(uint newValue) public { 22492 gas
stateVar = newValue; // Assigning new value to state variable
}
}
Listen on all transactions
Filter with transaction hash or addres
Scam Alert
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)


MetaMask
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
DEPLOY & RUN TRANSACTIONS
Publish to IPFS
At Address
Load contract from Addres
Transactions recorded 7
Deployed Contracts 1
STATEVARIABLEEXAMPLE A
Balance: 0 ETH
setStateVariable
newValue: "6620"
Calldata
Parameters
transact
stateVar
0: uint256: 6620
Low level interactions
CALLDATA
variables.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract StateVariableExample {
// State Variable -- This variable is stored on the blockchain
uint public stateVar;
// Function to demonstrate updating State Variable
function setStateVariable(uint newValue) public { 22492 gas
stateVar = newValue; // Assigning new value to state variable
}
}
Listen on all transactions
Filter with transaction hash or addres
Scam Alert
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)

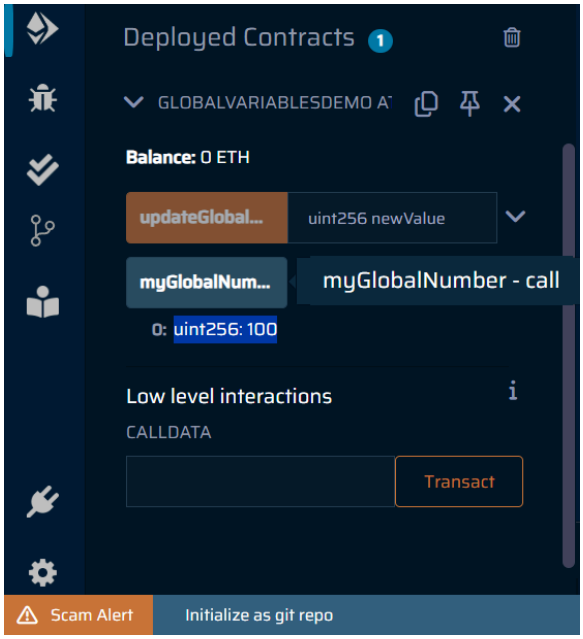
3. Global Variable

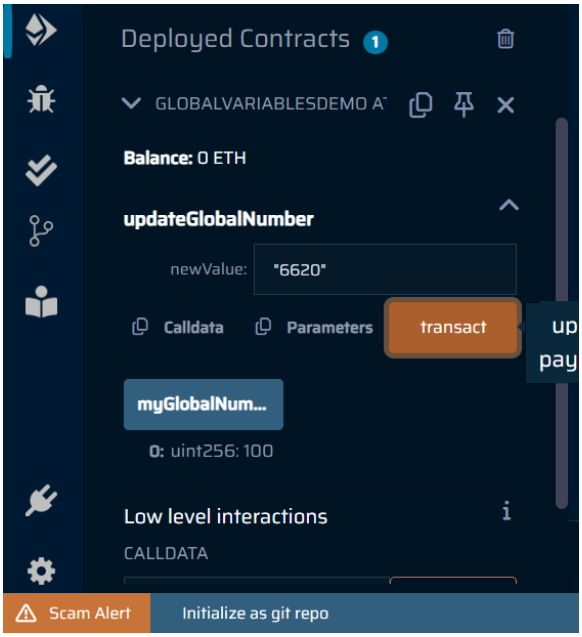
Function parameters are variables declared within a function's signature, serving as inputs to the function. Similar to local variables, function parameters have a limited scope and lifetime, existing only for the duration of the function's execution. They allow external values to be passed into a function, enabling it to perform computations or modifications based on the provided inputs.

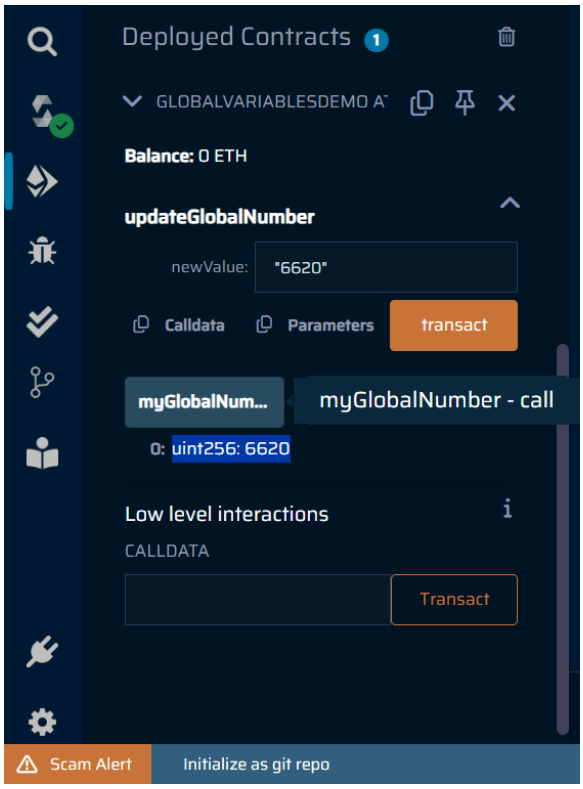
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract GlobalVariablesDemo {

    uint256 public myGlobalNumber;

    // Constructor to initialize the state variable
    constructor() {
        // The constructor is called when the contract is deployed.
        myGlobalNumber = 100; // Initializing the state variable myGlobalNumber with a value
of 0.
    }

    // Function to update the value of the state variable
    function updateGlobalNumber(uint256 newValue) public {
        myGlobalNumber = newValue; // The input value newValue is assigned to the state
variable myGlobalNumber.
    }
}
```



Output


Deployed Contracts 1
GLOBALVARIABLESDEMO A
Balance: 0 ETH
updateGlobal...
uint256 newValue
myGlobalNum...
myGlobalNumber - call
0: uint256: 100
Low level interactions
CALLDATA
Transact
Scam Alert
Initialize as git repo


Deployed Contracts 1
GLOBALVARIABLESDEMO A
Balance: 0 ETH
updateGlobalNumber
newValue:
"6620"
Calldata
Parameters
transact
myGlobalNum...
0: uint256: 100
Low level interactions
CALLDATA
Scam Alert
Initialize as git repo


Deployed Contracts 1
GLOBALVARIABLESDEMO A
Balance: 0 ETH
updateGlobalNumber
newValue:
"6620"
Calldata
Parameters
transact
myGlobalNum...
myGlobalNumber - call
0: uint256: 6620
Low level interactions
CALLDATA
Transact
Scam Alert
Initialize as git repo

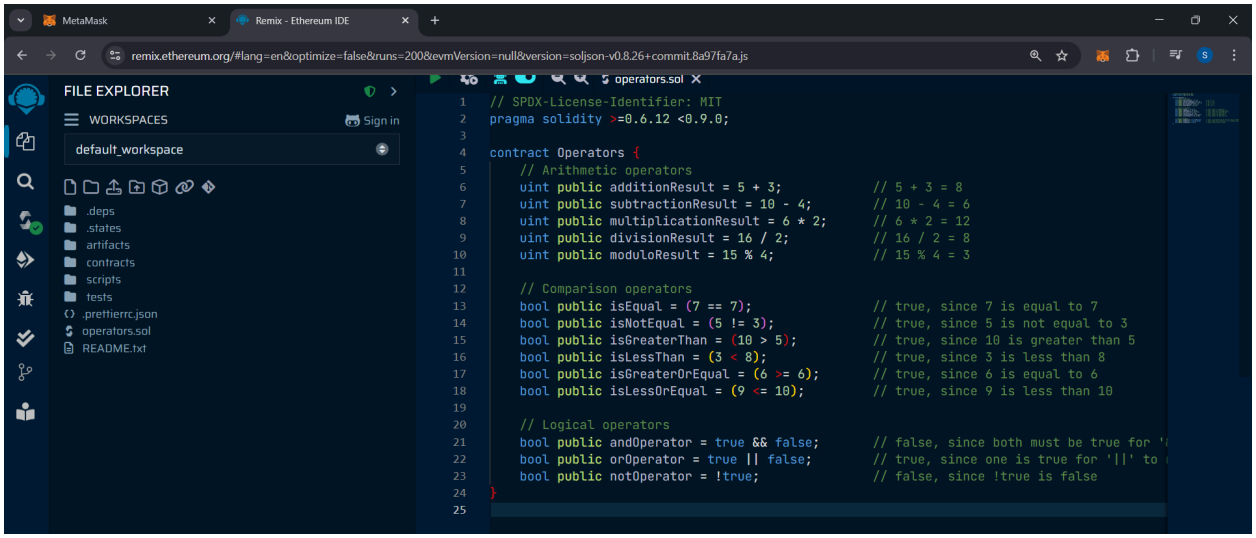
B. Operators

Solidity, Ethereum's smart contract language,

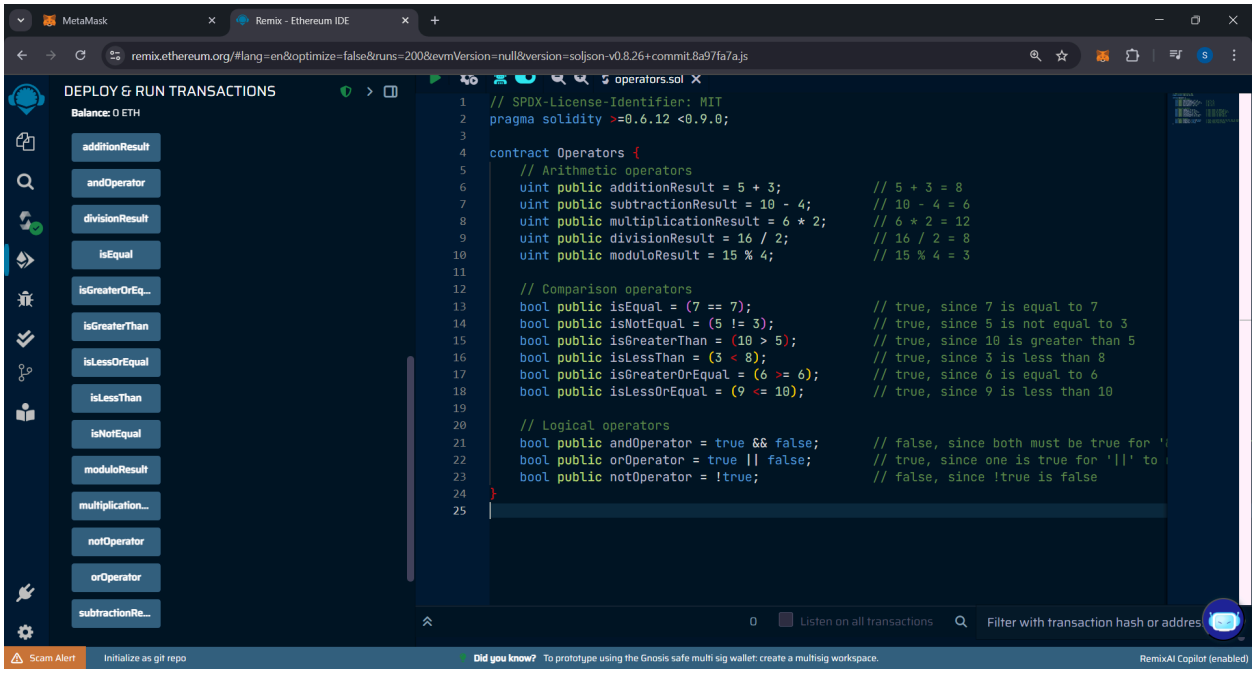
- features operators for arithmetic (e.g., +, -, *, /),
- comparisons (==, !=, <, >),
- logical (&&, ||, !),
- bitwise (&, |, ^, ~, <<, >>),
- assignment (=, +=, -=).

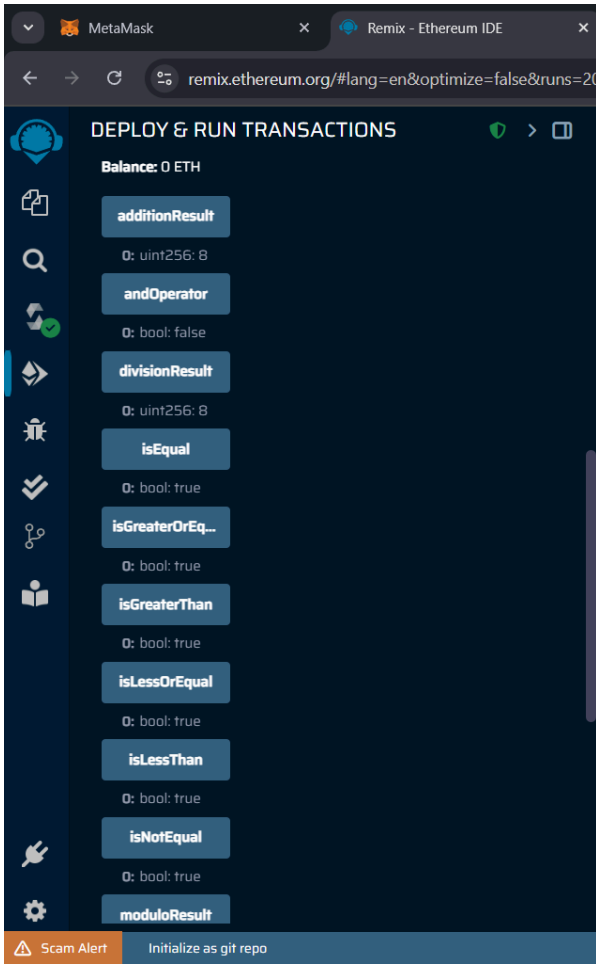
These operators empower developers to perform various computations, comparisons, and bit-level operations efficiently within smart contracts.

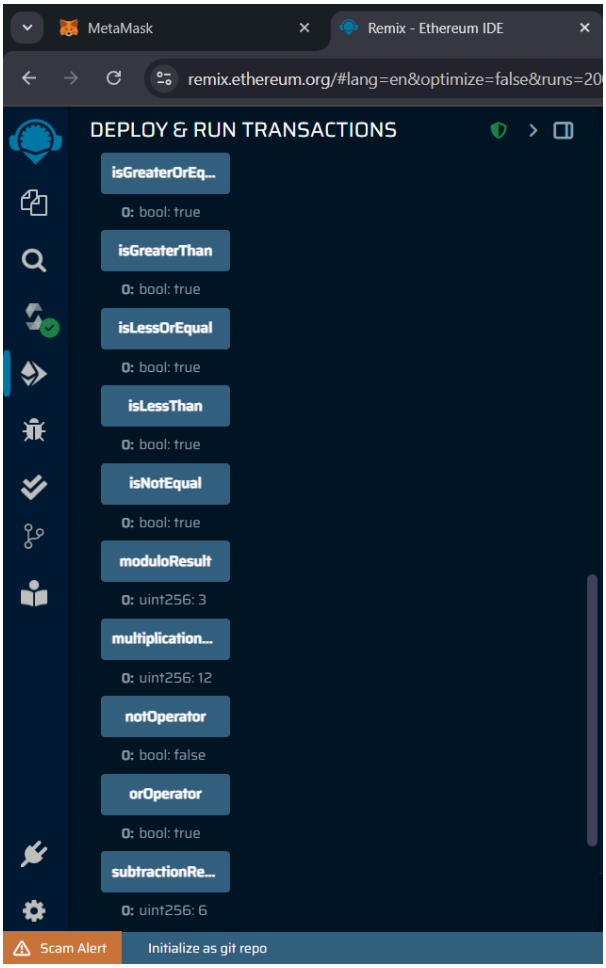
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;

contract Operators {
    // Arithmetic operators
    uint public additionResult = 5 + 3;            // 5 + 3 = 8
    uint public subtractionResult = 10 - 4;        // 10 - 4 = 6
    uint public multiplicationResult = 6 * 2;      // 6 * 2 = 12
    uint public divisionResult = 16 / 2;           // 16 / 2 = 8
    uint public moduloResult = 15 % 4;             // 15 % 4 = 3

    // Comparison operators
    bool public isEqual = (7 == 7);                // true, since 7 is equal to 7
    bool public isNotEqual = (5 != 3);             // true, since 5 is not equal to 3
    bool public isGreaterThan = (10 > 5);          // true, since 10 is greater than 5
    bool public isLessThan = (3 < 8);              // true, since 3 is less than 8
    bool public isGreaterOrEqual = (6 >= 6);       // true, since 6 is equal to 6
    bool public isLessOrEqual = (9 <= 10);         // true, since 9 is less than 10

    // Logical operators
    bool public andOperator = true && false;       // false, since both must be true for '&&' to
return true
    bool public orOperator = true || false;        // true, since one is true for '||' to return true
    bool public notOperator = !true;               // false, since !true is false
}
```


MetaMask
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
FILE EXPLORER
WORKSPACES
Sign in
default_workspace
.deps
.states
artifacts
contracts
scripts
tests
.prettierrc.json
operators.sol
README.txt
operators.sol
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;
contract Operators {
// Arithmetic operators
uint public additionResult = 5 + 3; // 5 + 3 = 8
uint public subtractionResult = 10 - 4; // 10 - 4 = 6
uint public multiplicationResult = 6 * 2; // 6 * 2 = 12
uint public divisionResult = 16 / 2; // 16 / 2 = 8
uint public moduloResult = 15 % 4; // 15 % 4 = 3
// Comparison operators
bool public isEqual = (7 == 7); // true, since 7 is equal to 7
bool public isNotEqual = (5 != 3); // true, since 5 is not equal to 3
bool public isGreaterThan = (10 > 5); // true, since 10 is greater than 5
bool public isLessThan = (3 < 8); // true, since 3 is less than 8
bool public isGreaterOrEqual = (6 >= 6); // true, since 6 is equal to 6
bool public isLessOrEqual = (9 <= 10); // true, since 9 is less than 10
// Logical operators
bool public andOperator = true && false; // false, since both must be true for
bool public orOperator = true || false; // true, since one is true for '||' to
bool public notOperator = !true; // false, since !true is false
}

Output


MetaMask
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
DEPLOY & RUN TRANSACTIONS
Balance: 0 ETH
additionResult
andOperator
divisionResult
isEqual
isGreaterOrEq...
isGreaterThan
isLessOrEqual
isLessThan
isNotEqual
moduloResult
multiplication...
notOperator
orOperator
subtractionRe...
operators.sol
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;
contract Operators {
// Arithmetic operators
uint public additionResult = 5 + 3; // 5 + 3 = 8
uint public subtractionResult = 10 - 4; // 10 - 4 = 6
uint public multiplicationResult = 6 * 2; // 6 * 2 = 12
uint public divisionResult = 16 / 2; // 16 / 2 = 8
uint public moduloResult = 15 % 4; // 15 % 4 = 3
// Comparison operators
bool public isEqual = (7 == 7); // true, since 7 is equal to 7
bool public isNotEqual = (5 != 3); // true, since 5 is not equal to 3
bool public isGreaterThan = (10 > 5); // true, since 10 is greater than 5
bool public isLessThan = (3 < 8); // true, since 3 is less than 8
bool public isGreaterOrEqual = (6 >= 6); // true, since 6 is equal to 6
bool public isLessOrEqual = (9 <= 10); // true, since 9 is less than 10
// Logical operators
bool public andOperator = true && false; // false, since both must be true for
bool public orOperator = true || false; // true, since one is true for '||' to
bool public notOperator = !true; // false, since !true is false
}
0
Listen on all transactions
Filter with transaction hash or addres
Scam Alert
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)

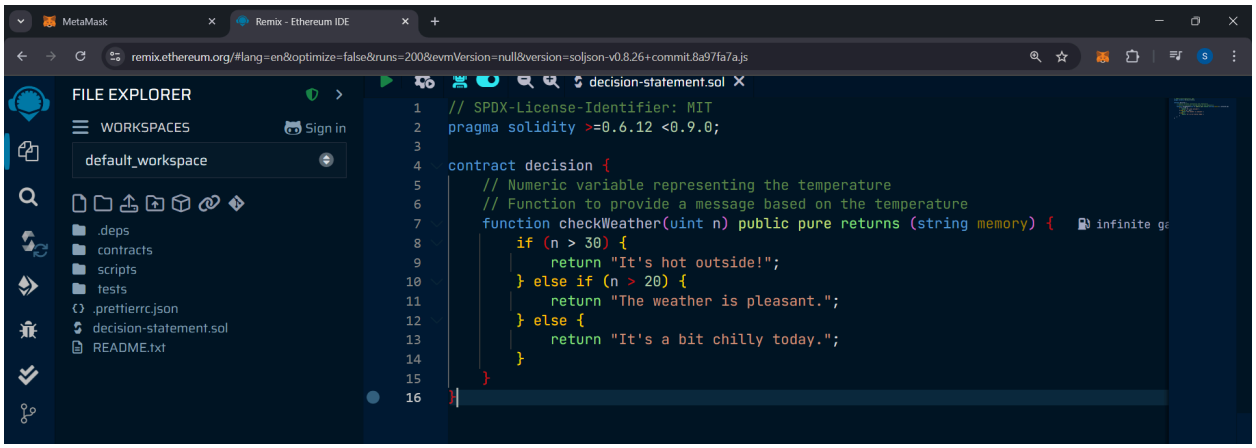

MetaMask
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=20
DEPLOY & RUN TRANSACTIONS
Balance: 0 ETH
additionResult
0: uint256: 8
andOperator
0: bool: false
divisionResult
0: uint256: 8
isEqual
0: bool: true
isGreaterOrEq...
0: bool: true
isGreaterThan
0: bool: true
isLessOrEqual
0: bool: true
isLessThan
0: bool: true
isNotEqual
0: bool: true
moduloResult
Scam Alert
Initialize as git repo


MetaMask
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=20
DEPLOY & RUN TRANSACTIONS
isGreaterOrEq...
0: bool: true
isGreaterThan
0: bool: true
isLessOrEqual
0: bool: true
isLessThan
0: bool: true
isNotEqual
0: bool: true
moduloResult
0: uint256: 3
multiplication...
0: uint256: 12
notOperator
0: bool: false
orOperator
0: bool: true
subtractionRe...
0: uint256: 6
Scam Alert
Initialize as git repo

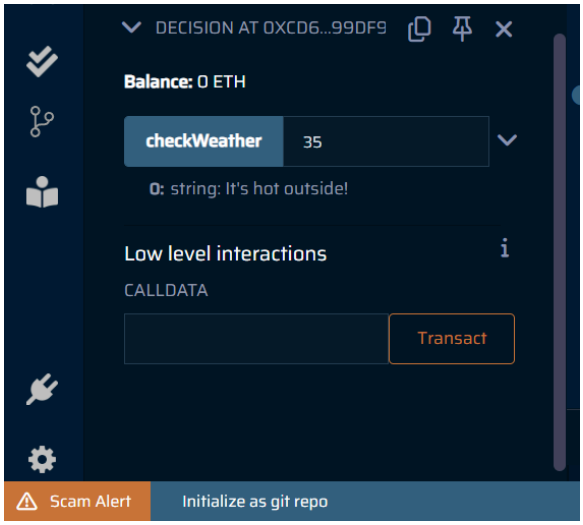
C. Decision Making - if, if else, if else if

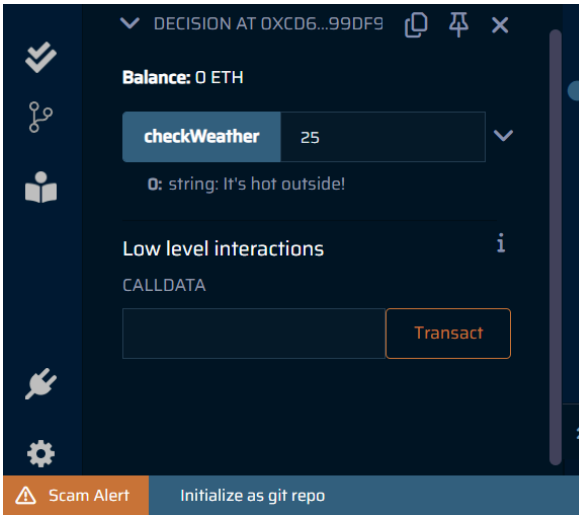
Solidity if, if-else, and if-else if statements enable condition-based decision-making. The if statement executes code when a condition is true, if-else adds an alternative for false conditions, and if-else if handles multiple sequential conditions. These constructs empower concise and adaptable logic in Ethereum smart contracts.

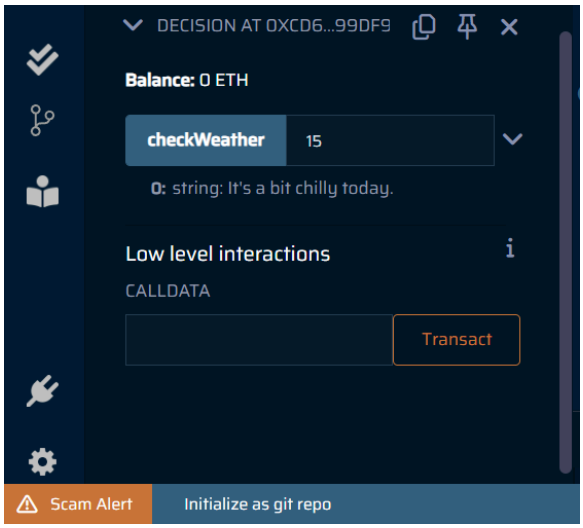
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;

contract decision {
   // Numeric variable representing the temperature
   // Function to provide a message based on the temperature
   function checkWeather(uint n) public pure returns (string memory) {
      if (n > 30) {
         return "It's hot outside!";
      } else if (n > 20) {
         return "The weather is pleasant.";
      } else {
         return "It's a bit chilly today.";
      }
   }
}
```



Output


DECISION AT 0XCD6...99DF9
Balance: 0 ETH
checkWeather
35
0: string: It's hot outside!
Low level interactions
CALLDATA
Transact
Scam Alert
Initialize as git repo


DECISION AT 0XCD6...99DF9
Balance: 0 ETH
checkWeather
25
0: string: It's hot outside!
Low level interactions
CALLDATA
Transact
Scam Alert
Initialize as git repo

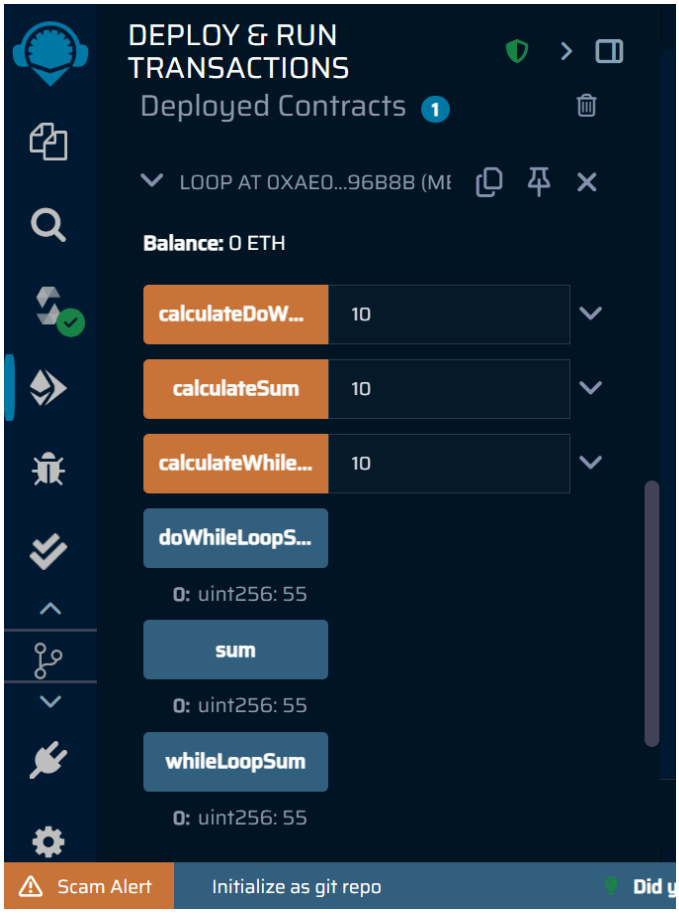

DECISION AT 0XCD6...99DF9
Balance: 0 ETH
checkWeather
15
0: string: It's a bit chilly today.
Low level interactions
CALLDATA
Transact
Scam Alert
Initialize as git repo

D. Loops - for, while, and do while

In Solidity, loops streamline repetitive tasks. The `for` loop efficiently iterates over a specific range. The `while` loop repeats as long as a given condition holds true. The `do-while` loop ensures a block of code runs at least once before checking the condition. These loops empower concise and effective iteration in Solidity smart contracts, enhancing efficiency and functionality.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;

contract Loop {

    // State variables to store the results of the sums
    uint public sum;
    uint public whileLoopSum;
    uint public doWhileLoopSum;

    // Function to calculate the sum of numbers from 1 to n using a for loop
    function calculateSum(uint n) public returns (uint) {
        // Initialize the sum variable
        sum = 0;

        // Loop to calculate the sum of numbers from 1 to n
        for (uint i = 1; i <= n; i++) {
            sum += i;
        }

        return sum;
    }

    // Function to calculate the sum of numbers from 1 to n using a while loop
    function calculateWhileLoopSum(uint n) public returns (uint) {
        // Initialize variables
        uint i = 1;
        whileLoopSum = 0;

        // Loop to calculate the sum using a while loop
        while (i <= n) {
            whileLoopSum += i;
            i++;
        }

        return whileLoopSum;
    }
```

```
    // Function to calculate the sum of numbers from 1 to n using a do-while loop
    function calculateDoWhileLoopSum(uint n) public returns (uint) {
        // Initialize variables
        uint i = 1;
        doWhileLoopSum = 0;

        // Loop to calculate the sum using a do-while loop
        do {
            doWhileLoopSum += i;
            i++;
        } while (i <= n);

        return doWhileLoopSum;
    }
}
```



Practical 7: Write the solidity Code and demonstrate the following:

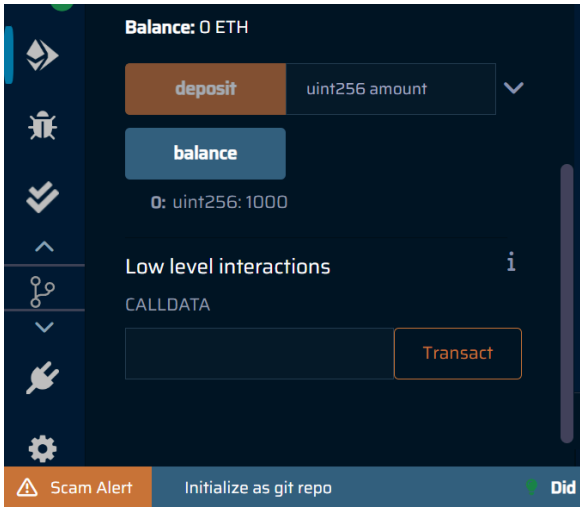
a. Functions
b. View Functions
c. Pure Functions
d. Cryptographic Function

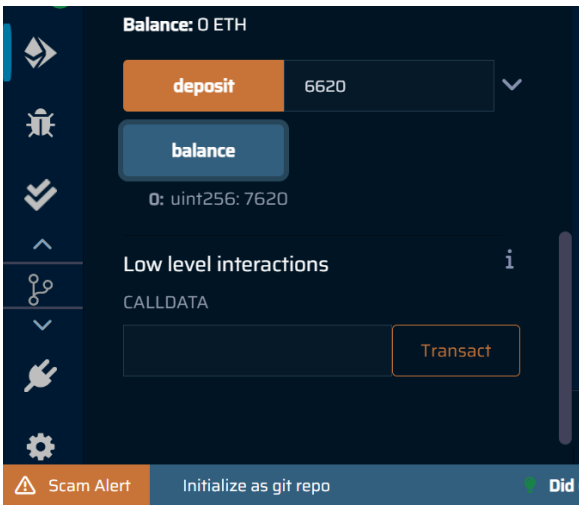
A. Functions (Non-View and Non-Pure)

- A regular function (without view or pure modifiers) can both read and modify the blockchain state.
- Regular functions are used when you need to update or change something on the blockchain (e.g., transferring tokens, updating balances, etc.).

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

contract Example {
    uint public balance = 1000;

    // Regular function: This modifies the state
    function deposit(uint amount) public {
        balance += amount;  // This changes the state
    }
}
```

Output



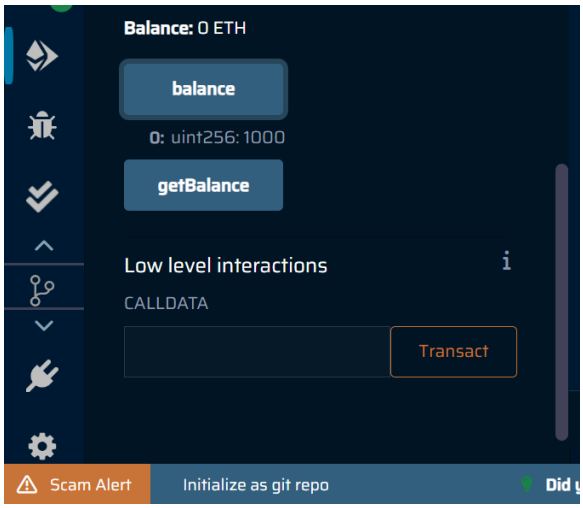


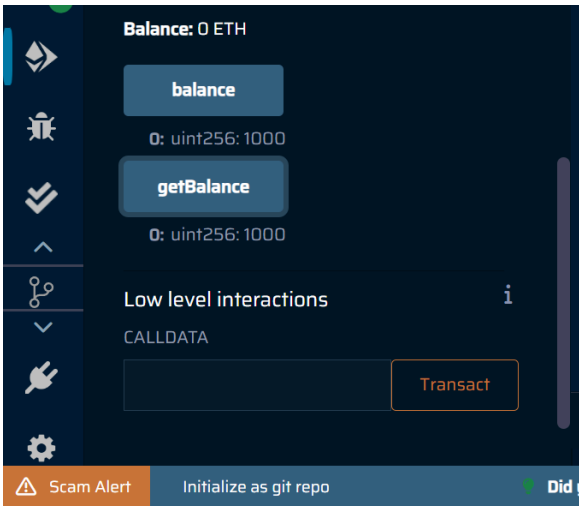
B. View Functions

- A view function is one that does not modify the blockchain state. It can only read the state variables and return values, but it cannot change anything on the blockchain.
- A view function that returns the current value of a state variable without modifying it.

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

contract Example {
    uint public balance = 1000;

    // View function: This reads the state, but does not modify it
    function getBalance() public view returns (uint) {
        return balance; // This function only reads the state
    }
}
```

Ouptut



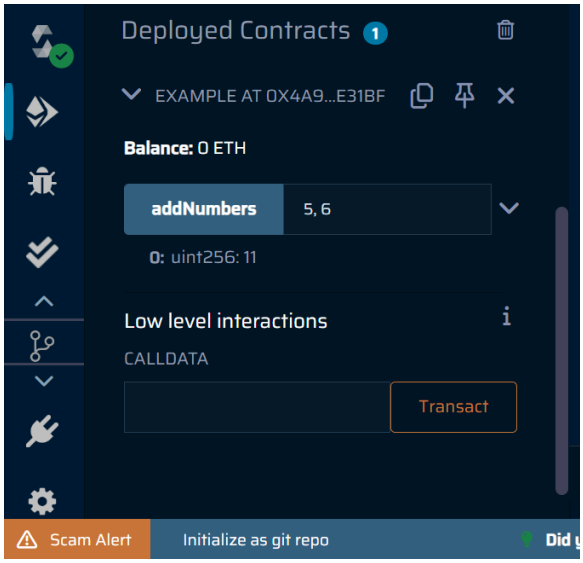


C. Pure Functions

A pure function is a function that neither reads nor modifies the blockchain state. It operates solely on the provided arguments and returns a result based on them.
Use pure functions when you need to perform computations or transformations based solely on the function arguments, without accessing or modifying any state.

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

contract Example {

    // Pure function: This does not read or modify the state
    function addNumbers(uint a, uint b) public pure returns (uint) {
        return a + b;  // This function just performs a calculation
    }
}
```

Output

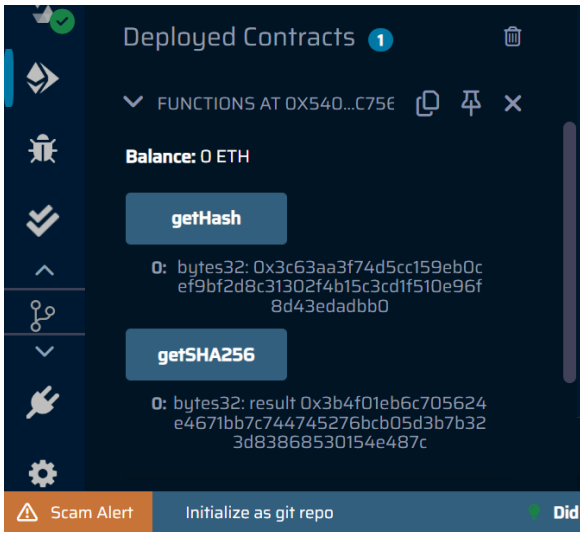


D. Cryptographic Function

Cryptographic functions in Solidity typically involve operations such as hashing or verifying digital signatures. These are built-in functions that perform cryptographic operations and are often used for security purposes (e.g., hashing data, signing messages, etc.).

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

contract Functions {
    // Function to hash data using keccak256
    function getSHA256() public pure returns(bytes32 result){
        return sha256("6620");
    }

    // Function to hash data using keccak256
    function getHash() public pure returns (bytes32) {
        return keccak256("6620");
    }
}
```

Output



Practical 8: Write the solidity Code and demonstrate the following:

a. Contracts
b. Inheritance
c. Constructors
d. Interfaces

A. Contracts

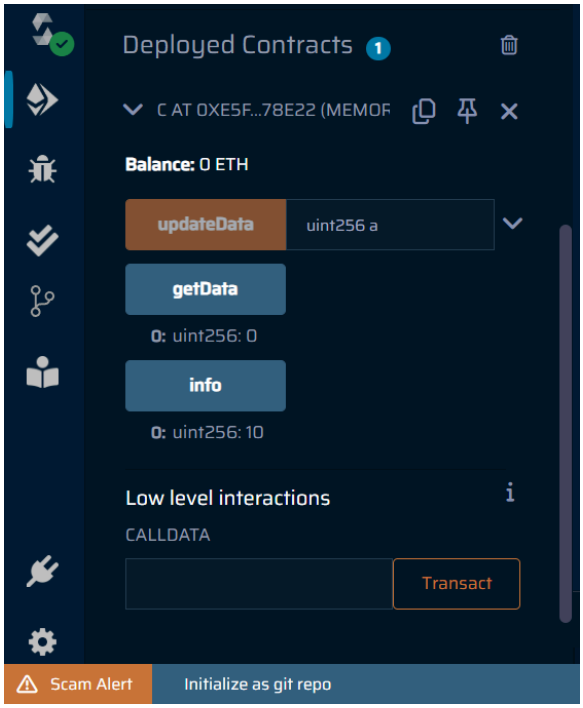
A Contract in Solidity is similar to a Class in C++. Contract has the following properties.

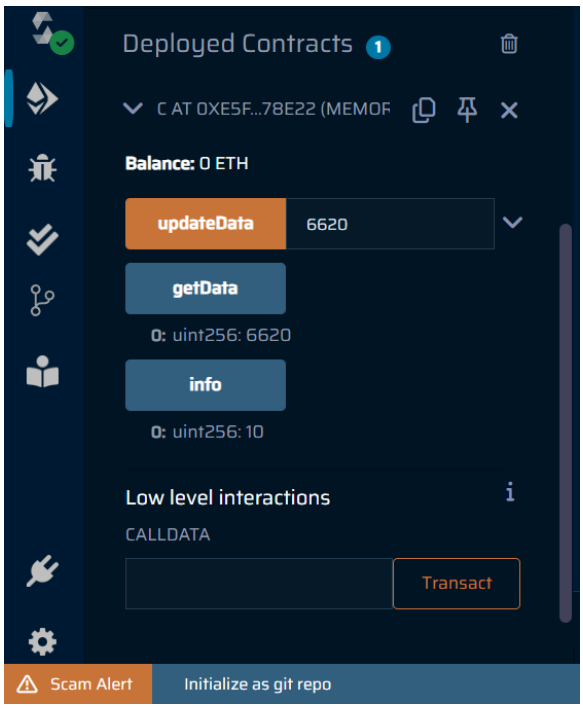
- Constructor − A special function declared with the constructor keyword which will be executed once per contract and is invoked when a contract is created.
- State Variables − Variables per Contract to store the state of the contract.
- Functions − Functions per Contract which can modify the state variables to alter the state of a contract.

B. Inheritance

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

contract C {
   uint private data; // Private state variable
   uint public info; // Public state variable

   // Constructor
   constructor() { info = 10;  }

   // Private function
   function increment(uint a) private pure returns (uint) {
      return a + 1;
   }

   // Public function
   function updateData(uint a) public {
      data = a;
   }

   function getData() public view virtual returns (uint) {
      return data;
```

```
    }

    function compute(uint a, uint b) internal pure returns (uint) {
        return a + b;
    }
}

// Derived Contract
contract E is C {
    uint private result;
    C private c;

    constructor() {
        c = new C();
    }

    function getComputedResult() public {
        result = compute(3, 5);
    }

    function getResult() public view returns (uint) {
        return result;
    }

    function getData() public view override returns (uint) {
        return c.getData();
    }
}
```


Deployed Contracts 1
C AT 0XE5F...78E22 (MEMOF
Balance: 0 ETH
updateData
uint256 a
getData
0: uint256: 0
info
0: uint256: 10
Low level interactions
CALLDATA
Transact
Scam Alert
Initialize as git repo


Deployed Contracts 1
C AT 0XE5F...78E22 (MEMOF
Balance: 0 ETH
updateData
6620
getData
0: uint256: 6620
info
0: uint256: 10
Low level interactions
CALLDATA
Transact
Scam Alert
Initialize as git repo

C. Constructors

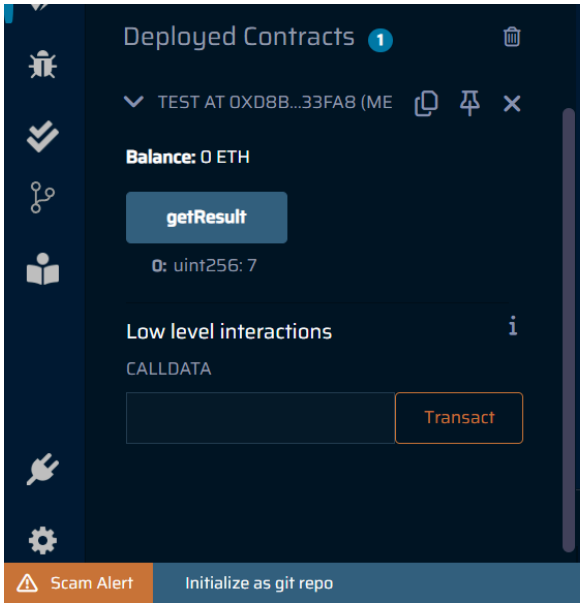
- Constructor is a special function declared using the constructor keyword. It is an optional function and is used to initialize state variables of a contract. Following are the key characteristics of a constructor.
- A contract can have only one constructor.
- A constructor code is executed once when a contract is created and it is used to initialize contract state.
- A constructor can be either public or internal.
- An internal constructor marks the contract as abstract.
- In case, no constructor is defined, a default constructor is present in the contract.

D. Interfaces

- Interfaces are similar to abstract contracts and are created using interface keyword. Following are the key characteristics of an interface.
- Interfaces can not have any function with implementation.
- Functions of an interface can be only of type external.
- Interfaces can not have constructors and state variables.

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

interface Calculator { // Define the Calculator interface
    function getResult() external pure returns (uint);
}

// Implement the Calculator interface in the Test contract
contract Test is Calculator {
    constructor() {}

    // Implementing the getResult function from the Calculator interface
    function getResult() external pure override returns (uint) {
        uint a = 5;
        uint b = 2;
        return a + b;
    }
}
```

Output

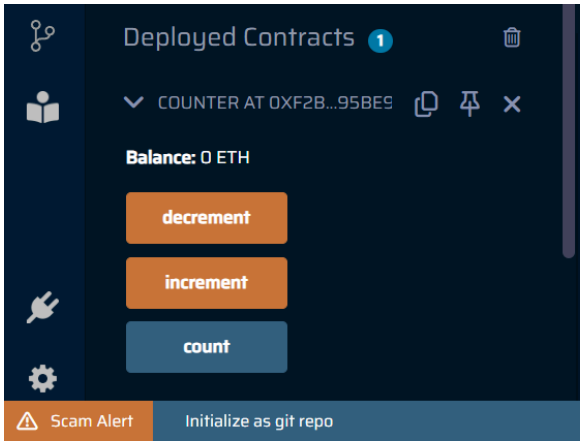


Practical 9: Program a Simple contract that can get, increment and decrement the count store in the contract.

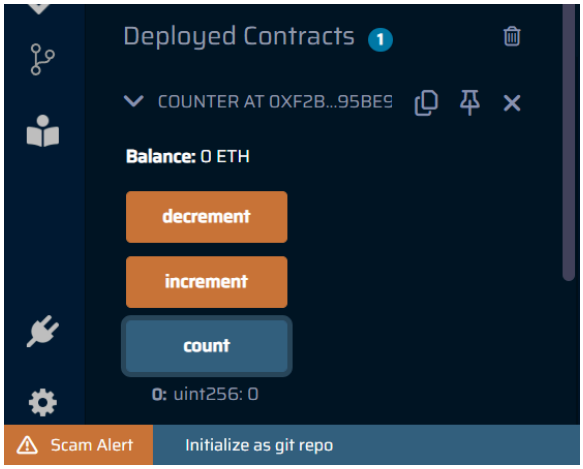
The `Counter` contract allows you to get, increment, and decrement a stored count. The `getCount` function retrieves the count, while `increment` and `decrement` adjust it by 1. The contract emits a `CountUpdated` event for transparency.

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

contract Counter {
    uint public count;

    // Function to increment count by 1
    function increment() public {
        count += 1;
    }

    // Function to decrement count by 1
    function decrement() public {
        // Prevent decrementing if count is 0
        require(count > 0, "Count cannot be less than 0");
        count -= 1;
    }
}
```

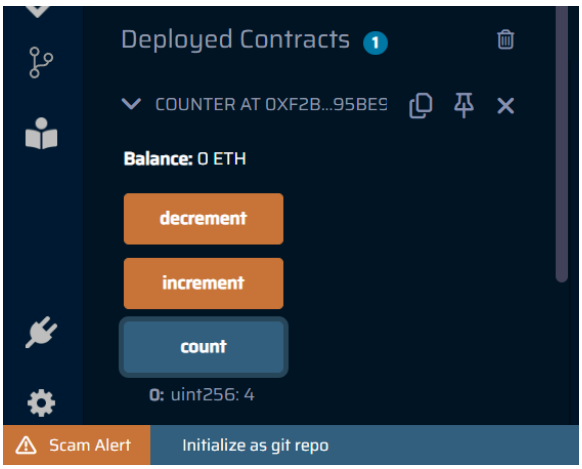
Output



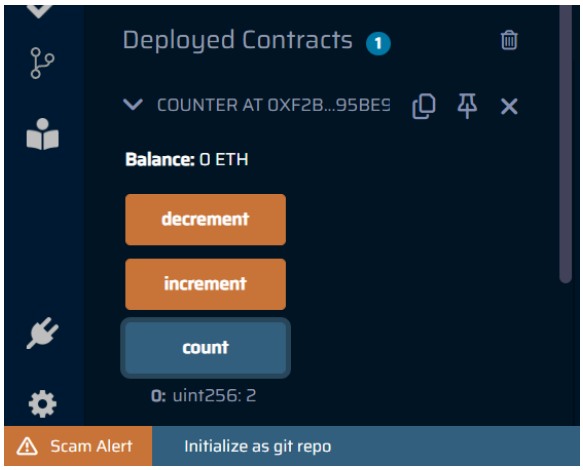
Getting count



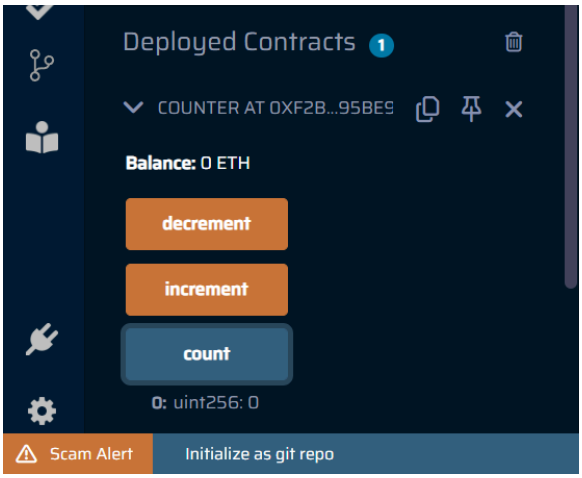
Incrementing 4 times

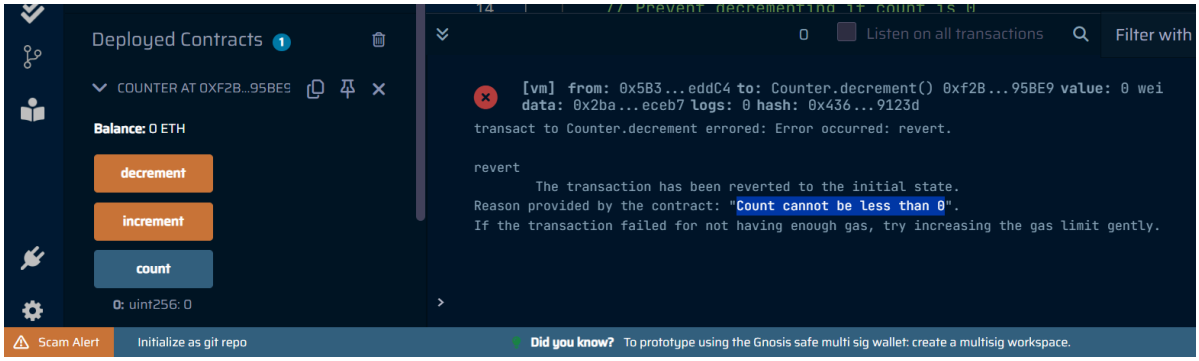


Decrementing twice



What when count is 0 and we decrement



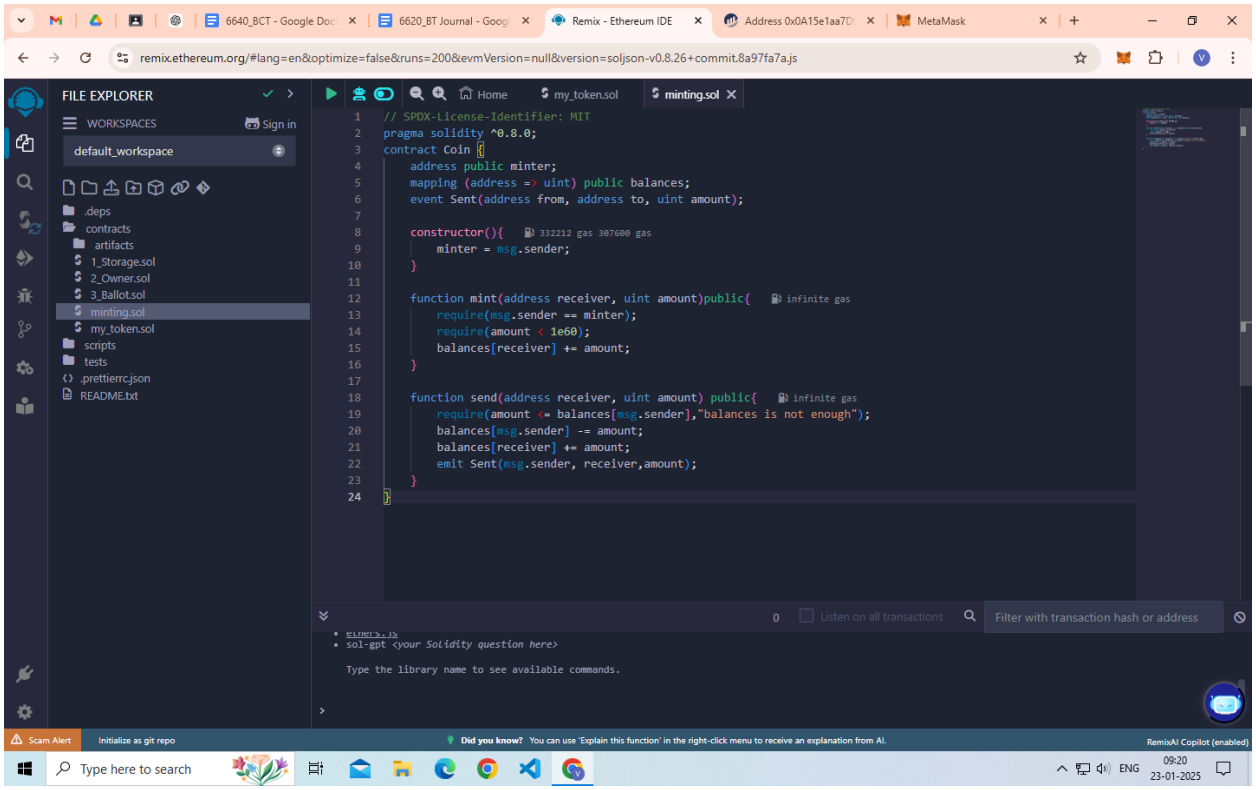


Practical 10: Write a simple smart contract to calculate the 'number of ethers' for the transaction of gas limit for the given scenario.

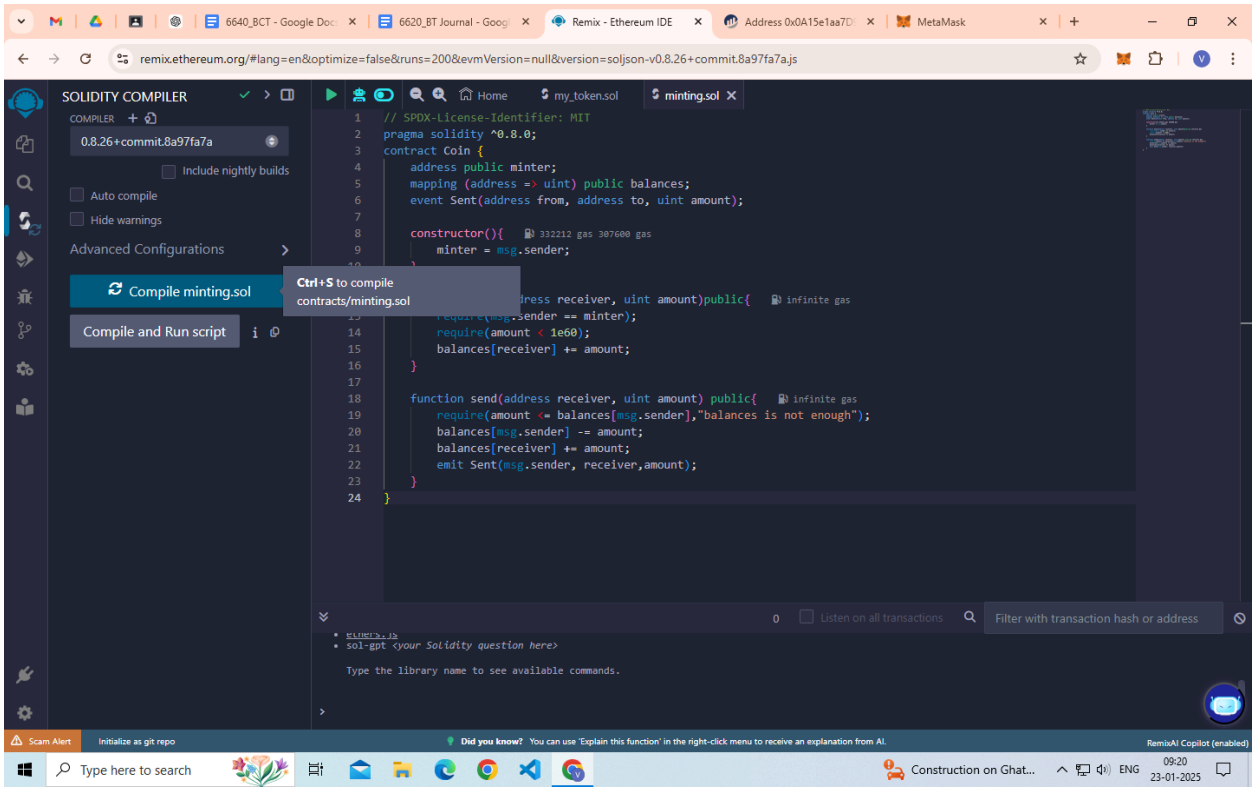
a. Mint Your Own Coins and Make Simple Transactions With Solidity
b. Creating ERC -20 Token

A. Mint Your Own Coins and Make Simple Transactions With Solidity

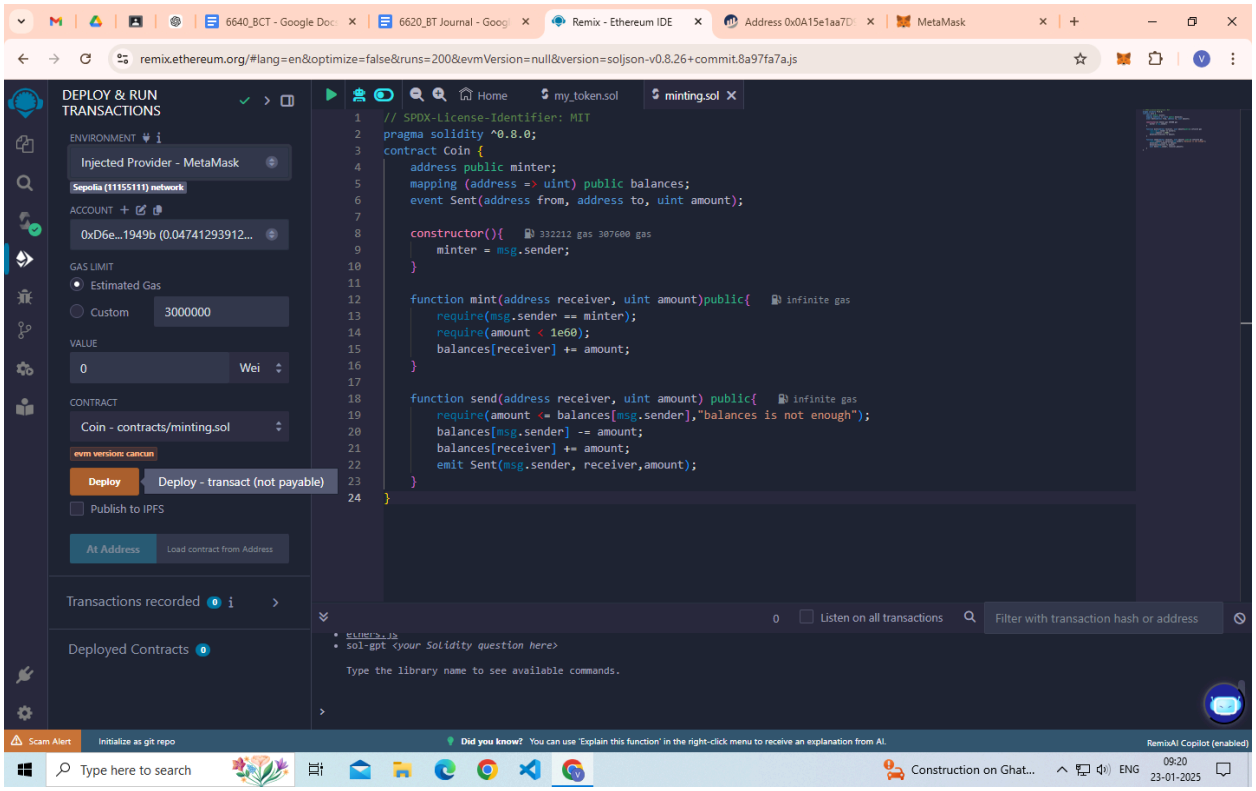
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Coin {
    address public minter;
    mapping (address => uint) public balances;
    event Sent(address from, address to, uint amount);

    constructor(){
        minter = msg.sender;
    }

    function mint(address receiver, uint amount)public{
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public{
        require(amount <= balances[msg.sender],"balances is not enough");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver,amount);
    }
}
```


FILE EXPLORER
WORKSPACES
default_workspace
.deps
contracts
artifacts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
minting.sol
my_token.sol
scripts
tests
.prettierrc.json
README.txt
Home
my_token.sol
minting.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Coin {
address public minter;
mapping (address => uint) public balances;
event Sent(address from, address to, uint amount);
constructor(){
minter = msg.sender;
}
function mint(address receiver, uint amount)public{
require(msg.sender == minter);
require(amount < 1e60);
balances[receiver] += amount;
}
function send(address receiver, uint amount) public{
require(amount <= balances[msg.sender],"balances is not enough");
balances[msg.sender] -= amount;
balances[receiver] += amount;
emit Sent(msg.sender, receiver,amount);
}
}
Filter with transaction hash or address
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
RemixAI Copilot (enabled)
Type here to search
ENG
09:20
23-01-2025

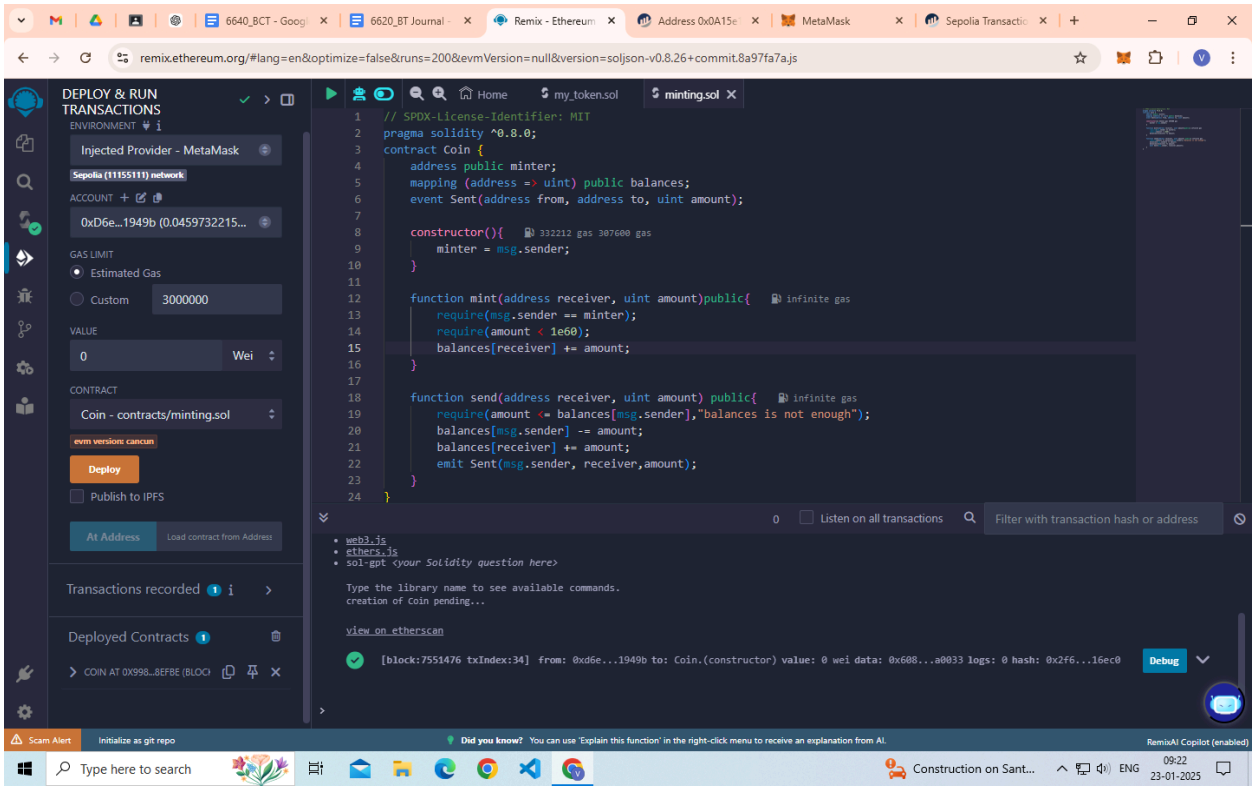
Compile


SOLIDITY COMPILER
COMPILER
0.8.26+commit.8a97fa7a
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile minting.sol
Ctrl+S to compile contracts/minting.sol
Compile and Run script
Home
my_token.sol
minting.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Coin {
address public minter;
mapping (address => uint) public balances;
event Sent(address from, address to, uint amount);
constructor(){
minter = msg.sender;
}
function send(address receiver, uint amount) public{
require(amount <= balances[msg.sender],"balances is not enough");
balances[msg.sender] -= amount;
balances[receiver] += amount;
emit Sent(msg.sender, receiver,amount);
}
}
Filter with transaction hash or address
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
RemixAI Copilot (enabled)
Type here to search
Construction on Ghat...
ENG
09:20
23-01-2025

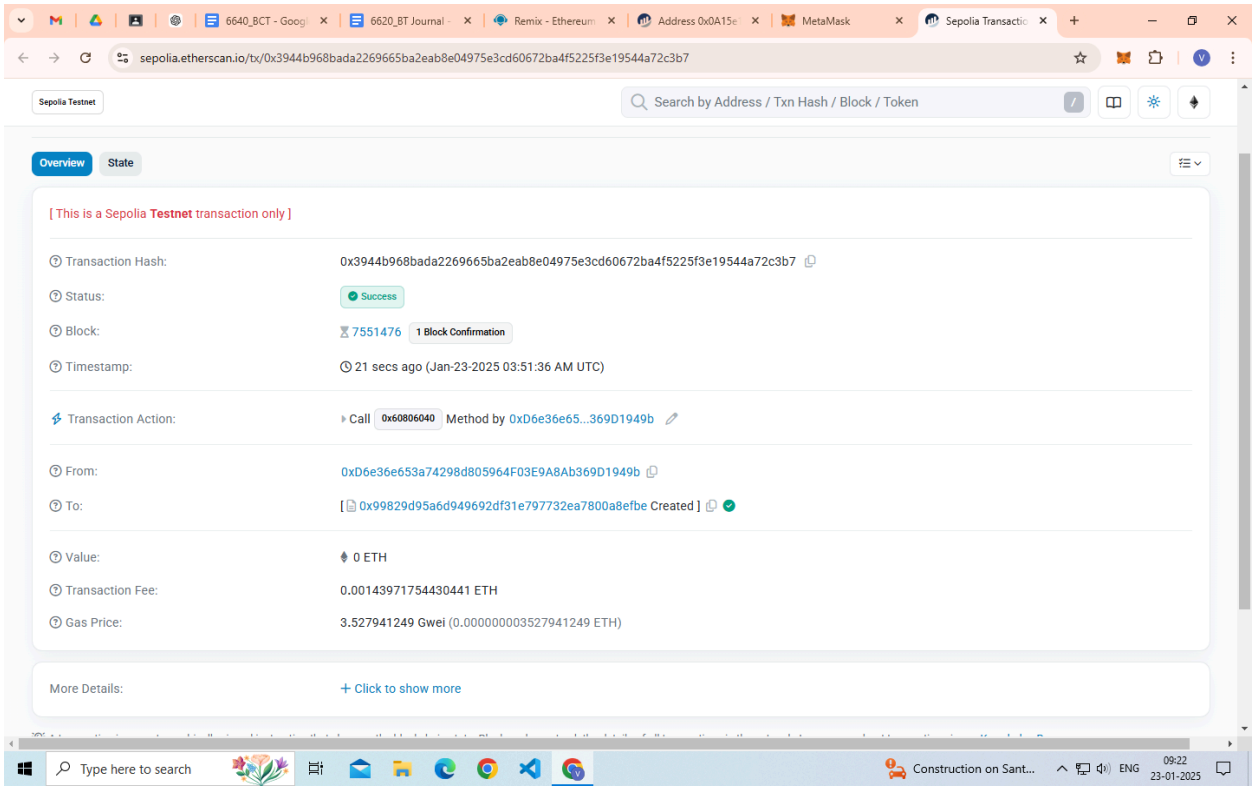
Deploy

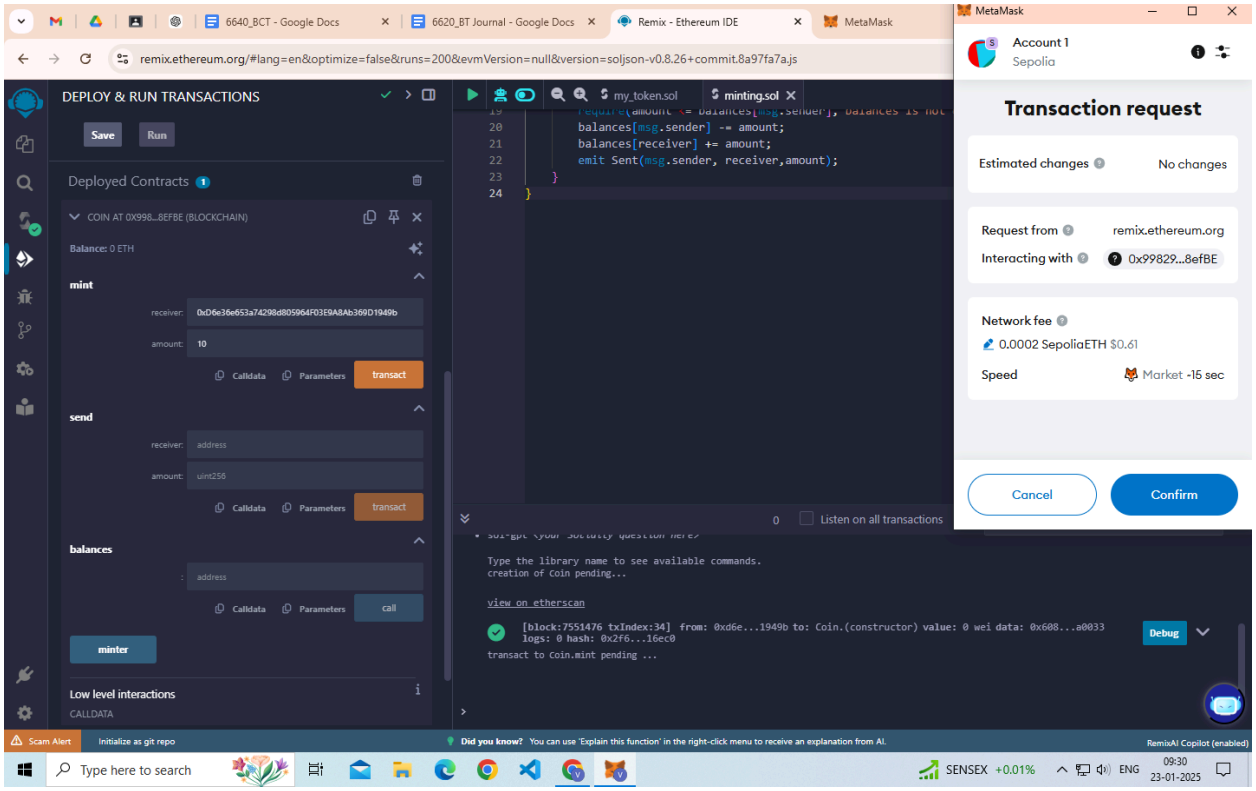


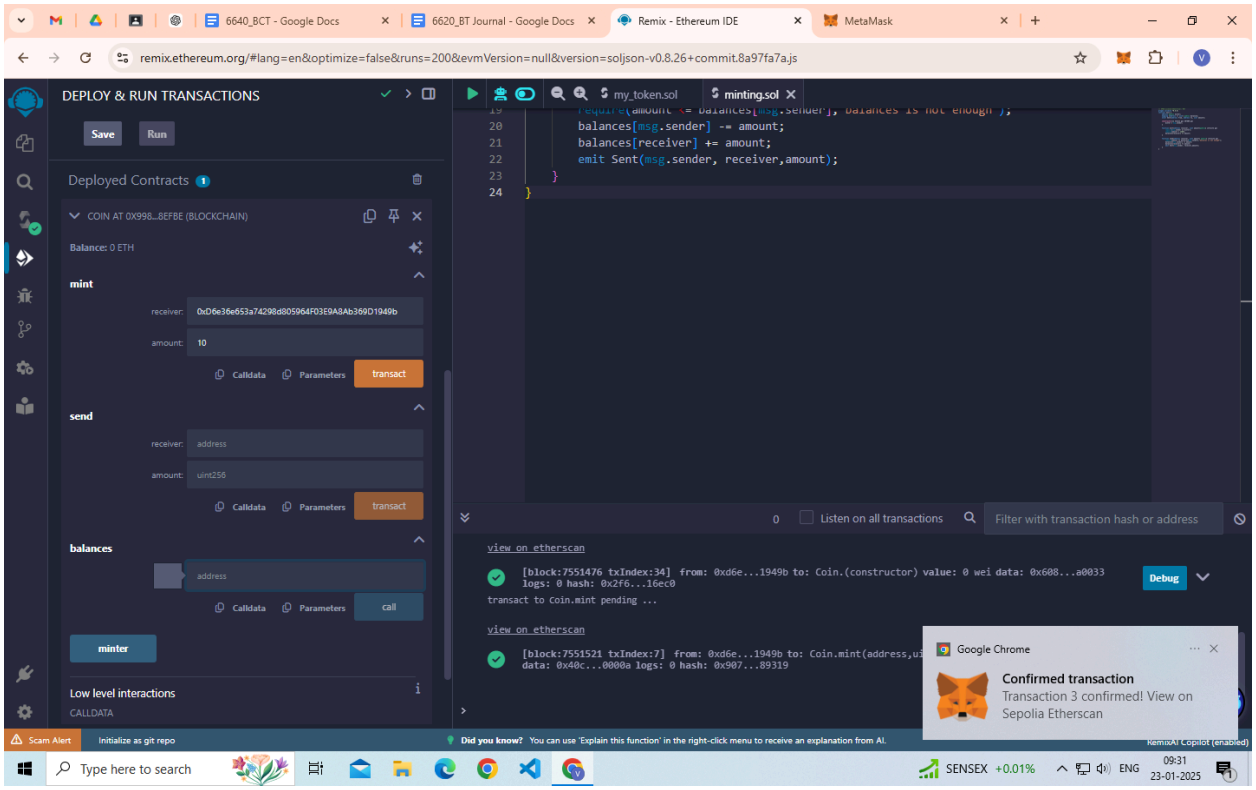
Select Environment as injector provider and connect meta mask wallet
Copy deployment address

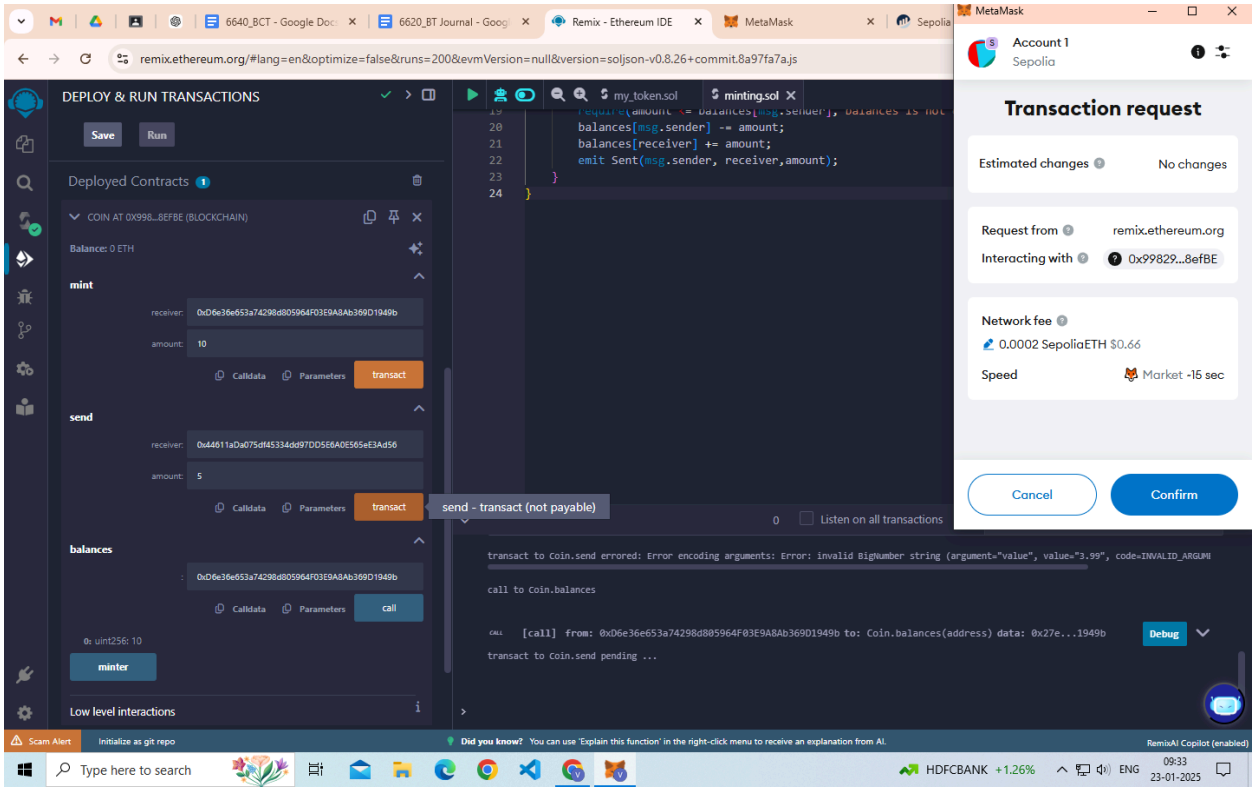


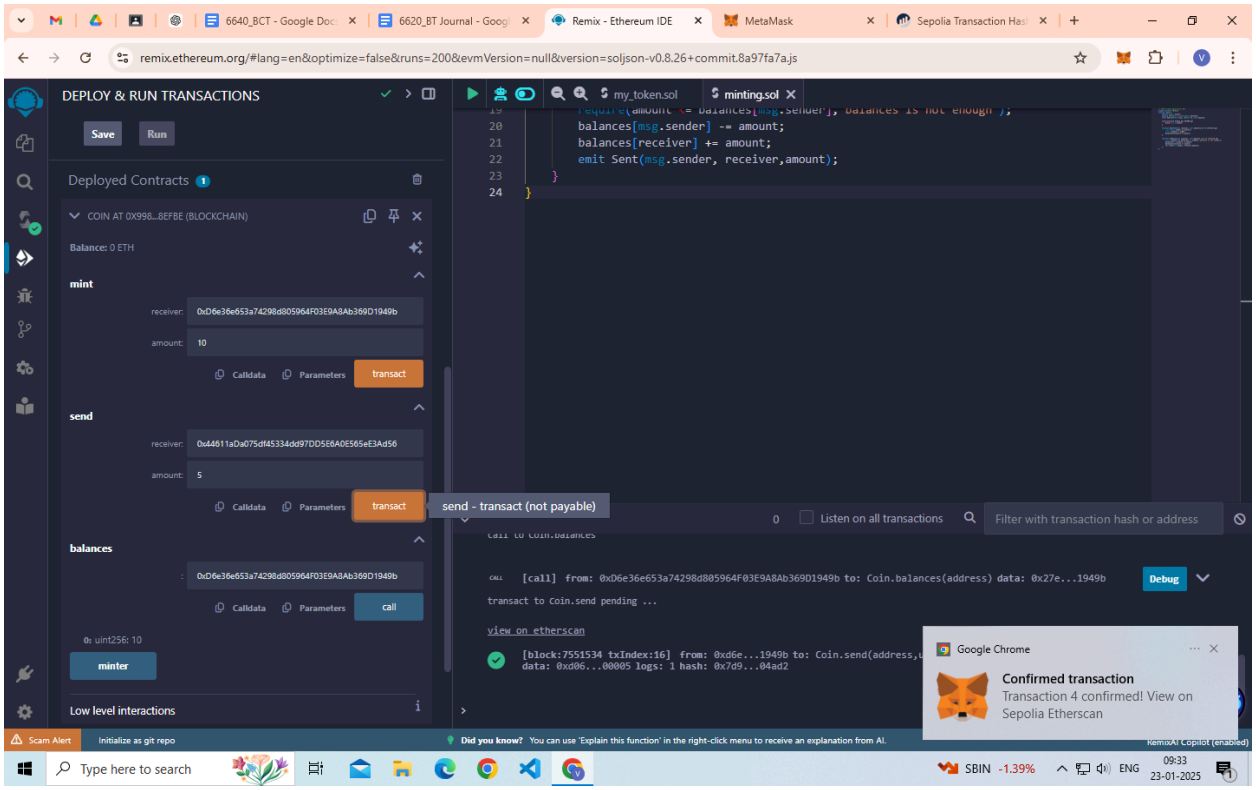
Deployment contract address
0x99829D95a6d949692Df31e797732ea7800a8efBE

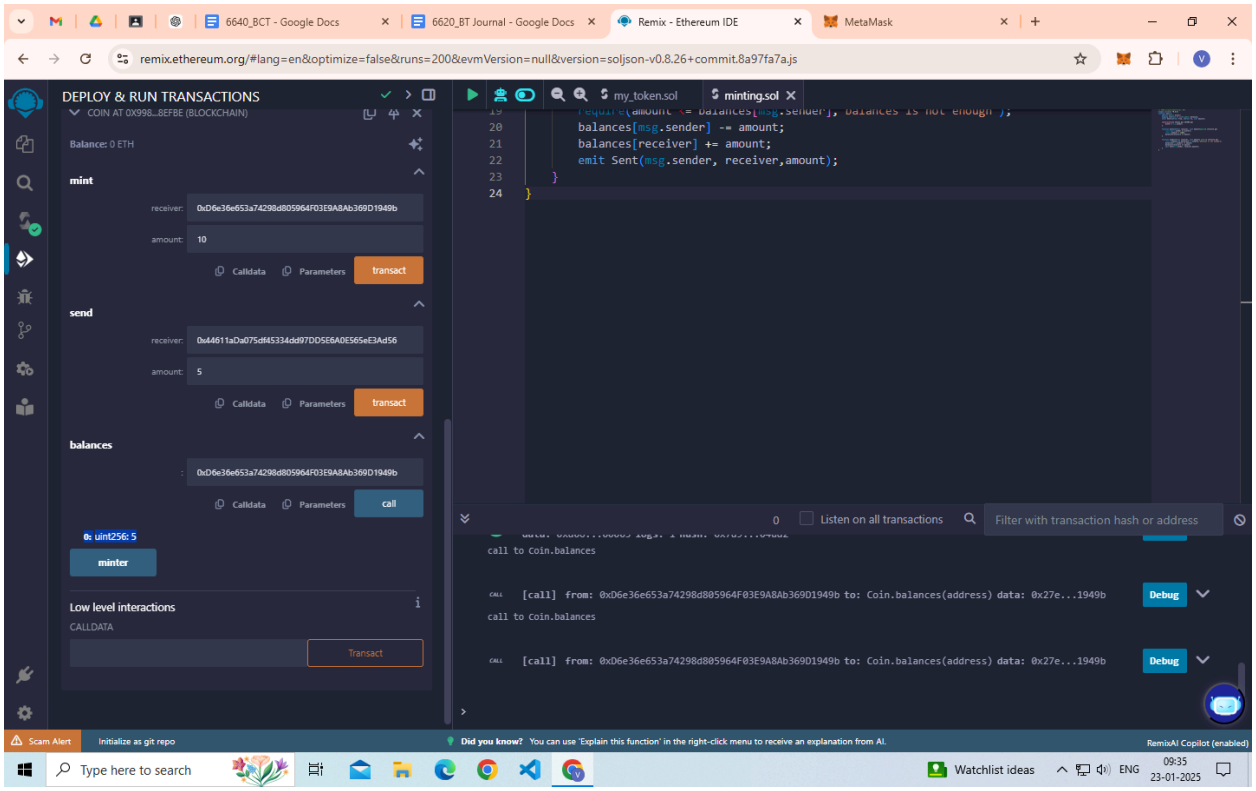

sepolia.etherscan.io/tx/0x3944b968bada2269665ba2eab8e04975e3cd60672ba4f5225f3e19544a72c3b7
Sepolia Testnet
Search by Address / Txn Hash / Block / Token
Overview
State
[This is a Sepolia Testnet transaction only]
Transaction Hash: 0x3944b968bada2269665ba2eab8e04975e3cd60672ba4f5225f3e19544a72c3b7
Status: Success
Block: 7551476 1 Block Confirmation
Timestamp: 21 secs ago (Jan-23-2025 03:51:36 AM UTC)
Transaction Action: Call 0x60806040 Method by 0xD6e36e65...369D1949b
From: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b
To: [0x99829d95a6d949692df31e797732ea7800a8efbe Created]
Value: 0 ETH
Transaction Fee: 0.00143971754430441 ETH
Gas Price: 3.527941249 Gwei (0.000000003527941249 ETH)
More Details: + Click to show more
Type here to search
Construction on Sant...
ENG
09:22
23-01-2025

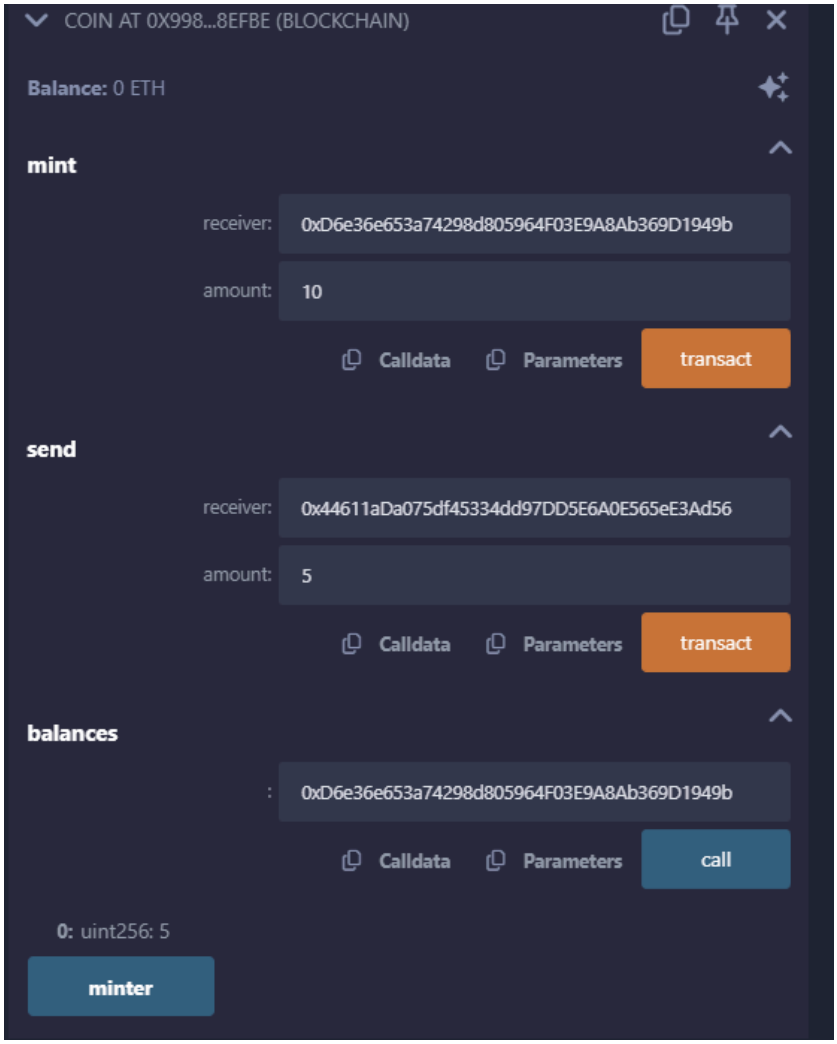

remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
DEPLOY & RUN TRANSACTIONS
Save
Run
Deployed Contracts
COIN AT 0X998...8EFBE (BLOCKCHAIN)
Balance: 0 ETH
mint
receiver: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b
amount: 10
Calldata
Parameters
transact
send
receiver: address
amount: uint256
balances
address
call
minter
Low level interactions
CALLDATA
my_token.sol
minting.sol
balances[msg.sender] -= amount;
balances[receiver] += amount;
emit Sent(msg.sender, receiver,amount);
Listen on all transactions
Type the library name to see available commands.
creation of Coin pending...
view on etherscan
[block:7551476 txIndex:34] from: 0xd6e...1949b to: Coin.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x2f6...16ec0
transact to Coin.mint pending ...
Debug
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
RemixAI Copilot (enabled)
MetaMask
Account 1
Sepolia
Transaction request
Estimated changes
No changes
Request from
remix.ethereum.org
Interacting with
0x99829...8efBE
Network fee
0.0002 SepoliaETH $0.61
Speed
Market ~15 sec
Cancel
Confirm
Type here to search
SENSEX +0.01%
ENG
09:30
23-01-2025

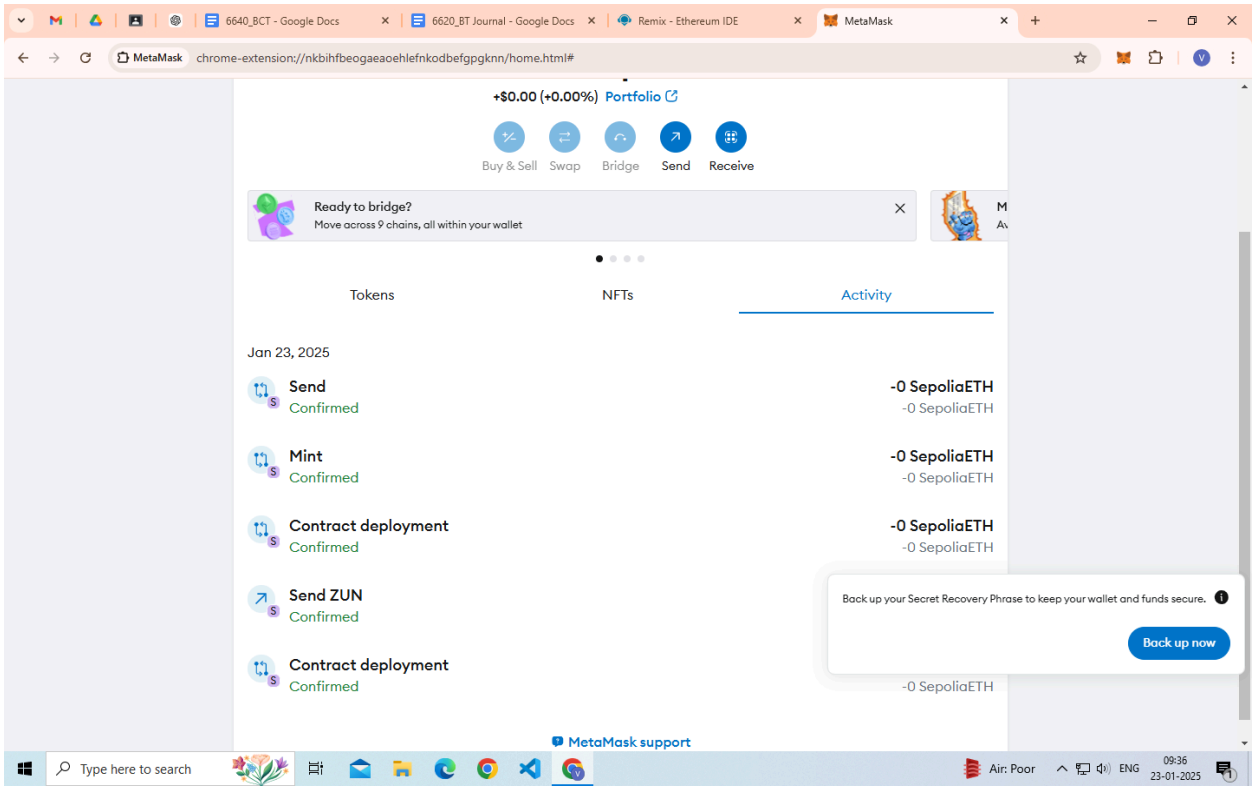

DEPLOY & RUN TRANSACTIONS
Save
Run
Deployed Contracts
COIN AT 0X998...8EFBE (BLOCKCHAIN)
Balance: 0 ETH
mint
receiver: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b
amount: 10
Calldata
Parameters
transact
send
receiver: address
amount: uint256
transact
balances
address
call
minter
Low level interactions
CALLDATA
my_token.sol
minting.sol
balances[msg.sender] -= amount;
balances[receiver] += amount;
emit Sent(msg.sender, receiver,amount);
}
}
Listen on all transactions
Filter with transaction hash or address
view on etherscan
[block:7551476 txIndex:34] from: 0xd6e...1949b to: Coin.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x2f6...16ec0
Debug
transact to Coin.mint pending ...
view on etherscan
[block:7551521 txIndex:7] from: 0xd6e...1949b to: Coin.mint(address,u
data: 0x40c...0000a logs: 0 hash: 0x907...89319
Google Chrome
Confirmed transaction
Transaction 3 confirmed! View on Sepolia Etherscan
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
RemixAI Copilot (enabled)
Type here to search
SENSEX +0.01%
ENG
09:31
23-01-2025


MetaMask
Account 1
Sepolia
Transaction request
Estimated changes
No changes
Request from
remix.ethereum.org
Interacting with
0x99829...8efBE
Network fee
0.0002 SepoliaETH $0.66
Speed
Market ~15 sec
Cancel
Confirm
DEPLOY & RUN TRANSACTIONS
Save
Run
Deployed Contracts
COIN AT 0X998...8EFBE (BLOCKCHAIN)
Balance: 0 ETH
mint
receiver: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b
amount: 10
transact
send
receiver: 0x44611aDa075df45334dd97DD5E6A0E565eE3Ad56
amount: 5
transact
send - transact (not payable)
balances
0xD6e36e653a74298d805964F03E9A8Ab369D1949b
call
0: uint256: 10
minter
Low level interactions
Listen on all transactions
transact to Coin.send errored: Error encoding arguments: Error: invalid BigNumber string (argument="value", value="3.99", code=INVALID_ARGUM
call to Coin.balances
[call] from: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b to: Coin.balances(address) data: 0x27e...1949b
Debug
transact to Coin.send pending ...
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
RemixAI Copilot (enabled)
Type here to search
HDFCBANK +1.26%
ENG
09:33
23-01-2025


DEPLOY & RUN TRANSACTIONS
Save
Run
Deployed Contracts
COIN AT 0X998...8EFBE (BLOCKCHAIN)
Balance: 0 ETH
mint
receiver: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b
amount: 10
Calldata
Parameters
transact
send
receiver: 0x44611aDa075df45334dd97DD5E6A0E565eE3Ad56
amount: 5
Calldata
Parameters
transact
send - transact (not payable)
balances
0xD6e36e653a74298d805964F03E9A8Ab369D1949b
Calldata
Parameters
call
0: uint256: 10
minter
Low level interactions
my_token.sol
minting.sol
balances[msg.sender] -= amount;
balances[receiver] += amount;
emit Sent(msg.sender, receiver,amount);
Listen on all transactions
Filter with transaction hash or address
[call] from: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b to: Coin.balances(address) data: 0x27e...1949b
Debug
transact to Coin.send pending ...
view on etherscan
Google Chrome
Confirmed transaction
Transaction 4 confirmed! View on Sepolia Etherscan
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Type here to search
SBIN -1.39%
ENG
09:33
23-01-2025


DEPLOY & RUN TRANSACTIONS
COIN AT 0X998...8EFBE (BLOCKCHAIN)
Balance: 0 ETH
mint
receiver: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b
amount: 10
Calldata
Parameters
transact
send
receiver: 0x44611aDa075df45334dd97DD5E6A0E565eE3Ad56
amount: 5
Calldata
Parameters
transact
balances
0xD6e36e653a74298d805964F03E9A8Ab369D1949b
Calldata
Parameters
call
0: uint256: 5
minter
Low level interactions
CALLDATA
Transact
my_token.sol
minting.sol
balances[msg.sender] -= amount;
balances[receiver] += amount;
emit Sent(msg.sender, receiver,amount);
Listen on all transactions
Filter with transaction hash or address
call to Coin.balances
[call] from: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b to: Coin.balances(address) data: 0x27e...1949b
Debug
call to Coin.balances
[call] from: 0xD6e36e653a74298d805964F03E9A8Ab369D1949b to: Coin.balances(address) data: 0x27e...1949b
Debug
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Type here to search
Watchlist ideas
ENG
09:35
23-01-2025


COIN AT 0X998...8EFBE (BLOCKCHAIN)
Balance: 0 ETH
mint
receiver:
0xD6e36e653a74298d805964F03E9A8Ab369D1949b
amount:
10
Calldata
Parameters
transact
send
receiver:
0x44611aDa075df45334dd97DD5E6A0E565eE3Ad56
amount:
5
Calldata
Parameters
transact
balances
:
0xD6e36e653a74298d805964F03E9A8Ab369D1949b
Calldata
Parameters
call
0: uint256: 5
minter

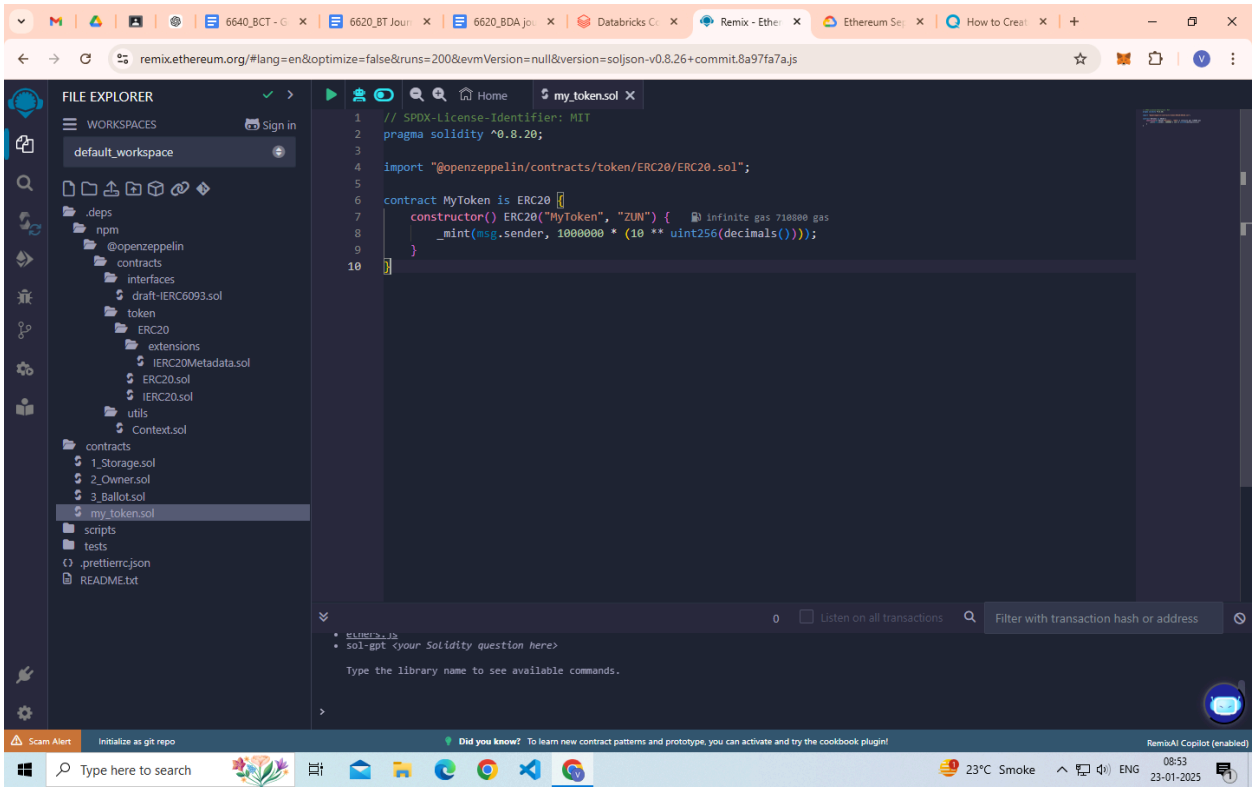

6640_BCT - Google Docs
6620_BT Journal - Google Docs
Remix - Ethereum IDE
MetaMask
chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#
+$0.00 (+0.00%) Portfolio
Buy & Sell
Swap
Bridge
Send
Receive
Ready to bridge?
Move across 9 chains, all within your wallet
Tokens
NFTs
Activity
Jan 23, 2025
Send
Confirmed
-0 SepoliaETH
-0 SepoliaETH
Mint
Confirmed
-0 SepoliaETH
-0 SepoliaETH
Contract deployment
Confirmed
-0 SepoliaETH
-0 SepoliaETH
Send ZUN
Confirmed
Contract deployment
Confirmed
-0 SepoliaETH
Back up your Secret Recovery Phrase to keep your wallet and funds secure.
Back up now
MetaMask support
Type here to search
Air: Poor
ENG
09:36
23-01-2025

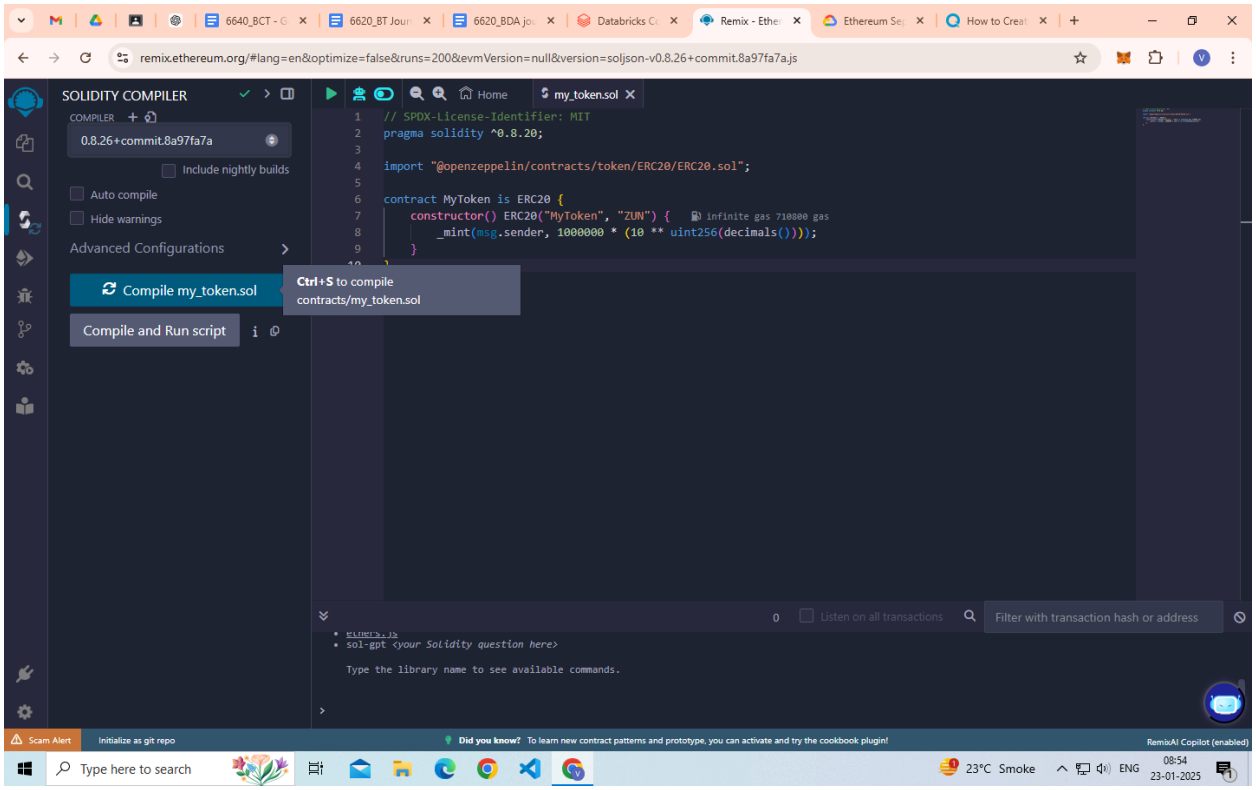
B. Creating ERC -20 Token

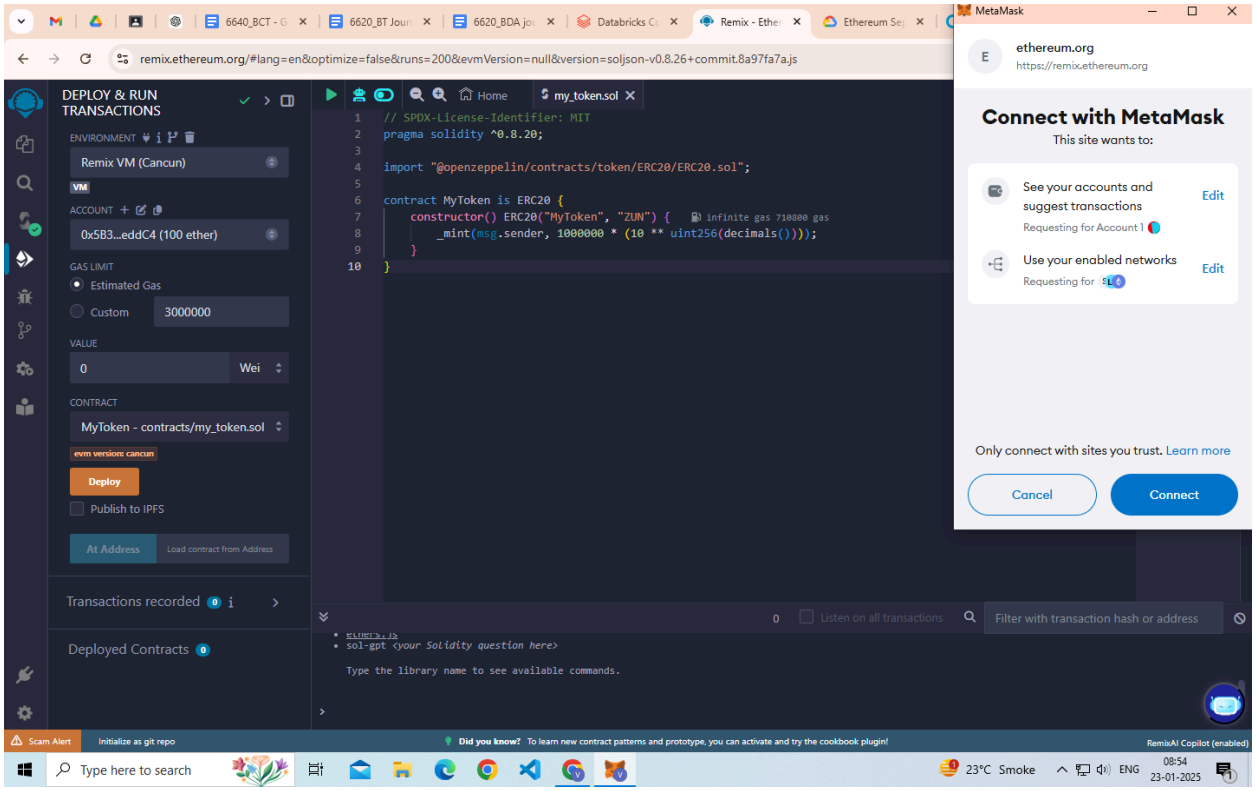
Open MetaMask Wallet
Get ETH fromGoogle Cloud seplia Network test network faucet
https://cloud.google.com/application/web3/faucet/ethereum/sepolia

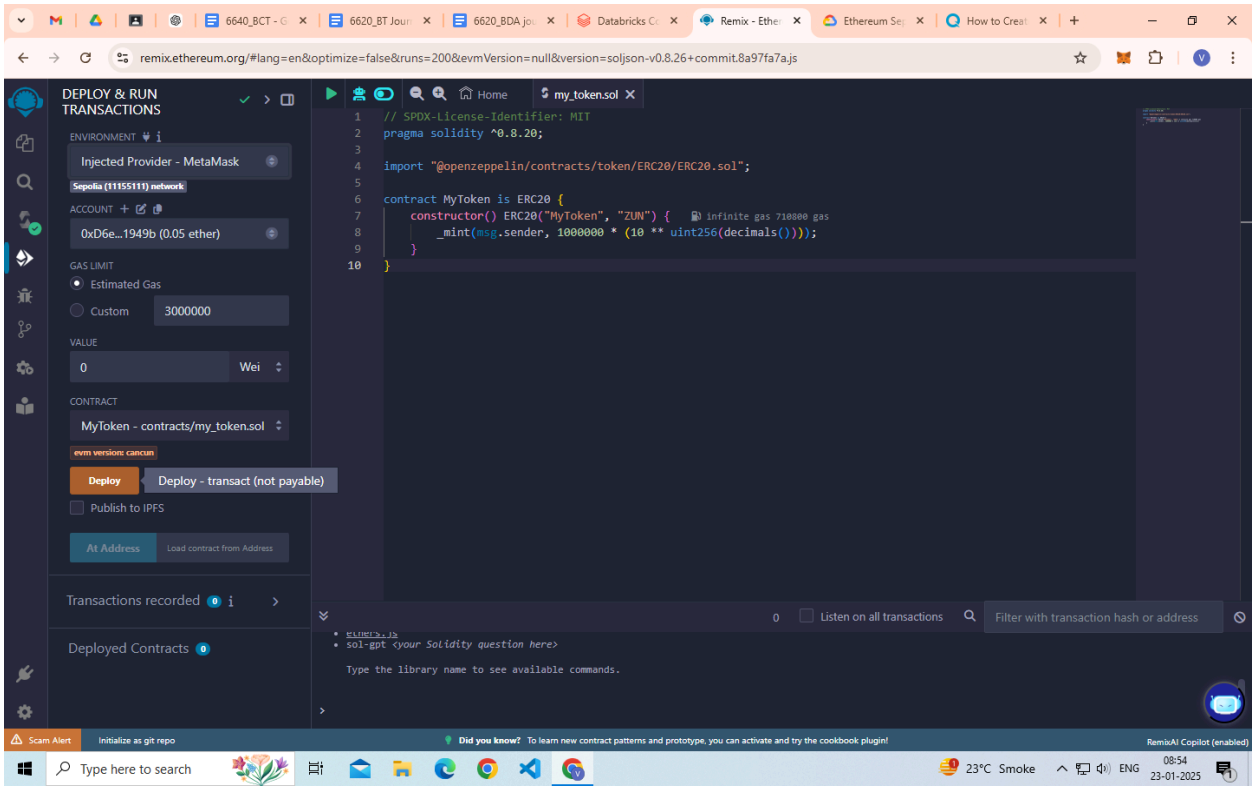
REF:
https://www.quicknode.com/guides/ethereum-development/smart-contracts/how-to-create-and-deploy-an-erc20-token

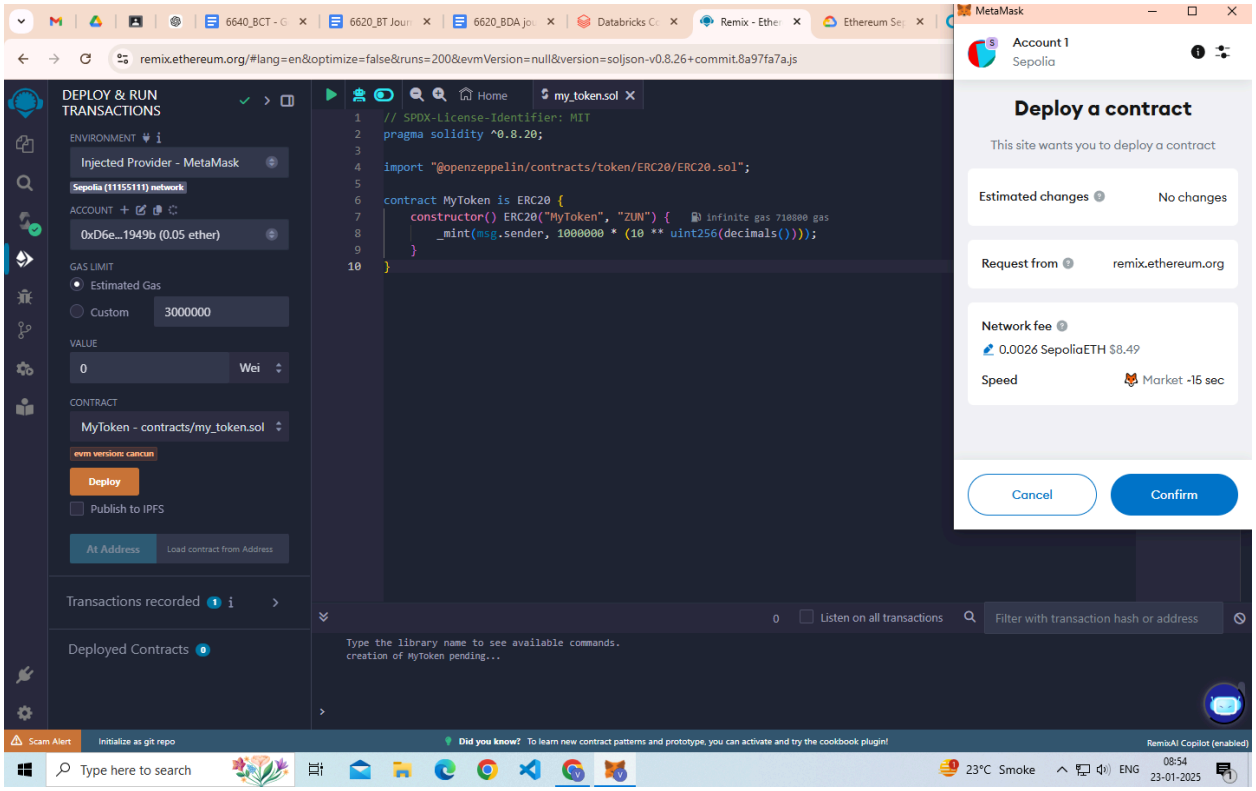
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract MyToken is ERC20 {
    constructor() ERC20("MyToken", "ZUN") {
        _mint(msg.sender, 1000000 * (10 ** uint256(decimals())));
    }
}
```

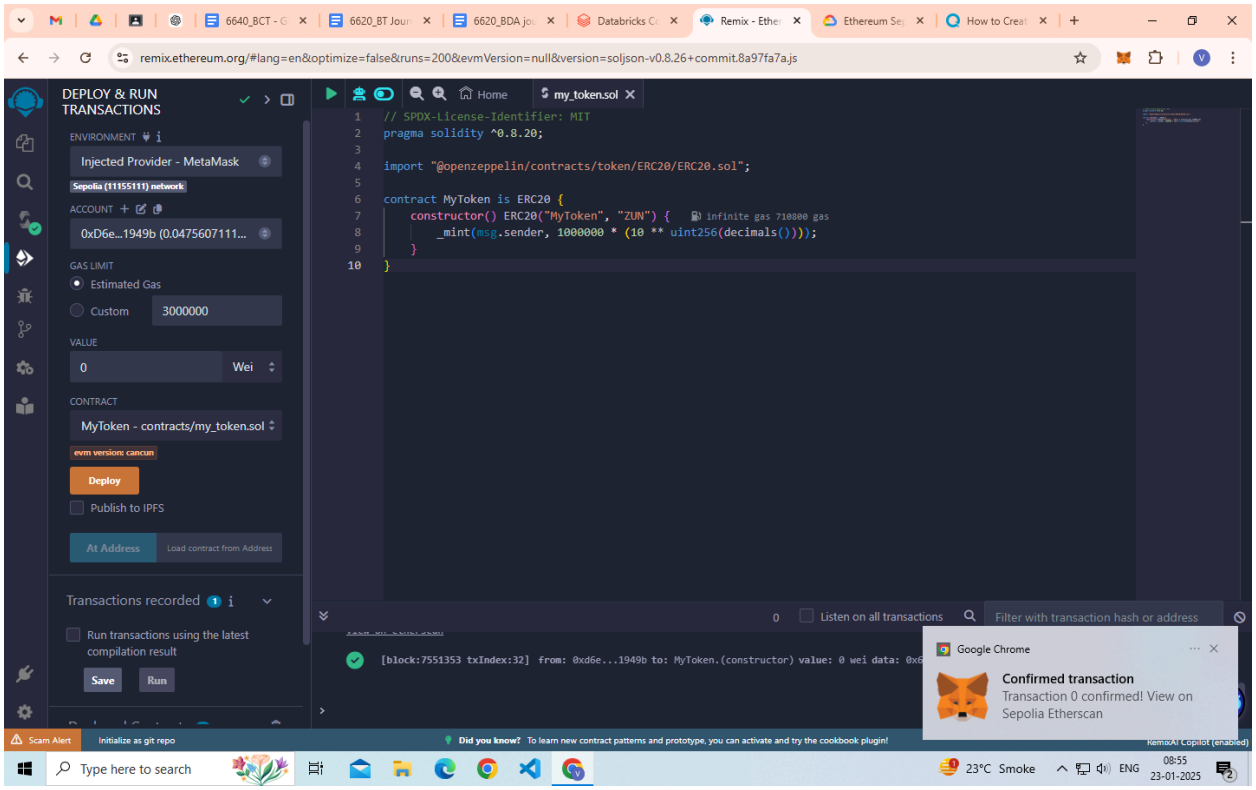


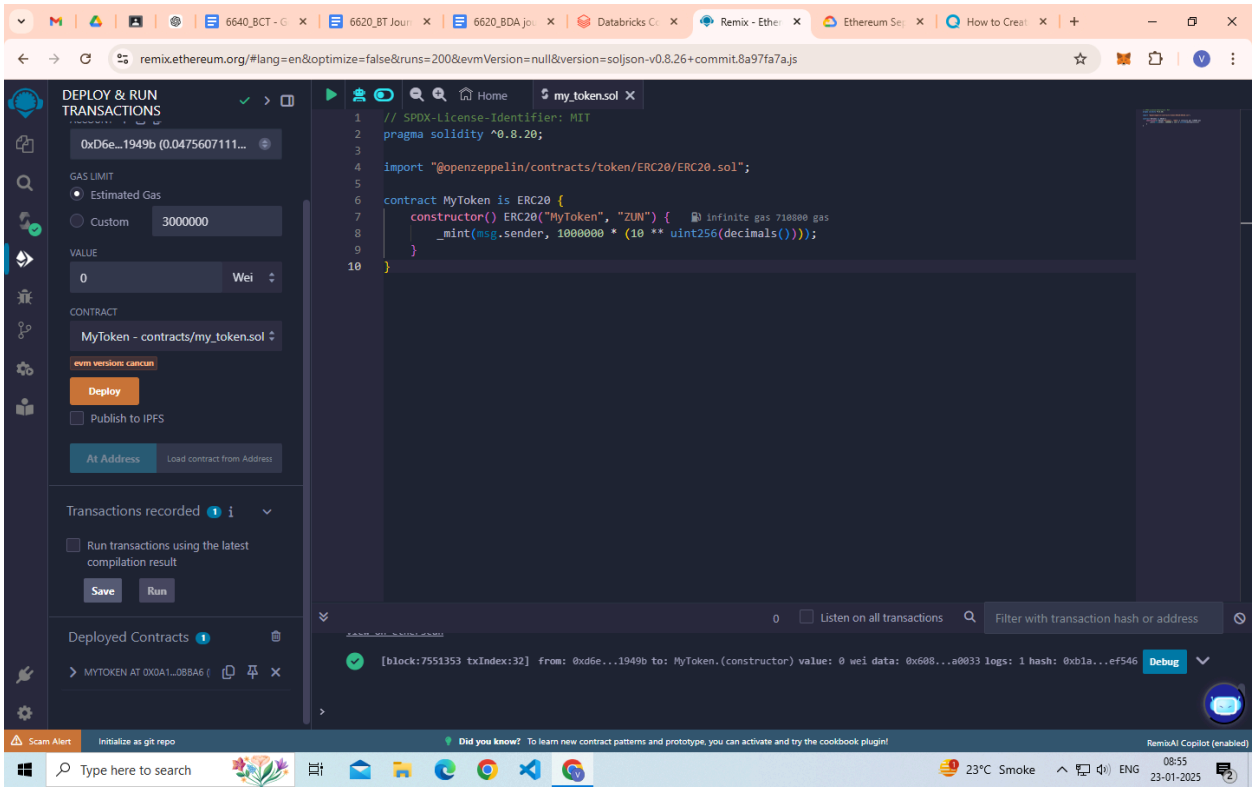

SOLIDITY COMPILER
COMPILER
0.8.26+commit.8a97fa7a
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile my_token.sol
Ctrl+S to compile
contracts/my_token.sol
Compile and Run script
Home
my_token.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
contract MyToken is ERC20 {
constructor() ERC20("MyToken", "ZUN") {
_mint(msg.sender, 1000000 * (10 ** uint256(decimals())));
}
}
Listen on all transactions
Filter with transaction hash or address
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Scam Alert
Initialize as git repo
Did you know? To learn new contract patterns and prototype, you can activate and try the cookbook plugin!
RemixAI Copilot (enabled)
Type here to search
23°C Smoke
ENG
08:54
23-01-2025


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (100 ether)
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
MyToken - contracts/my_token.sol
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
MetaMask
ethereum.org
https://remix.ethereum.org
Connect with MetaMask
This site wants to:
See your accounts and suggest transactions
Requesting for Account 1
Edit
Use your enabled networks
Requesting for
Edit
Only connect with sites you trust. Learn more
Cancel
Connect
Home
my_token.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
contract MyToken is ERC20 {
constructor() ERC20("MyToken", "ZUN") {
_mint(msg.sender, 1000000 * (10 ** uint256(decimals())));
}
}
Listen on all transactions
Filter with transaction hash or address
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Scam Alert
Initialize as git repo
Did you know? To learn new contract patterns and prototype, you can activate and try the cookbook plugin!
RemixAI Copilot (enabled)
Type here to search
23°C Smoke
ENG
08:54
23-01-2025

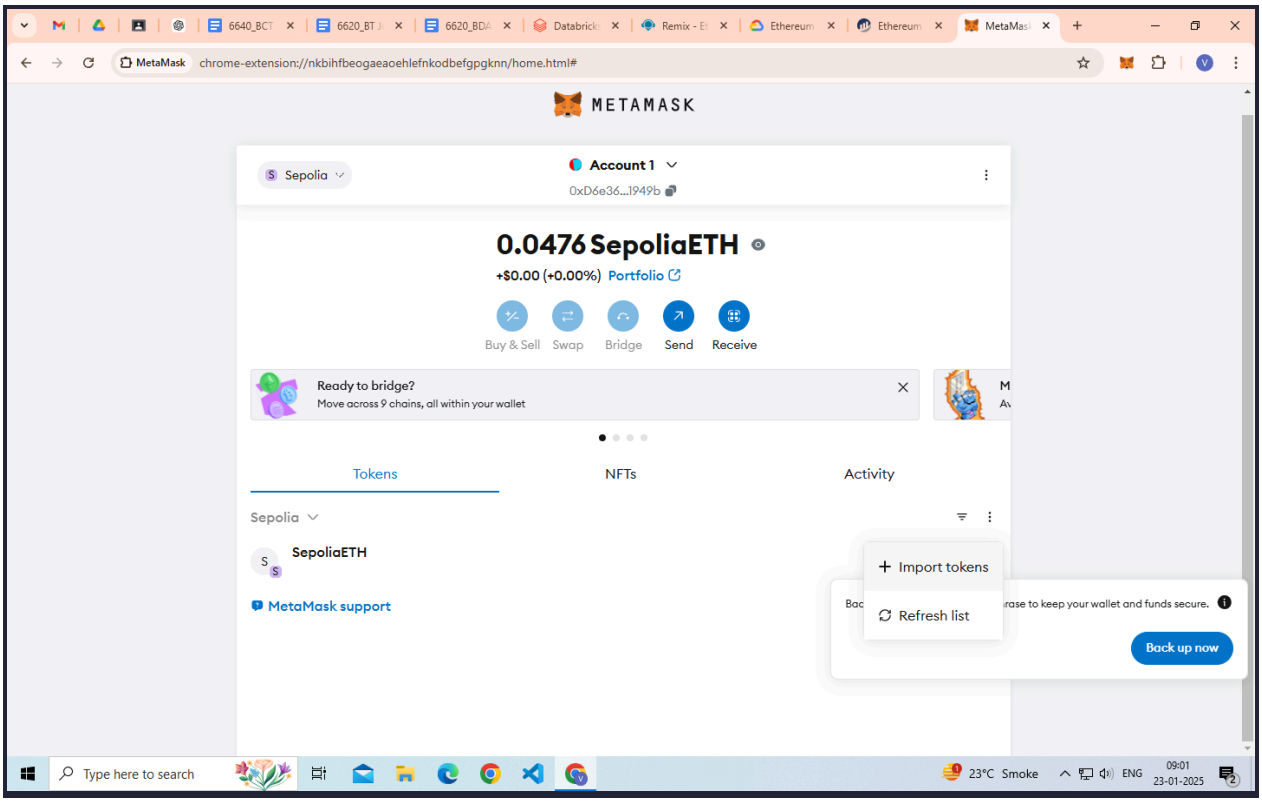

DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Injected Provider - MetaMask
Sepolia (11155111) network
ACCOUNT
0xD6e...1949b (0.05 ether)
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
MyToken - contracts/my_token.sol
evm version: cancun
Deploy
Deploy - transact (not payable)
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 0
Deployed Contracts 0
Home
my_token.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
contract MyToken is ERC20 {
constructor() ERC20("MyToken", "ZUN") { infinite gas 710800 gas
_mint(msg.sender, 1000000 * (10 ** uint256(decimals())));
}
}
Listen on all transactions
Filter with transaction hash or address
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Scam Alert
Initialize as git repo
Did you know? To learn new contract patterns and prototype, you can activate and try the cookbook plugin!
RemixAI Copilot (enabled)
Type here to search
23°C Smoke
ENG
08:54
23-01-2025


MetaMask
Account 1
Sepolia
Deploy a contract
This site wants you to deploy a contract
Estimated changes
No changes
Request from
remix.ethereum.org
Network fee
0.0026 SepoliaETH $8.49
Speed
Market ~15 sec
Cancel
Confirm
DEPLOY & RUN TRANSACTIONS
Transactions recorded 1
Deployed Contracts 0
Type the library name to see available commands.
creation of MyToken pending...

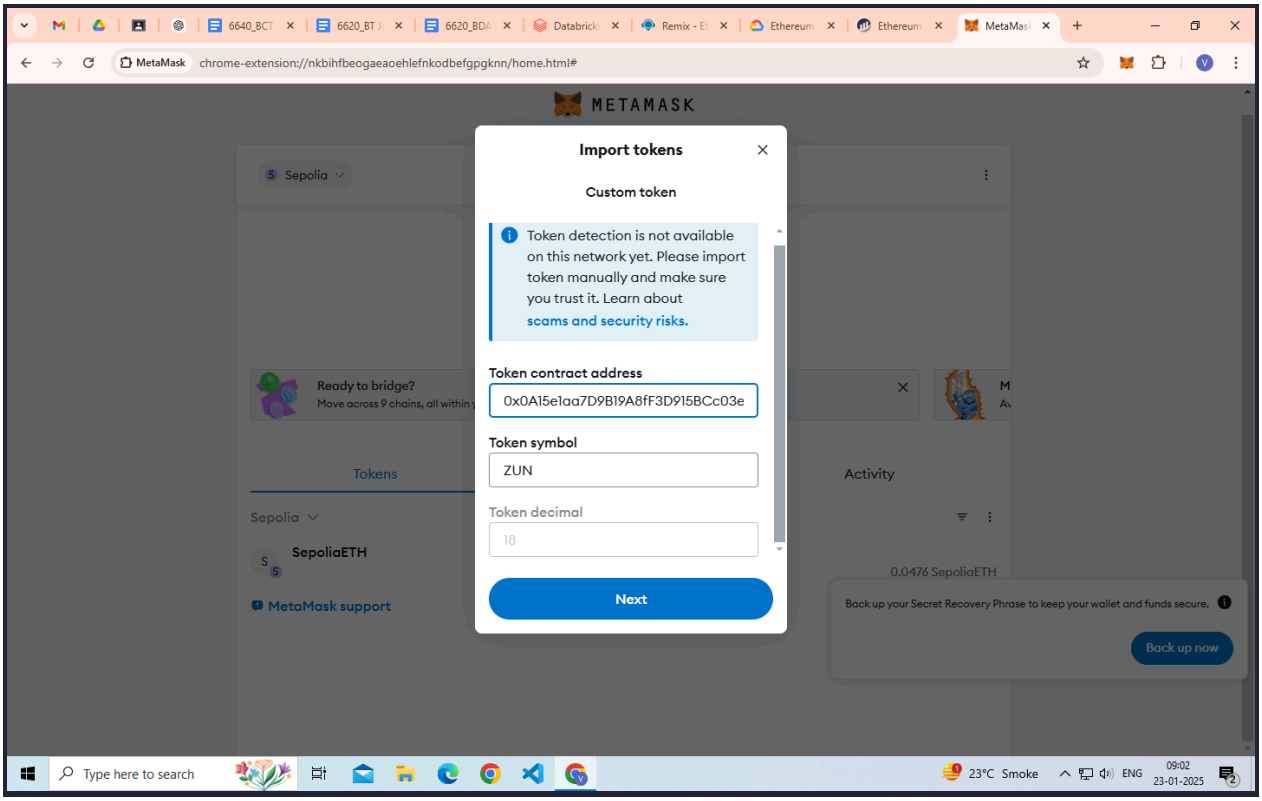


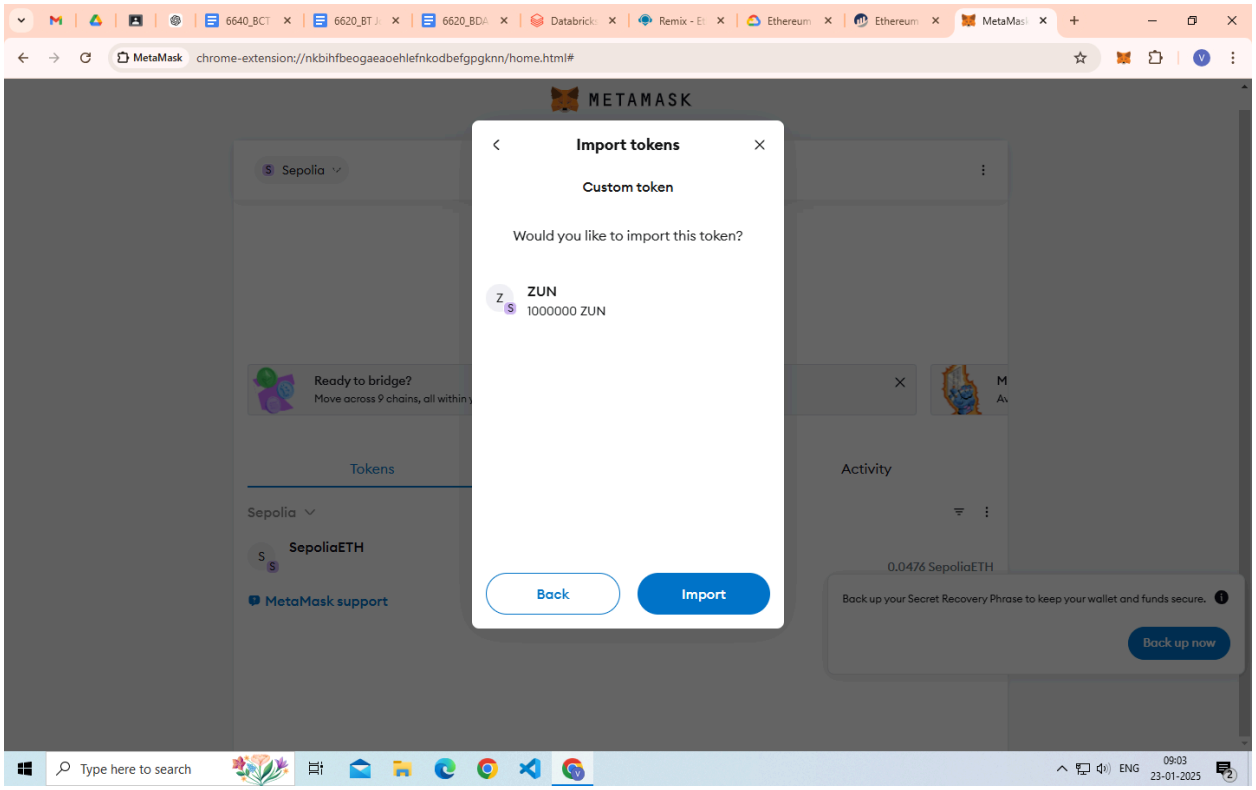


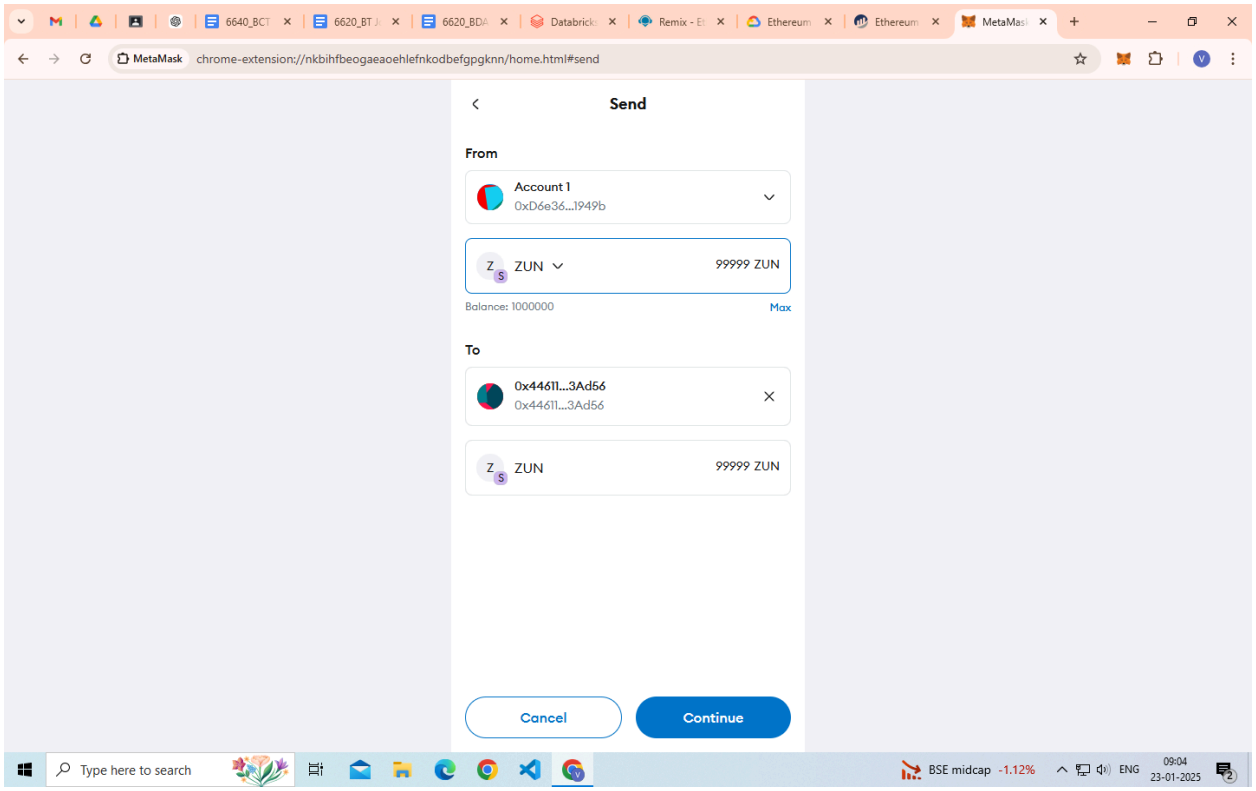
0x0A15e1aa7D9B19A8fF3D915BCc03e4F07060bBA6

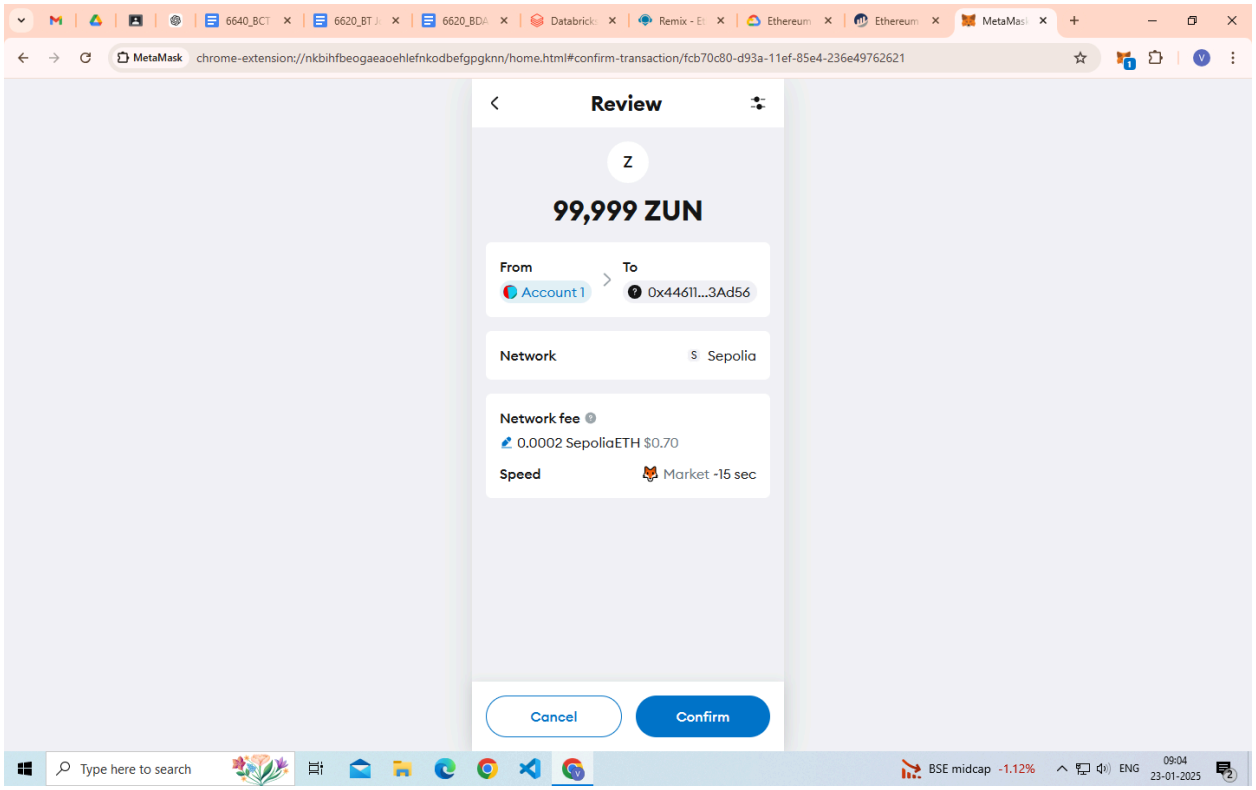


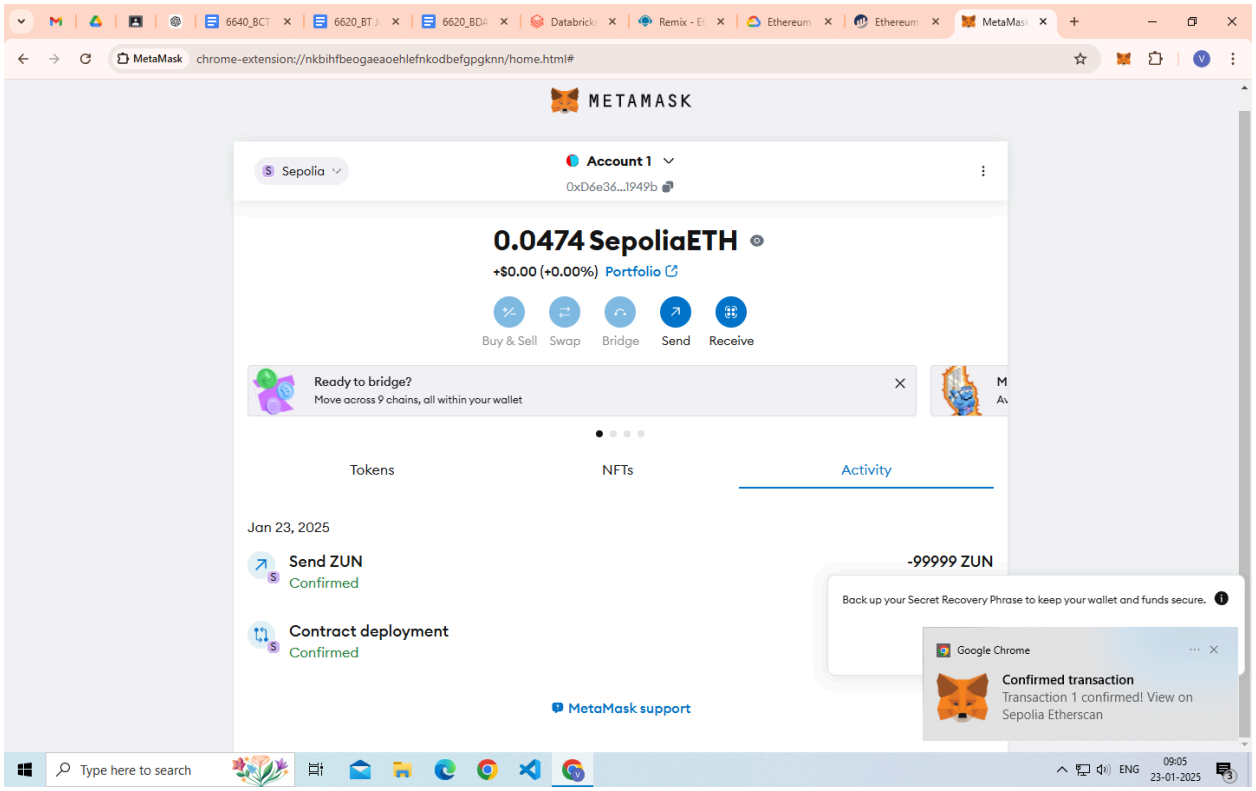
Enter Remix transaction ID

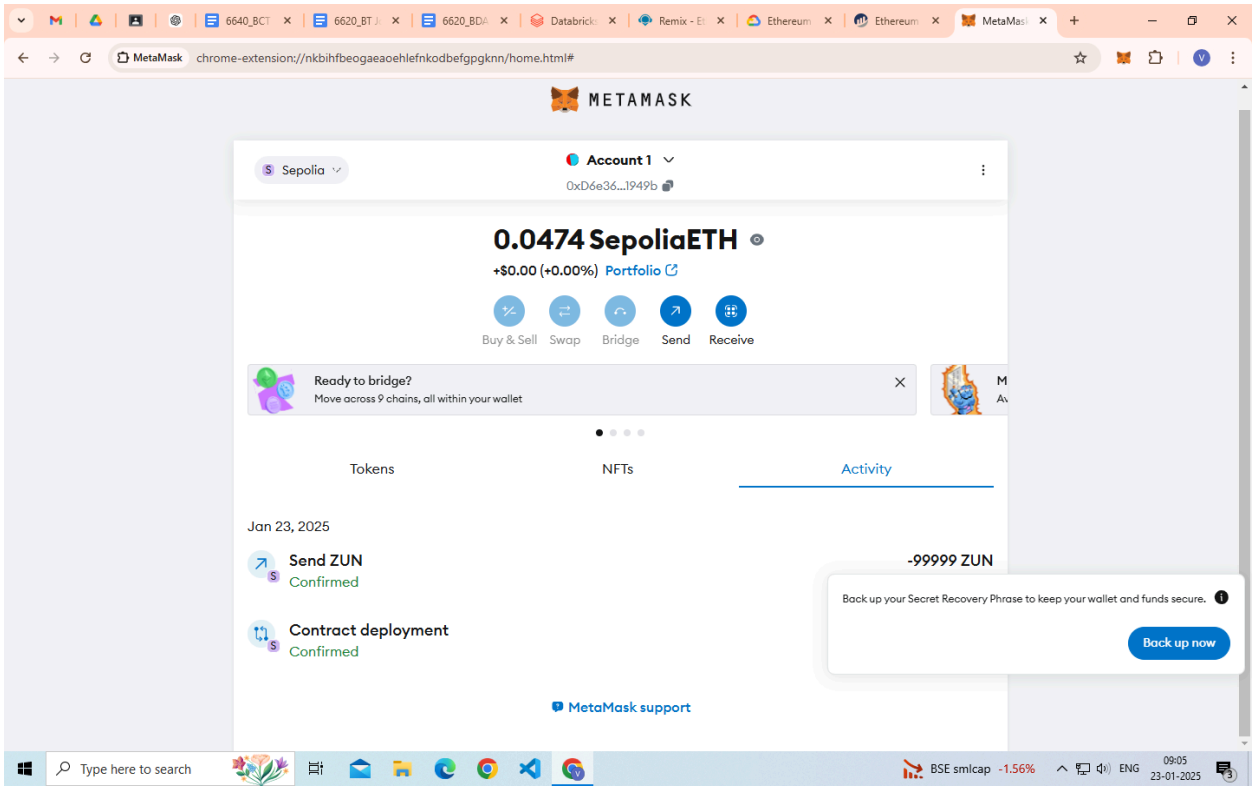



METAMASK
Import tokens
Custom token
Would you like to import this token?
ZUN
1000000 ZUN
Back
Import
Sepolia
Ready to bridge?
Tokens
Activity
SepoliaETH
0.0476 SepoliaETH
MetaMask support
Back up your Secret Recovery Phrase to keep your wallet and funds secure.
Back up now
chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#
Type here to search
ENG
09:03
23-01-2025

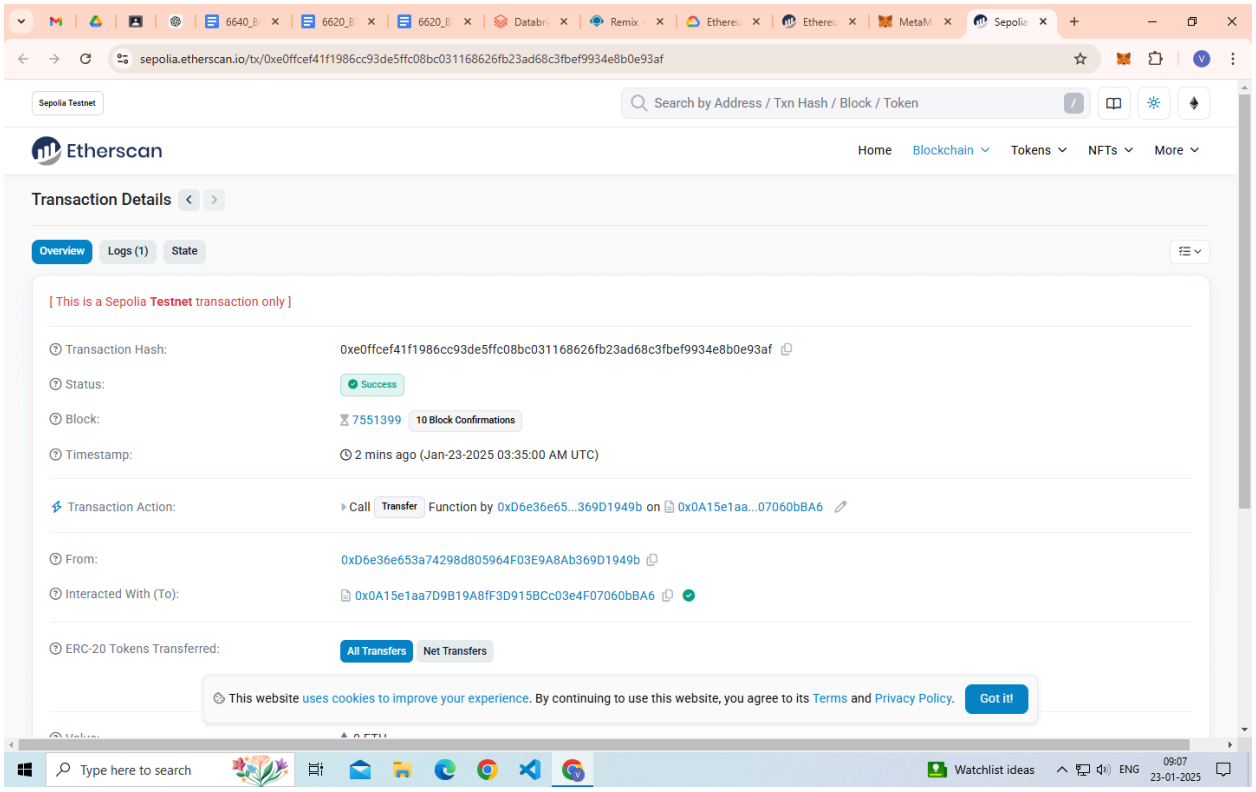

Send
From
Account 1
0xD6e36...1949b
ZUN
99999 ZUN
Balance: 1000000
Max
To
0x44611...3Ad56
0x44611...3Ad56
ZUN
99999 ZUN
Cancel
Continue
chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#send
Type here to search
BSE midcap -1.12%
ENG
09:04
23-01-2025

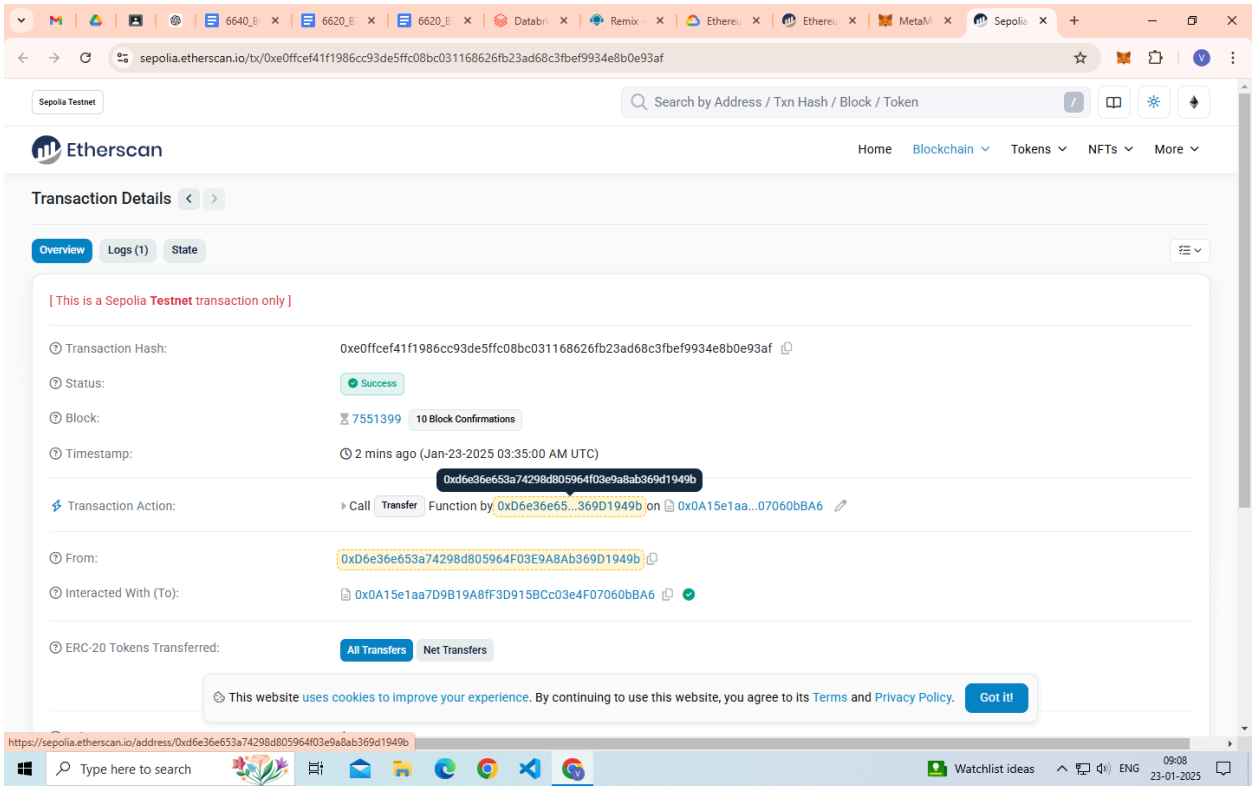

MetaMask
chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#confirm-transaction/fcb70c80-d93a-11ef-85e4-236e49762621
Review
Z
99,999 ZUN
From
Account 1
To
0x44611...3Ad56
Network
Sepolia
Network fee
0.0002 SepoliaETH $0.70
Speed
Market -15 sec
Cancel
Confirm
Type here to search
BSE midcap -1.12%
ENG
09:04
23-01-2025

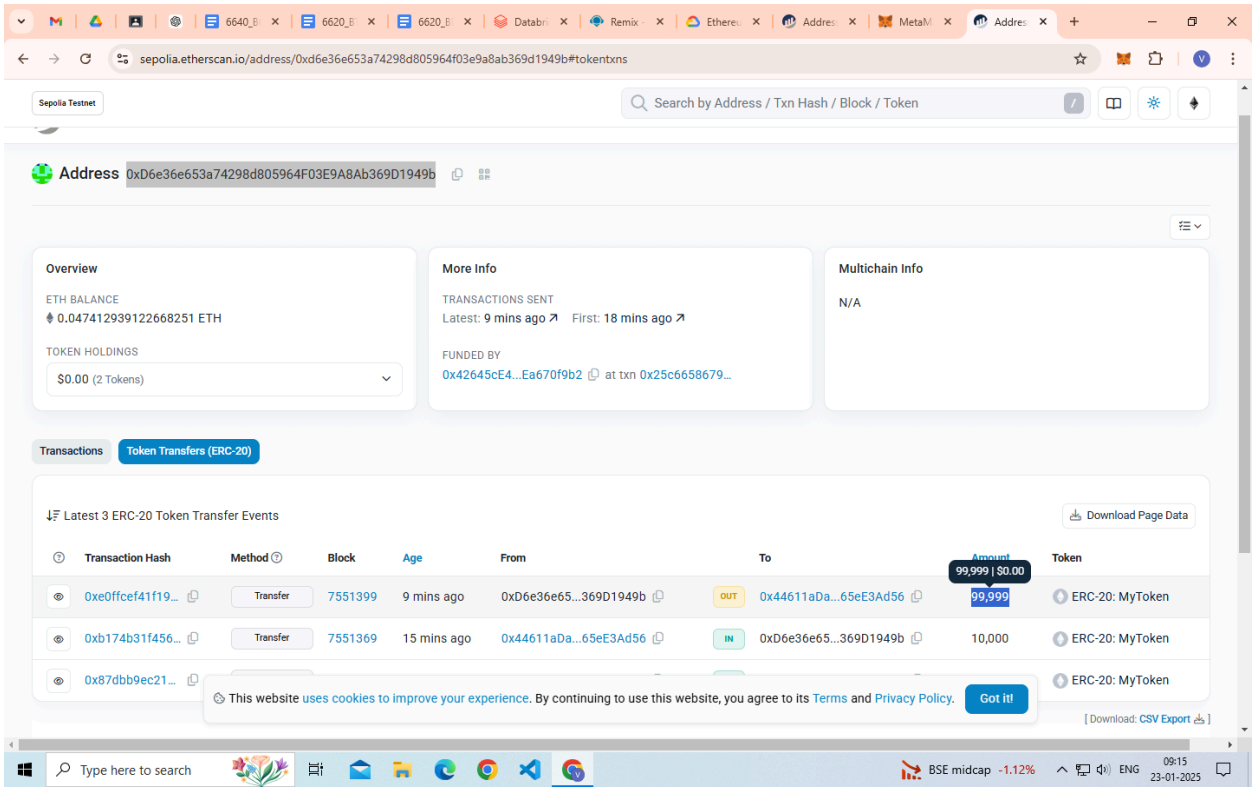

MetaMask
chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#
METAMASK
Sepolia
Account 1
0xD6e36...1949b
0.0474 SepoliaETH
+$0.00 (+0.00%) Portfolio
Buy & Sell
Swap
Bridge
Send
Receive
Ready to bridge?
Move across 9 chains, all within your wallet
Tokens
NFTs
Activity
Jan 23, 2025
Send ZUN
Confirmed
-99999 ZUN
Contract deployment
Confirmed
MetaMask support
Back up your Secret Recovery Phrase to keep your wallet and funds secure.
Google Chrome
Confirmed transaction
Transaction 1 confirmed! View on Sepolia Etherscan
Type here to search
ENG
09:05
23-01-2025



Click on Meta mask confirm transaction notification




sepolia.etherscan.io/tx/0xe0ffcef41f1986cc93de5ffc08bc031168626fb23ad68c3fbef9934e8b0e93af
Sepolia Testnet
Search by Address / Txn Hash / Block / Token
Etherscan
Home
Blockchain
Tokens
NFTs
More
Transaction Details
Overview
Logs (1)
State
[This is a Sepolia Testnet transaction only]
Transaction Hash:
0xe0ffcef41f1986cc93de5ffc08bc031168626fb23ad68c3fbef9934e8b0e93af
Status:
Success
Block:
7551399
10 Block Confirmations
Timestamp:
2 mins ago (Jan-23-2025 03:35:00 AM UTC)
Transaction Action:
Call
Transfer
Function by
0xD6e36e65...369D1949b
on
0x0A15e1aa...07060bBA6
0xd6e36e36e653a74298d805964f03e9a8ab369d1949b
From:
0xD6e36e653a74298d805964F03E9A8Ab369D1949b
Interacted With (To):
0x0A15e1aa7D9B19A8fF3D915BCc03e4F07060bBA6
ERC-20 Tokens Transferred:
All Transfers
Net Transfers
This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy.
Got it!
https://sepolia.etherscan.io/address/0xd6e36e653a74298d805964f03e9a8ab369d1949b
Type here to search
Watchlist ideas
ENG
09:08
23-01-2025


sepolia.etherscan.io/address/0xd6e36e653a74298d805964f03e9a8ab369d1949b#tokentxns
Sepolia Testnet
Search by Address / Txn Hash / Block / Token
Address
0xD6e36e653a74298d805964F03E9A8Ab369D1949b
Overview
ETH BALANCE
0.047412939122668251 ETH
TOKEN HOLDINGS
$0.00 (2 Tokens)
More Info
TRANSACTIONS SENT
Latest: 9 mins ago
First: 18 mins ago
FUNDED BY
0x42645cE4...Ea670f9b2
at txn 0x25c6658679...
Multichain Info
N/A
Transactions
Token Transfers (ERC-20)
Latest 3 ERC-20 Token Transfer Events
Download Page Data
Transaction Hash
Method
Block
Age
From
To
Amount
Token
0xe0ffcef41f19...
Transfer
7551399
9 mins ago
0xD6e36e65...369D1949b
OUT
0x44611aDa...65eE3Ad56
99,999
ERC-20: MyToken
0xb174b31f456...
Transfer
7551369
15 mins ago
0x44611aDa...65eE3Ad56
IN
0xD6e36e65...369D1949b
10,000
ERC-20: MyToken
0x87dbb9ec21...
ERC-20: MyToken
99,999 | $0.00
This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy.
Got it!
[Download: CSV Export]
Type here to search
BSE midcap -1.12%
ENG
09:15
23-01-2025