ASSIGNMENT – 4

(Blockchain Technology)

Submitted to: Mr. Shashikant

Name: Ishit Singh

Class: 3NC1

Roll No.: 102115023

Semester: Jan’24-May’24

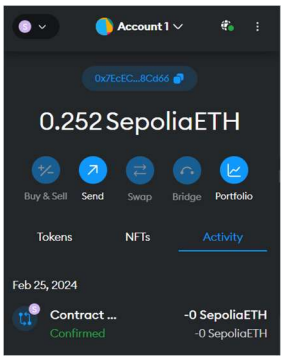
CODE:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract ERC20{

    address contractRunner;                 // address of contract/token owner
    string public name;                 // name of the token
    string public symbol;               // symbol of the token
    uint8 public immutable decimals;    // number of decimal places that the
token can be divided into [IMMUTABLE ---> Can't be altered again]
    uint public immutable totalSupply;  // total supply of tokens that have
been created for a particular project

    /*  Mapping is a hash table in Solidity that stores data as key-value
pairs, where the key can be any of the built-in data types
        1. Never let an array in Solidity grow too large because iterating
through a large array could cost more in Solidity gas fees than the
transaction's value,
        2. making mappings a more gas efficient smart contract implementation
    */
    mapping(address => uint) balances;                          // Account
address --- has count of tokens ---> uint
    // spender => (owner => no. of tokens allowed)
    mapping(address => mapping(address => uint)) allowances;    // Account
address --- has delegated account --> address --- to have tokens amount of ---
> uint

    // EVENTS : Data stored on blockchain that is only to be STORED and
neither Modified OR Accessed
    event Transfer(address indexed _from, address indexed _to, uint value);
    event Approval(address indexed _owner, address indexed _spender, uint
value);

    constructor(string memory _name, string memory _symbol, uint8 _decimals,
uint _totalSupply){
        contractRunner = msg.sender;
        name = _name;
        symbol = _symbol;
        decimals = _decimals;
        totalSupply = _totalSupply;
        balances[contractRunner] = totalSupply;
    }

    // function returns the balance of tokens held by a particular address.
    function balanceOf(address _owner) public view returns(uint){
        require(_owner != address(0), "Zero Address");
        return balances[_owner];
```

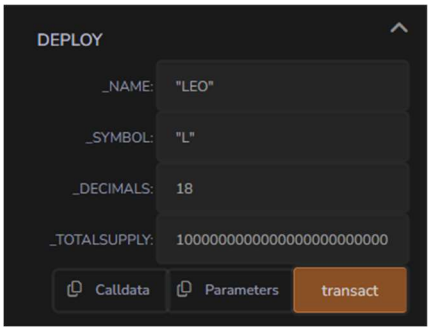
```
    }

    // function allows an address to send tokens to another address.
    function transfer(address _to, uint _tokenVal) public returns(bool){
        require((balances[contractRunner] >= _tokenVal) &&
(balances[contractRunner] >= 0), "Insufficient Balance");

        balances[contractRunner] -= _tokenVal;
        balances[_to] += _tokenVal;
        emit Transfer(contractRunner, _to, _tokenVal);
        return true;
    }

    // function allows an address [that has approved of transfer] to transfer
tokens from one address to another address.
    function transferFrom(address _from, address _to, uint tokenVal) public
returns(bool){
        require(allowances[contractRunner][_from] >= tokenVal, "Tokens
transfer not allowed");
        require((balances[_from] >= tokenVal) && (balances[_from] > 0),
"Insufficient Balance");

        balances[_from] -= tokenVal;
        balances[_to] += tokenVal;
        allowances[contractRunner][_from] -= tokenVal;
        emit Transfer(_from, _to, tokenVal);
        return true;
    }

    // function allows an address(msg.sender) to approve another
address(spender) to spend tokens on their behalf.
    function approve(address _spender, uint _tokenVal) public returns(bool){
        require(balances[contractRunner] >= _tokenVal,"Insufficient balance");
        allowances[_spender][contractRunner] = _tokenVal;
        emit Approval(contractRunner, _spender, _tokenVal);
        return true;
    }

    // function returns the amount of tokens that an approved address can
spend on behalf of another address.
    function allowanceFunc(address _owner, address _spender) public view
returns(uint){
        return allowances[_spender][_owner];
    }


}
```

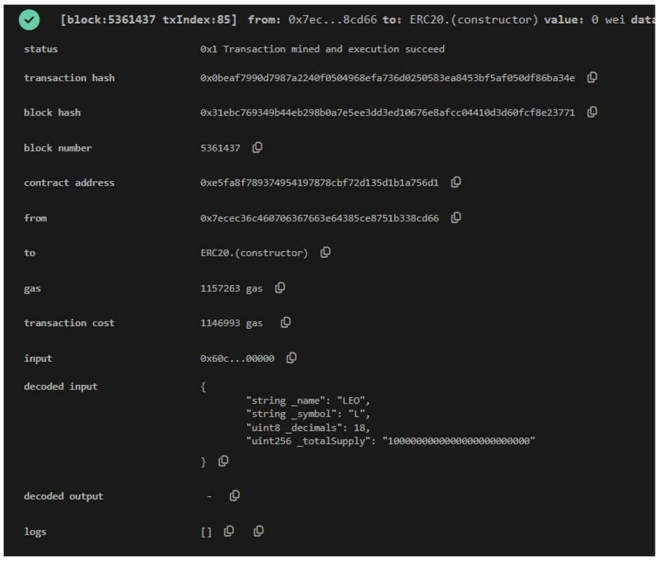
OUTPUT:

1. Connecting with METAMASK

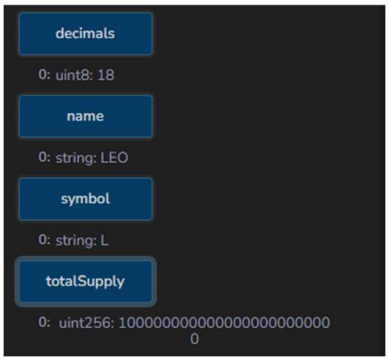


2. Deploying Contract





3. Running Functions



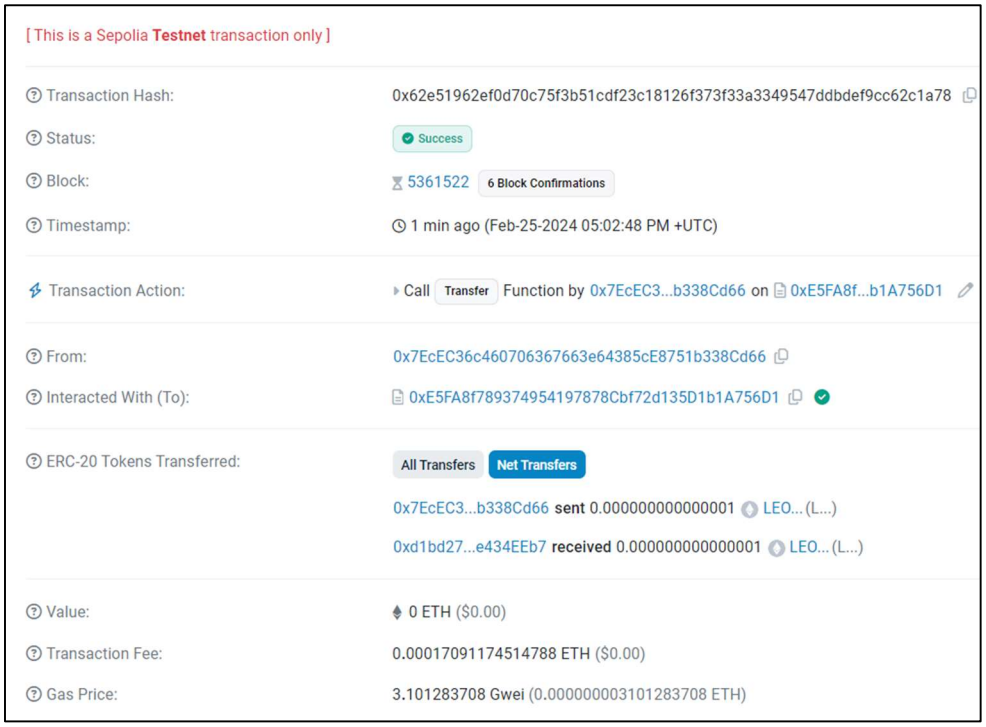


Fig a) Sending 1000 “LEO” tokens [Transfer]

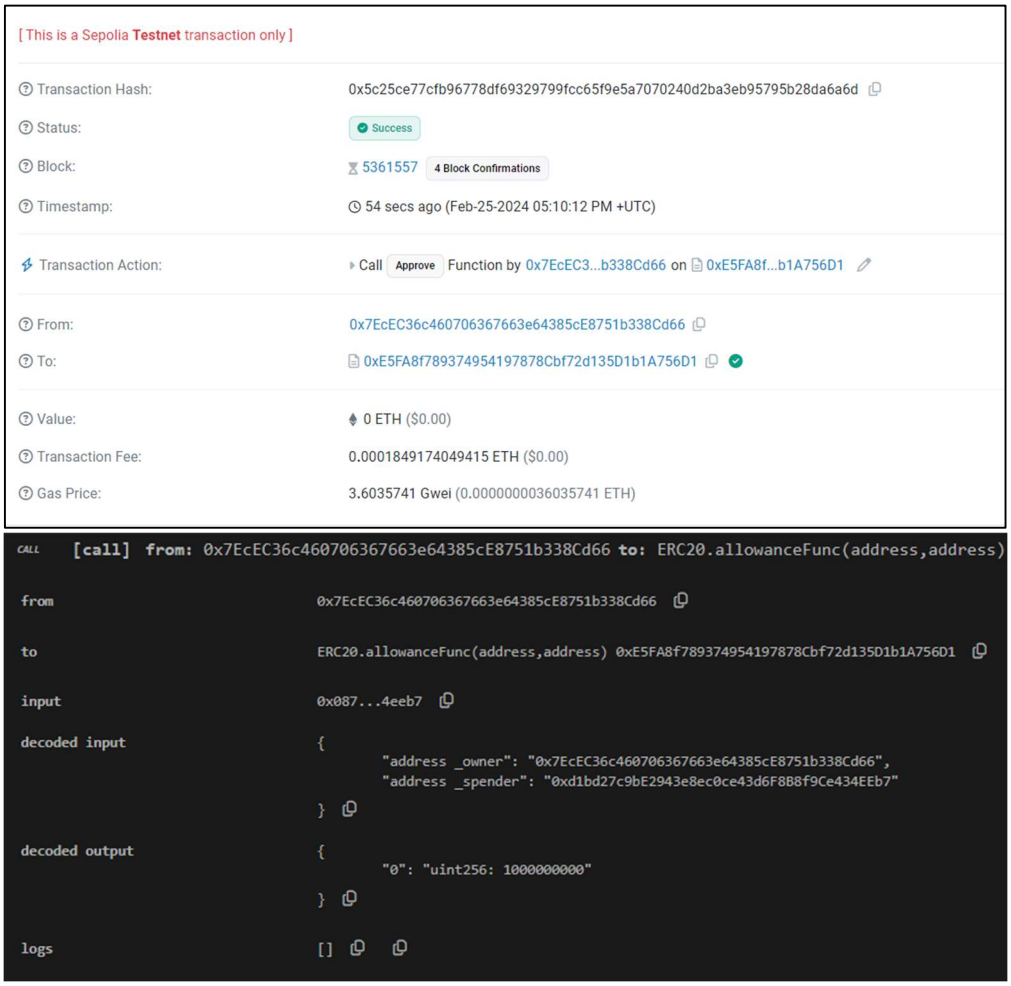


Fig b) Giving allowance to another account [Approve] & displaying the allowance value [AllowanceFunc]