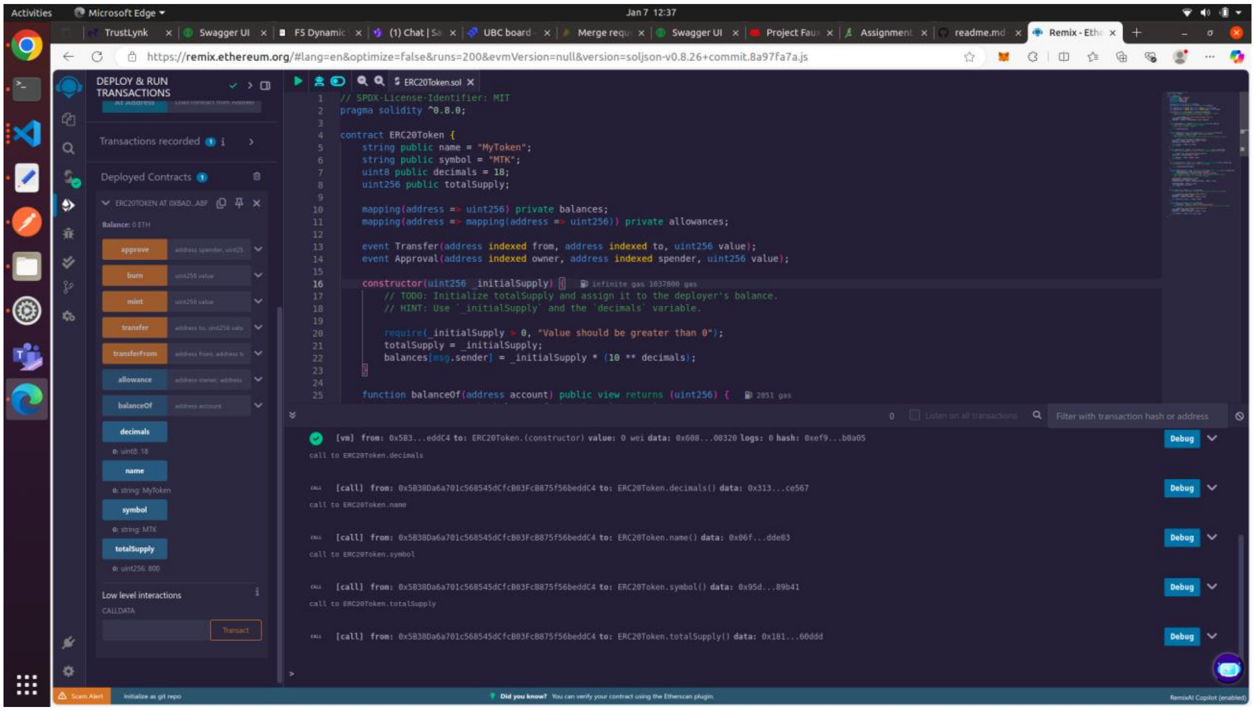
<u>SNAPSHOTS</u> (Student Name: Jagadeeswar Goud (NPCI))

1. **Deploy Contract:** Contract deployed, and initial supply is 800 Ethers. Also, you can view the balance, totalSupply, name and symbol.



Balance: 0 ETH

decimals — 0: uint8: 18

name — 0: string: MyToken

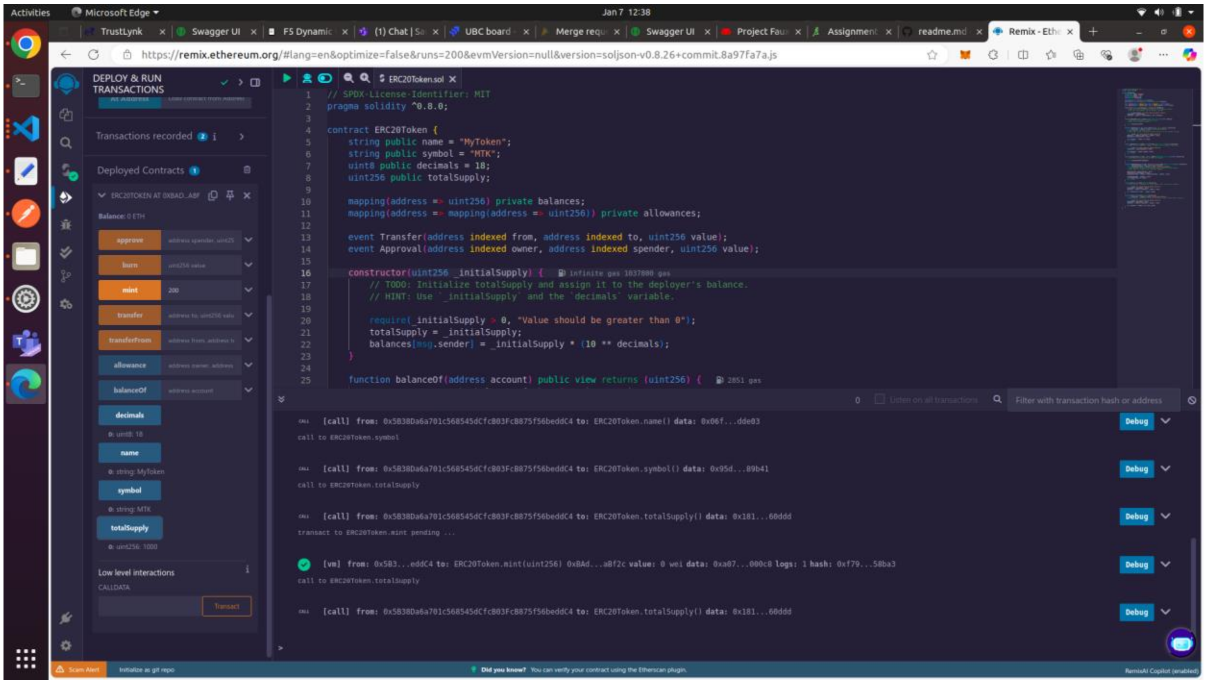
symbol — 0: string: MTK

totalSupply — 0: uint256: 800

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

contract ERC20Token {
    string public name = "MyToken";
    string public symbol = "MTK";
    uint8 public decimals = 18;
    uint256 public totalSupply;

    mapping(address => uint256) private balances;
    mapping(address => mapping(address => uint256)) private allowances;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    constructor(uint256 _initialSupply) {   infinite gas 1037800 gas
        // TODO: Initialize totalSupply and assign it to the deployer's balance.
        // HINT: Use `_initialSupply` and the `decimals` variable.

        require(_initialSupply > 0, "Value should be greater than 0");
        totalSupply = _initialSupply;
        balances[msg.sender] = _initialSupply * (10 ** decimals);
    }

    function balanceOf(address account) public view returns (uint256) {   2851 gas
```

```
[vm] from: 0x5B3...eddC4 to: ERC20Token.(constructor) value: 0 wei data: 0x608...00320 logs: 0 hash: 0xef9...b0a05
call to ERC20Token.decimals

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.decimals() data: 0x313...ce567
call to ERC20Token.name

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.name() data: 0x06f...dde03
call to ERC20Token.symbol

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.symbol() data: 0x95d...89b41
call to ERC20Token.totalSupply

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd
```

2. **Mint:** Minting 200 and the totalSuppy is 1000 Now.



Balance: 0 ETH

mint: 200

decimals — 0: uint8: 18

name — 0: string: MyToken

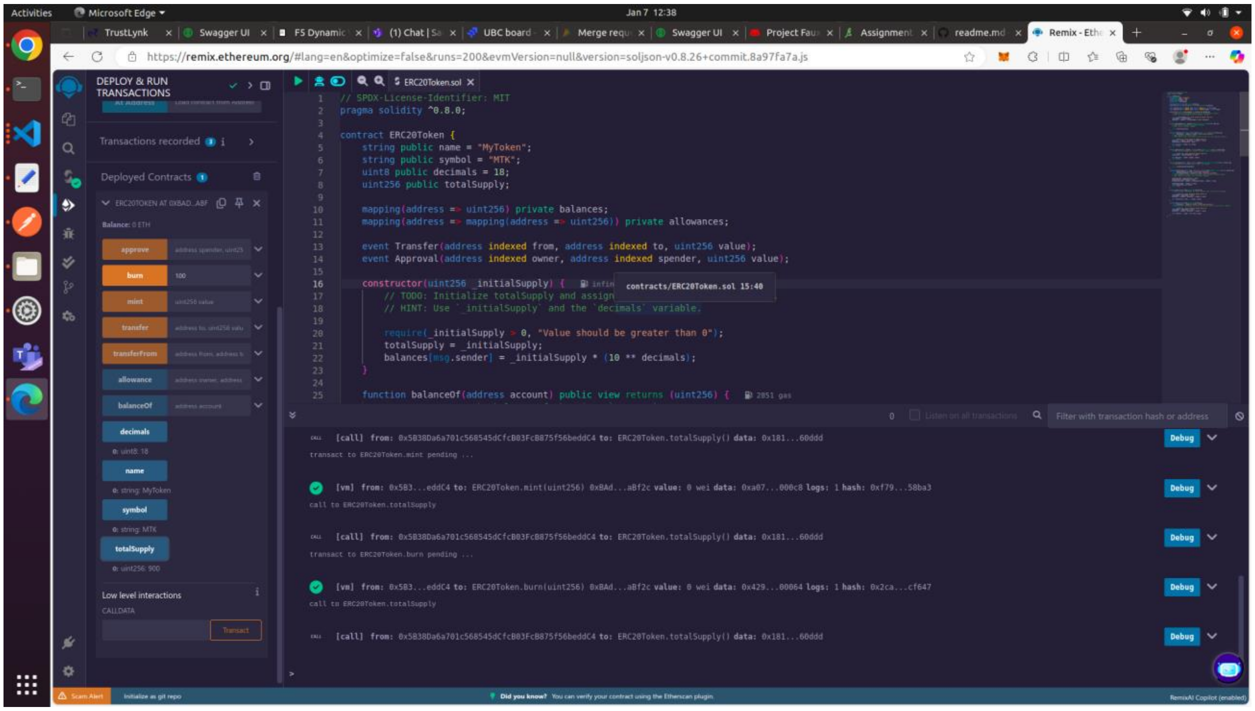
symbol — 0: string: MTK

totalSupply — 0: uint256: 1000

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

contract ERC20Token {
    string public name = "MyToken";
    string public symbol = "MTK";
    uint8 public decimals = 18;
    uint256 public totalSupply;

    mapping(address => uint256) private balances;
    mapping(address => mapping(address => uint256)) private allowances;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    constructor(uint256 _initialSupply) {   infinite gas 1037800 gas
        // TODO: Initialize totalSupply and assign it to the deployer's balance.
        // HINT: Use `_initialSupply` and the `decimals` variable.

        require(_initialSupply > 0, "Value should be greater than 0");
        totalSupply = _initialSupply;
        balances[msg.sender] = _initialSupply * (10 ** decimals);
    }

    function balanceOf(address account) public view returns (uint256) {   2851 gas
```

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.name() data: 0x06f...dde03
call to ERC20Token.symbol

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.symbol() data: 0x95d...89b41
call to ERC20Token.totalSupply

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd
transact to ERC20Token.mint pending ...

[vm] from: 0x5B3...eddC4 to: ERC20Token.mint(uint256) 0xBAd...aBf2c value: 0 wei data: 0xa07...000c8 logs: 1 hash: 0xf79...58ba3
call to ERC20Token.totalSupply

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd
```

3. **Burn:** Burnt 100 and now the totalSuppy is 900 Now.



https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js

Balance: 0 ETH

- approve: address spender, uint25
- burn: 100
- mint: uint256 value
- transfer: address to, uint256 valu
- transferFrom: address from, address t
- allowance: address owner, address
- balanceOf: address account
- decimals — 0: uint8: 18
- name — 0: string: MyToken
- symbol — 0: string: MTK
- totalSupply — 0: uint256: 900

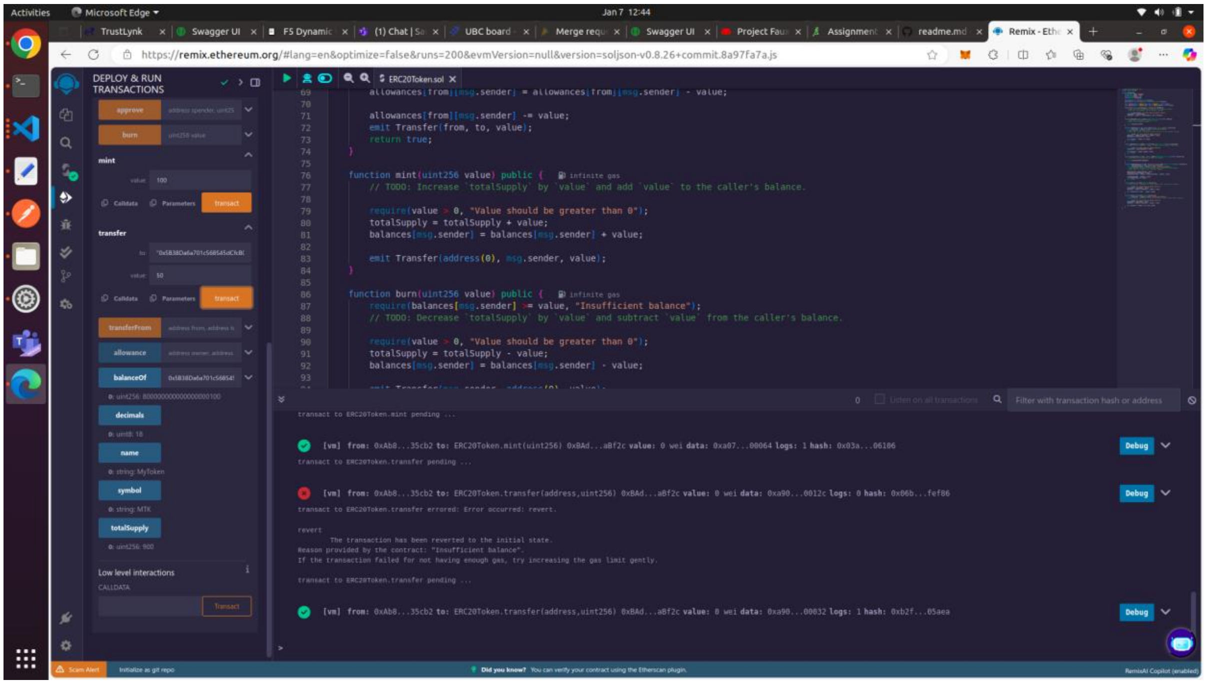
Low level interactions
CALLDATA
Transact

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

contract ERC20Token {
    string public name = "MyToken";
    string public symbol = "MTK";
    uint8 public decimals = 18;
    uint256 public totalSupply;

    mapping(address => uint256) private balances;
    mapping(address => mapping(address => uint256)) private allowances;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    constructor(uint256 _initialSupply) {    infin   contracts/ERC20Token.sol 15:40
        // TODO: Initialize totalSupply and assign
        // HINT: Use `_initialSupply` and the `decimals` variable.

        require(_initialSupply > 0, "Value should be greater than 0");
        totalSupply = _initialSupply;
        balances[msg.sender] = _initialSupply * (10 ** decimals);
    }

    function balanceOf(address account) public view returns (uint256) {    2851 gas
```

```
0    Listen on all transactions    Filter with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd    Debug
transact to ERC20Token.mint pending ...

✓ [vm] from: 0x5B3...eddC4 to: ERC20Token.mint(uint256) 0xBAd...aBf2c value: 0 wei data: 0xa07...000c8 logs: 1 hash: 0xf79...58ba3    Debug
call to ERC20Token.totalSupply

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd    Debug
transact to ERC20Token.burn pending ...

✓ [vm] from: 0x5B3...eddC4 to: ERC20Token.burn(uint256) 0xBAd...aBf2c value: 0 wei data: 0x429...00064 logs: 1 hash: 0x2ca...cf647    Debug
call to ERC20Token.totalSupply

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd    Debug

>
```

Did you know? You can verify your contract using the Etherscan plugin.

4. **Transfer:** Transfer when balance is sufficient.



https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js

- approve: address spender, uint25
- burn: uint256 value
- mint — value: 100 — Calldata — Parameters — transact
- transfer — to: "0x5B38Da6a701c568545dCfcB(— value: 50 — Calldata — Parameters — transact
- transferFrom: address from, address t
- allowance: address owner, address
- balanceOf: 0x5B38Da6a701c56854! — 0: uint256: 800000000000000000100
- decimals — 0: uint8: 18
- name — 0: string: MyToken
- symbol — 0: string: MTK
- totalSupply — 0: uint256: 900

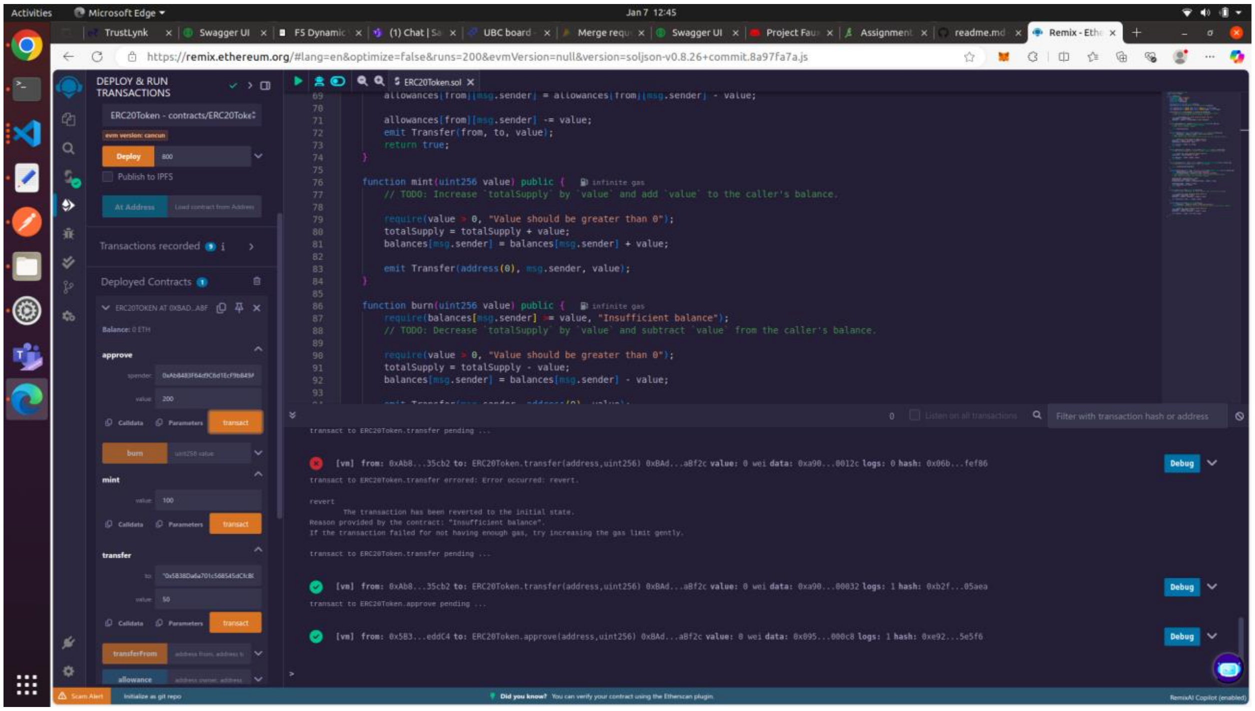
Low level interactions
CALLDATA
Transact

```
        allowances[from][msg.sender] -= value;
        emit Transfer(from, to, value);
        return true;
    }

    function mint(uint256 value) public {    infinite gas
        // TODO: Increase `totalSupply` by `value` and add `value` to the caller's balance.

        require(value > 0, "Value should be greater than 0");
        totalSupply = totalSupply + value;
        balances[msg.sender] = balances[msg.sender] + value;

        emit Transfer(address(0), msg.sender, value);
    }

    function burn(uint256 value) public {    infinite gas
        require(balances[msg.sender] >= value, "Insufficient balance");
        // TODO: Decrease `totalSupply` by `value` and subtract `value` from the caller's balance.

        require(value > 0, "Value should be greater than 0");
        totalSupply = totalSupply - value;
        balances[msg.sender] = balances[msg.sender] - value;
```

```
0    Listen on all transactions    Filter with transaction hash or address

transact to ERC20Token.mint pending ...

✓ [vm] from: 0xAb8...35cb2 to: ERC20Token.mint(uint256) 0xBAd...aBf2c value: 0 wei data: 0xa07...00064 logs: 1 hash: 0x03a...06106    Debug
transact to ERC20Token.transfer pending ...

✗ [vm] from: 0xAb8...35cb2 to: ERC20Token.transfer(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0xa90...0012c logs: 0 hash: 0x06b...fef86    Debug
transact to ERC20Token.transfer errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Insufficient balance".
If the transaction failed for not having enough gas, try increasing the gas limit gently.

transact to ERC20Token.transfer pending ...

✓ [vm] from: 0xAb8...35cb2 to: ERC20Token.transfer(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0xa90...00032 logs: 1 hash: 0xb2f...05aea    Debug

>
```

Did you know? You can verify your contract using the Etherscan plugin.

5. **Allowance and Approve:** Adding allowance and approve.



Deploy: 800

approve — spender: 0xAb8483F64d9C6d1EcF9b849A, value: 200

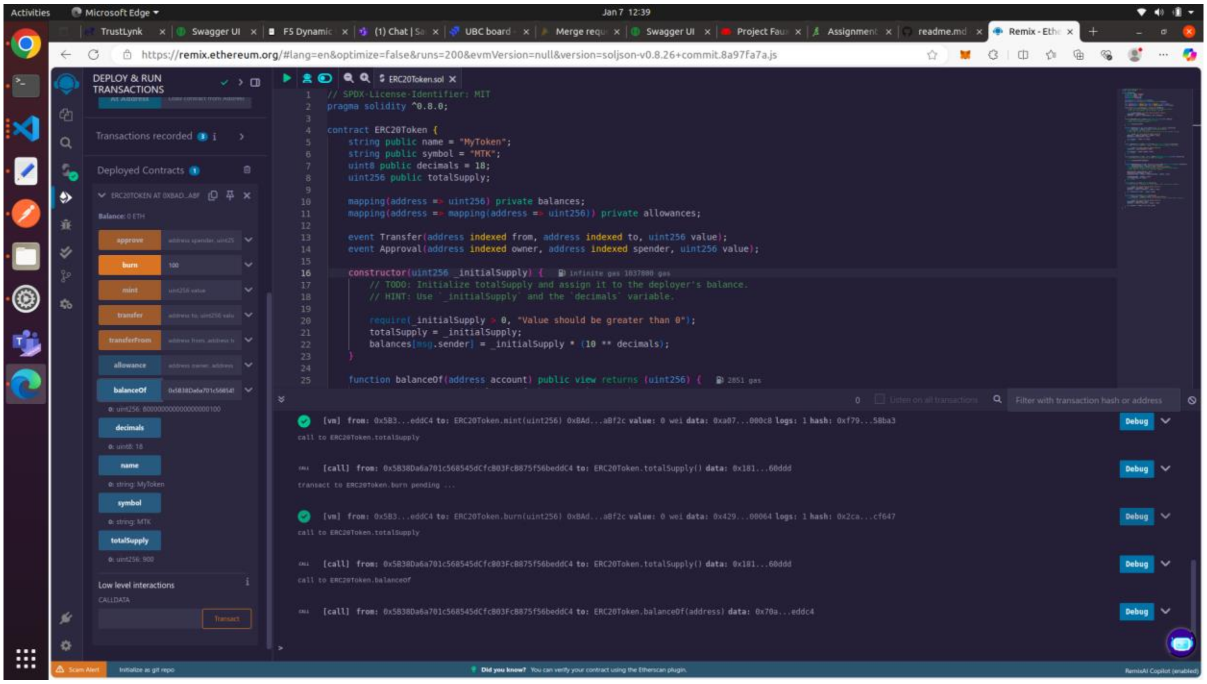
mint — value: 100

transfer — to: "0x5B38Da6a701c568545dCfcB0, value: 50

```
        allowances[from][msg.sender] = allowances[from][msg.sender] - value;

        allowances[from][msg.sender] -= value;
        emit Transfer(from, to, value);
        return true;
    }

    function mint(uint256 value) public {    infinite gas
        // TODO: Increase `totalSupply` by `value` and add `value` to the caller's balance.

        require(value > 0, "Value should be greater than 0");
        totalSupply = totalSupply + value;
        balances[msg.sender] = balances[msg.sender] + value;

        emit Transfer(address(0), msg.sender, value);
    }

    function burn(uint256 value) public {    infinite gas
        require(balances[msg.sender] >= value, "Insufficient balance");
        // TODO: Decrease `totalSupply` by `value` and subtract `value` from the caller's balance.

        require(value > 0, "Value should be greater than 0");
        totalSupply = totalSupply - value;
        balances[msg.sender] = balances[msg.sender] - value;

```

```
transact to ERC20Token.transfer pending ...

[vm] from: 0xAb8...35cb2 to: ERC20Token.transfer(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0xa90...0012c logs: 0 hash: 0x06b...fef86
transact to ERC20Token.transfer errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Insufficient balance".
If the transaction failed for not having enough gas, try increasing the gas limit gently.

transact to ERC20Token.transfer pending ...

[vm] from: 0xAb8...35cb2 to: ERC20Token.transfer(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0xa90...00032 logs: 1 hash: 0xb2f...05aea
transact to ERC20Token.approve pending ...

[vm] from: 0x5B3...eddC4 to: ERC20Token.approve(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0x095...000c8 logs: 1 hash: 0xe92...5e5f6
```

6. **Balance:** Fetching balance for an account.



burn: 100

balanceOf: 0x5B38Da6a701c568545d
0: uint256: 800000000000000000100

decimals — 0: uint8: 18

name — 0: string: MyToken

symbol — 0: string: MTK

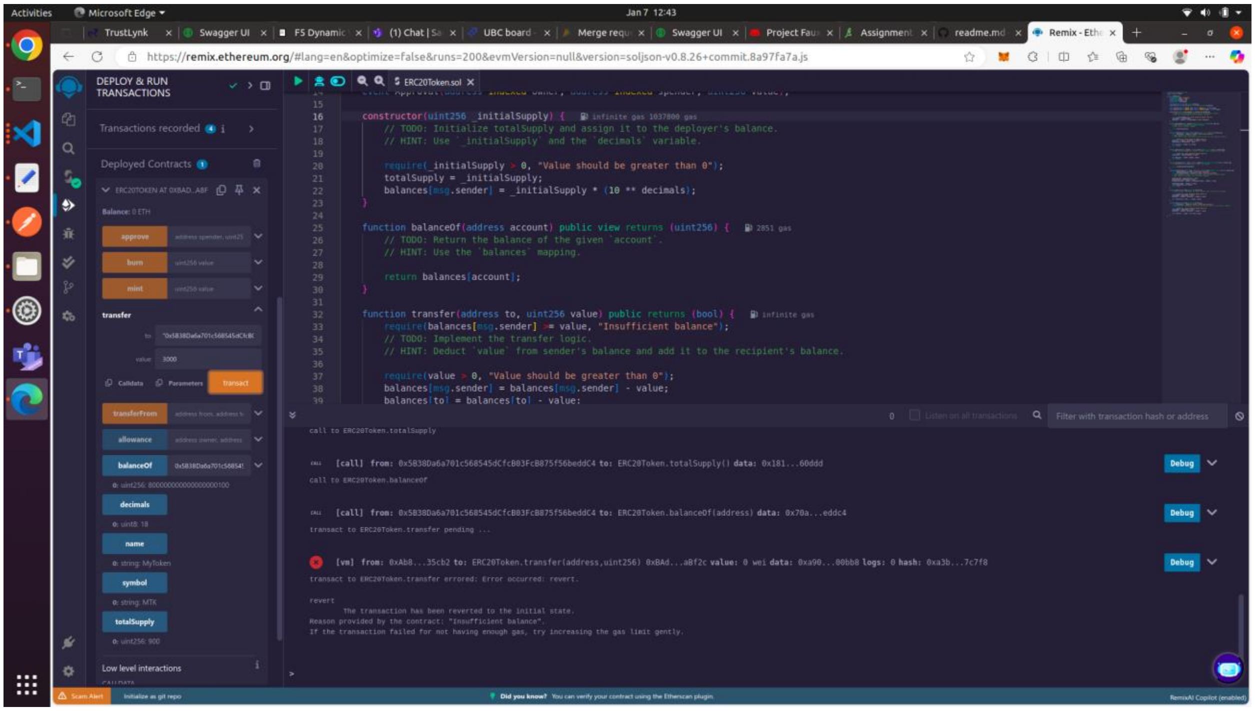
totalSupply — 0: uint256: 900

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

contract ERC20Token {
    string public name = "MyToken";
    string public symbol = "MTK";
    uint8 public decimals = 18;
    uint256 public totalSupply;

    mapping(address => uint256) private balances;
    mapping(address => mapping(address => uint256)) private allowances;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    constructor(uint256 _initialSupply) {    infinite gas 1037800 gas
        // TODO: Initialize totalSupply and assign it to the deployer's balance.
        // HINT: Use `_initialSupply` and the `decimals` variable.

        require(_initialSupply > 0, "Value should be greater than 0");
        totalSupply = _initialSupply;
        balances[msg.sender] = _initialSupply * (10 ** decimals);
    }

    function balanceOf(address account) public view returns (uint256) {    2851 gas
```

```
[vm] from: 0x5B3...eddC4 to: ERC20Token.mint(uint256) 0xBAd...aBf2c value: 0 wei data: 0xa07...000c8 logs: 1 hash: 0xf79...58ba3
call to ERC20Token.totalSupply

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd
transact to ERC20Token.burn pending ...

[vm] from: 0x5B3...eddC4 to: ERC20Token.burn(uint256) 0xBAd...aBf2c value: 0 wei data: 0x429...00064 logs: 1 hash: 0x2ca...cf647
call to ERC20Token.totalSupply

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd
call to ERC20Token.balanceOf

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.balanceOf(address) data: 0x70a...eddc4
```

ERROR Cases:

1. **Transfer:** Insufficient balance



DEPLOY & RUN TRANSACTIONS

Transactions recorded 4

Deployed Contracts 1

ERC20TOKEN AT 0XBAD...A8F

Balance: 0 ETH

transfer

to: "0x5B38Da6a701c568545dCfcB0

value: 3000

balanceOf: 0x5B38Da6a701c568545

0: uint256: 8000000000000000000100

decimals

0: uint8: 18

name

0: string: MyToken

symbol

0: string: MTK

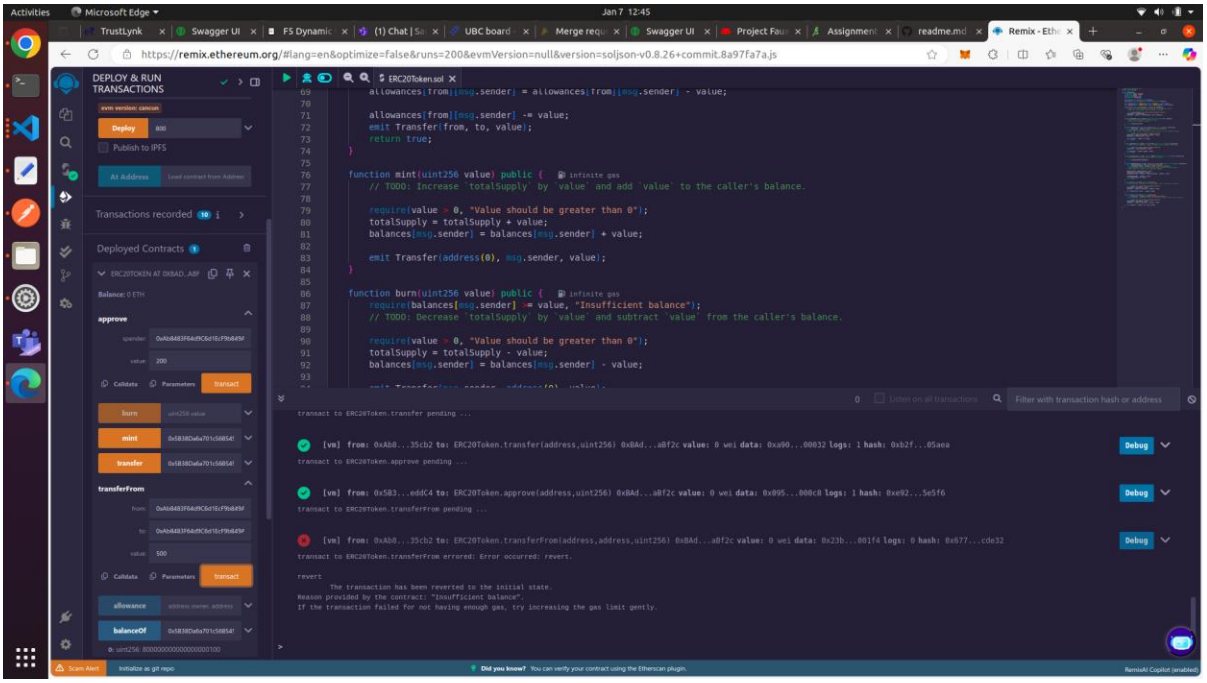
totalSupply

0: uint256: 900

```
    event Approval(address indexed owner, address indexed spender, uint256 value);

    constructor(uint256 _initialSupply) {
        // TODO: Initialize totalSupply and assign it to the deployer's balance.
        // HINT: Use `_initialSupply` and the `decimals` variable.

        require(_initialSupply > 0, "Value should be greater than 0");
        totalSupply = _initialSupply;
        balances[msg.sender] = _initialSupply * (10 ** decimals);
    }

    function balanceOf(address account) public view returns (uint256) {
        // TODO: Return the balance of the given `account`.
        // HINT: Use the `balances` mapping.

        return balances[account];
    }

    function transfer(address to, uint256 value) public returns (bool) {
        require(balances[msg.sender] >= value, "Insufficient balance");
        // TODO: Implement the transfer logic.
        // HINT: Deduct `value` from sender's balance and add it to the recipient's balance.

        require(value > 0, "Value should be greater than 0");
        balances[msg.sender] = balances[msg.sender] - value;
        balances[to] = balances[to] - value;
```

```
call to ERC20Token.totalSupply

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.totalSupply() data: 0x181...60ddd
call to ERC20Token.balanceOf

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ERC20Token.balanceOf(address) data: 0x70a...eddc4
transact to ERC20Token.transfer pending ...

[vm] from: 0xAb8...35cb2 to: ERC20Token.transfer(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0xa90...00bb8 logs: 0 hash: 0xa3b...7c7f8
transact to ERC20Token.transfer errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Insufficient balance".
If the transaction failed for not having enough gas, try increasing the gas limit gently.
```

2. **TransferFrom:** Insufficient Balance



Deploy & Run Transactions, ERC20TOKEN AT 0XBAD...ABF, Balance: 0 ETH

approve: spender: 0xAb8483F64d9C6d1EcF9b849A, value: 200

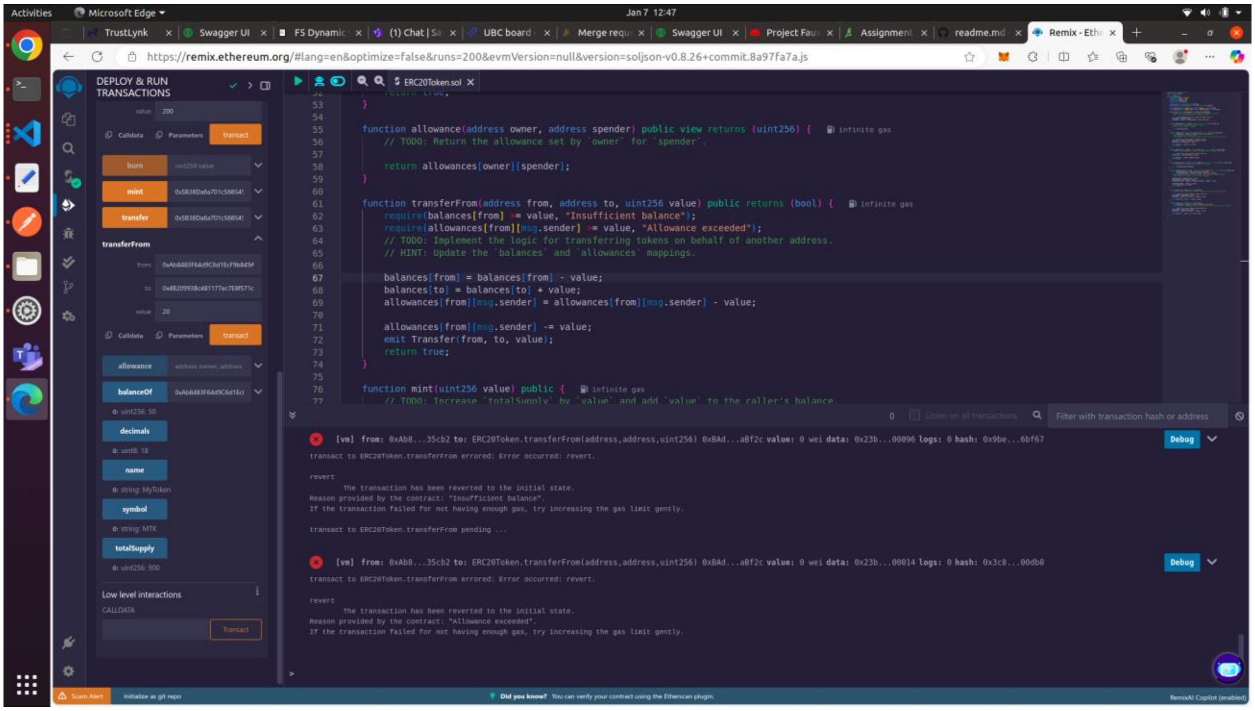
transferFrom: from: 0xAb8483F64d9C6d1EcF9b849A, to: 0xAb8483F64d9C6d1EcF9b849A, value: 500

balanceOf 0x5B38Da6a701c568545: 0: uint256: 800000000000000000100

```
        allowances[from][msg.sender] = allowances[from][msg.sender] - value;

        allowances[from][msg.sender] -= value;
        emit Transfer(from, to, value);
        return true;
    }

    function mint(uint256 value) public {
        // TODO: Increase `totalSupply` by `value` and add `value` to the caller's balance.

        require(value > 0, "Value should be greater than 0");
        totalSupply = totalSupply + value;
        balances[msg.sender] = balances[msg.sender] + value;

        emit Transfer(address(0), msg.sender, value);
    }

    function burn(uint256 value) public {
        require(balances[msg.sender] >= value, "Insufficient balance");
        // TODO: Decrease `totalSupply` by `value` and subtract `value` from the caller's balance.

        require(value > 0, "Value should be greater than 0");
        totalSupply = totalSupply - value;
        balances[msg.sender] = balances[msg.sender] - value;

        emit Transfer(msg.sender, address(0), value);
```

```
transact to ERC20Token.transfer pending ...

[vm] from: 0xAb8...35cb2 to: ERC20Token.transfer(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0xa90...00032 logs: 1 hash: 0xb2f...05aea
transact to ERC20Token.approve pending ...

[vm] from: 0x5B3...eddC4 to: ERC20Token.approve(address,uint256) 0xBAd...aBf2c value: 0 wei data: 0x095...000c8 logs: 1 hash: 0xe92...5e5f6
transact to ERC20Token.transferFrom pending ...

[vm] from: 0xAb8...35cb2 to: ERC20Token.transferFrom(address,address,uint256) 0xBAd...aBf2c value: 0 wei data: 0x23b...001f4 logs: 0 hash: 0x677...cde32
transact to ERC20Token.transferFrom errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Insufficient balance".
If the transaction failed for not having enough gas, try increasing the gas limit gently.
```

3. **Allowance:** Allowance exceeded.



Deploy & Run Transactions

approve: value: 200

transferFrom: from: 0xAb8483F64d9C6d1EcF9b849A, to: 0x4B20993Bc481177ec7E8f571c, value: 20

balanceOf 0xAb8483F64d9C6d1Ec: 0: uint256: 50

decimals: 0: uint8: 18

name: 0: string: MyToken

symbol: 0: string: MTK

totalSupply: 0: uint256: 900

```
    }

    function allowance(address owner, address spender) public view returns (uint256) {
        // TODO: Return the allowance set by `owner` for `spender`.

        return allowances[owner][spender];
    }

    function transferFrom(address from, address to, uint256 value) public returns (bool) {
        require(balances[from] >= value, "Insufficient balance");
        require(allowances[from][msg.sender] >= value, "Allowance exceeded");
        // TODO: Implement the logic for transferring tokens on behalf of another address.
        // HINT: Update the `balances` and `allowances` mappings.

        balances[from] = balances[from] - value;
        balances[to] = balances[to] + value;
        allowances[from][msg.sender] = allowances[from][msg.sender] - value;

        allowances[from][msg.sender] -= value;
        emit Transfer(from, to, value);
        return true;
    }

    function mint(uint256 value) public {
        // TODO: Increase `totalSupply` by `value` and add `value` to the caller's balance.
```

```
[vm] from: 0xAb8...35cb2 to: ERC20Token.transferFrom(address,address,uint256) 0xBAd...aBf2c value: 0 wei data: 0x23b...00096 logs: 0 hash: 0x9be...6bf67
transact to ERC20Token.transferFrom errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Insufficient balance".
If the transaction failed for not having enough gas, try increasing the gas limit gently.

transact to ERC20Token.transferFrom pending ...

[vm] from: 0xAb8...35cb2 to: ERC20Token.transferFrom(address,address,uint256) 0xBAd...aBf2c value: 0 wei data: 0x23b...00014 logs: 0 hash: 0x3c8...00db8
transact to ERC20Token.transferFrom errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Allowance exceeded".
If the transaction failed for not having enough gas, try increasing the gas limit gently.
```