Name of the Experiement : Build a Use Case – Tokenized Supply Chain Prototype

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

1. Start
2. Define stakeholders of the supply chain — Manufacturer, Transporter, Retailer, and Customer.
3. Create a token smart contract to represent product ownership or shipment units.
4. Mint tokens when new products are created by the manufacturer.
5. Transfer tokens at each stage of the supply chain:
   -Manufacturer → Transporter
   -Transporter → Retailer
   -Retailer → Customer
6. Each transfer is stored on the blockchain for transparency and proof of delivery.
7. Tokens represent both ownership and traceability of goods.
8. Verify token balances to confirm product movement.
9. End

* Software used

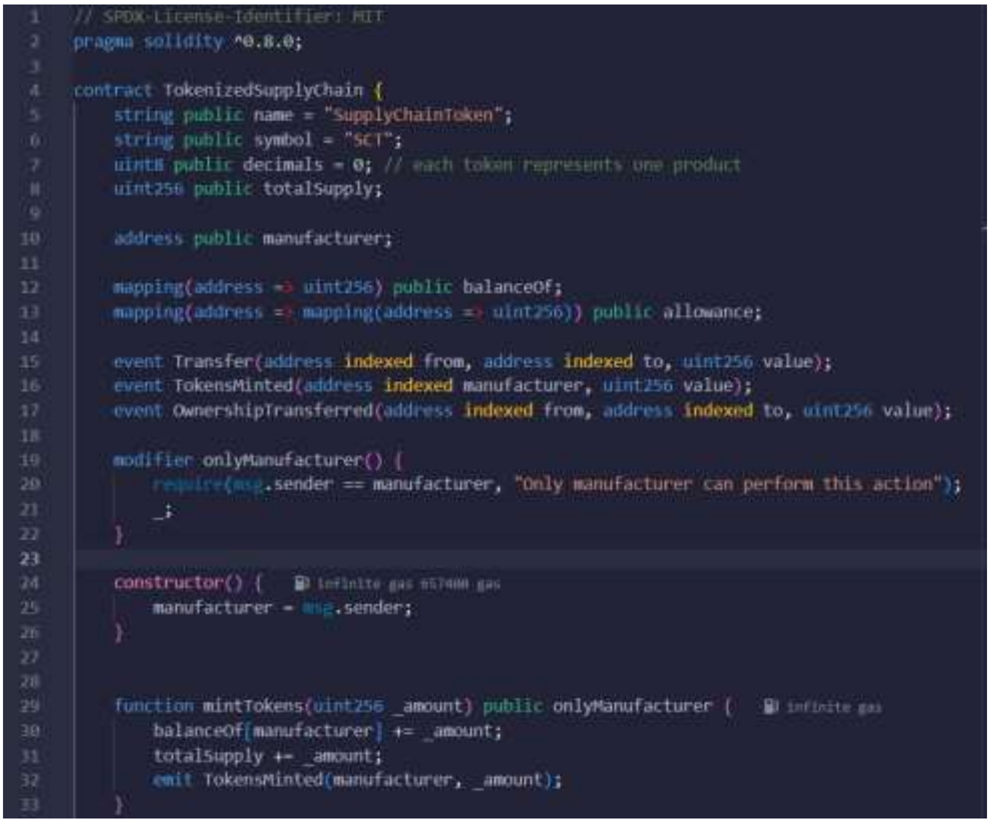
1. Remix IDE
2. MetaMask Wallet
3. Solidity
4. Ethereum Test Network (Sepolia)

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

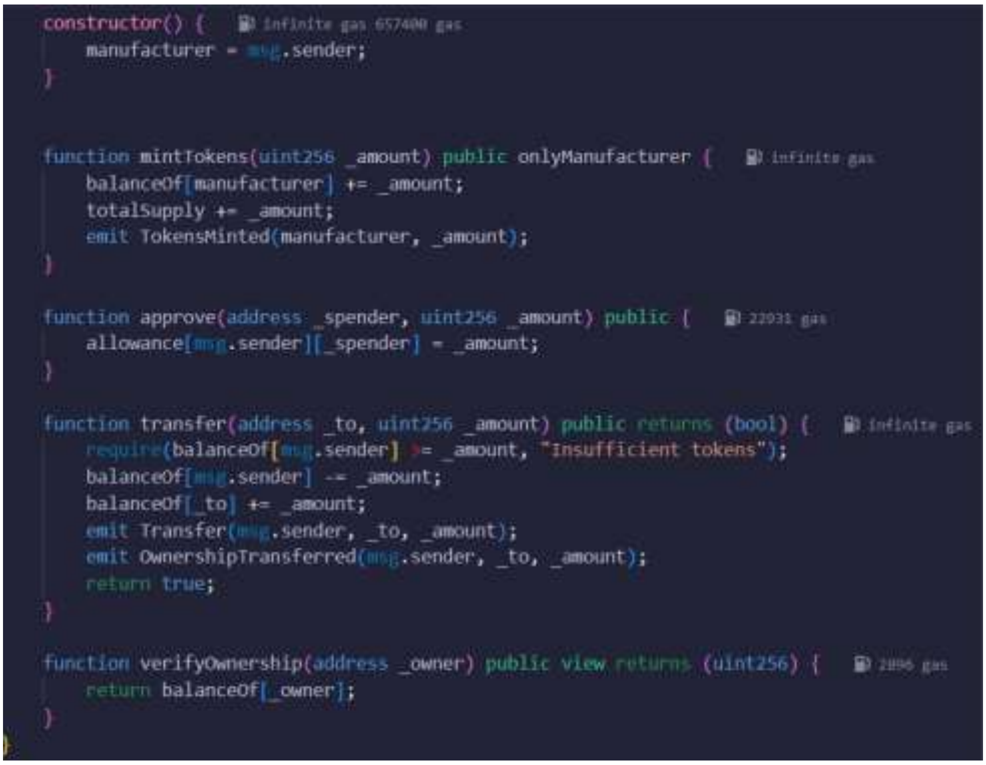
* Testing Phase: Compilation of Code (error detection)

1.Smart Contract Creation

Created a Solidity file TokenizedSupplyChain.sol defining an ERC-20-like token for tracking products.



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

contract TokenizedSupplyChain {
    string public name = "SupplyChainToken";
    string public symbol = "SCT";
    uint8 public decimals = 0; // each token represents one product
    uint256 public totalSupply;

    address public manufacturer;

    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event TokensMinted(address indexed manufacturer, uint256 value);
    event OwnershipTransferred(address indexed from, address indexed to, uint256 value);

    modifier onlyManufacturer() {
        require(msg.sender == manufacturer, "Only manufacturer can perform this action");
        _;
    }

    constructor() {    infinite gas 657400 gas
        manufacturer = msg.sender;
    }


    function mintTokens(uint256 _amount) public onlyManufacturer {    infinite gas
        balanceOf[manufacturer] += _amount;
        totalSupply += _amount;
        emit TokensMinted(manufacturer, _amount);
    }
```



```solidity
    constructor() {    infinite gas 657400 gas
        manufacturer = msg.sender;
    }


    function mintTokens(uint256 _amount) public onlyManufacturer {    infinite gas
        balanceOf[manufacturer] += _amount;
        totalSupply += _amount;
        emit TokensMinted(manufacturer, _amount);
    }

    function approve(address _spender, uint256 _amount) public {    22031 gas
        allowance[msg.sender][_spender] = _amount;
    }

    function transfer(address _to, uint256 _amount) public returns (bool) {    infinite gas
        require(balanceOf[msg.sender] >= _amount, "Insufficient tokens");
        balanceOf[msg.sender] -= _amount;
        balanceOf[_to] += _amount;
        emit Transfer(msg.sender, _to, _amount);
        emit OwnershipTransferred(msg.sender, _to, _amount);
        return true;
    }

    function verifyOwnership(address _owner) public view returns (uint256) {    2896 gas
        return balanceOf[_owner];
    }
}
```

***As applicable according to the experiment.**
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

Test the Functions

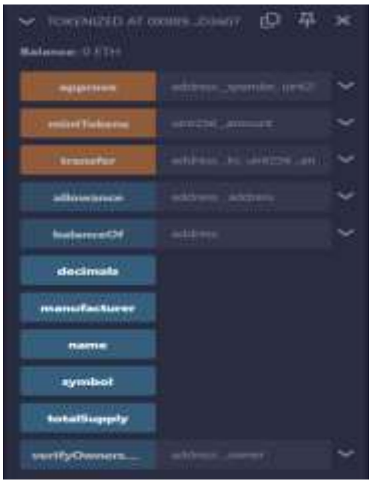
Call mintTokens(100) — manufacturer mints 100 tokens.

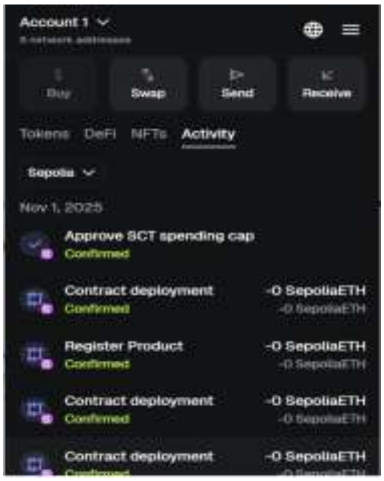
Check balanceOf(manufacturer_address) — verify token balance.

Call transfer(address_of_transporter, 50) — transfer 50 tokens to transporter.

Switch accounts in Remix → call transfer() again to simulate next steps (retailer, customer).

Finally, call verifyOwnership(address) for each participant to check who owns how many tokens.





Nov 1, 2025

Approve SCT spending cap
Confirmed

Contract deployment -0 SepoliaETH
Confirmed -0 SepoliaETH

Register Product -0 SepoliaETH
Confirmed -0 SepoliaETH

Contract deployment -0 SepoliaETH
Confirmed -0 SepoliaETH

Contract deployment -0 SepoliaETH
Confirmed -0 SepoliaETH

* Observations

1. Tokenizing products enables transparent and verifiable transfer of goods in a supply chain.
2. Each blockchain transaction acts as proof of ownership, ensuring authenticity and accountability.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

**As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.