



School: Campus:

Academic Year: Subject Name: Subject Code:

Semester: Program: Branch: Specialization:

Date:

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiment : 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)

* 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.

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract TokenizedSupplyChain {
5     string public name = "SupplyChainToken";
6     string public symbol = "SCT";
7     uint8 public decimals = 0; // each token represents one product
8     uint256 public totalSupply;
9
10    address public manufacturer;
11
12    mapping(address => uint256) public balanceOf;
13    mapping(address => mapping(address => uint256)) public allowance;
14
15    event Transfer(address indexed from, address indexed to, uint256 value);
16    event TokensMinted(address indexed manufacturer, uint256 value);
17    event OwnershipTransferred(address indexed from, address indexed to, uint256 value);
18
19    modifier onlyManufacturer() {
20        require(msg.sender == manufacturer, "Only manufacturer can perform this action");
21        ...
22    }
23
24    constructor() {    █ infinite gas 657400 gas
25        manufacturer = msg.sender;
26    }
27
28
29    function mintTokens(uint256 _amount) public onlyManufacturer {    █ infinite gas
30        balanceOf[manufacturer] += _amount;
31        totalSupply += _amount;
32        emit TokensMinted(manufacturer, _amount);
33    }
34
35
36    constructor() {    █ infinite gas 657400 gas
37        manufacturer = msg.sender;
38    }
39
40
41    function mintTokens(uint256 _amount) public onlyManufacturer {    █ infinite gas
42        balanceOf[manufacturer] += _amount;
43        totalSupply += _amount;
44        emit TokensMinted(manufacturer, _amount);
45    }
46
47    function approve(address _spender, uint256 _amount) public {    █ 22931 gas
48        allowance[msg.sender][_spender] = _amount;
49    }
50
51
52    function transfer(address _to, uint256 _amount) public returns (bool) {    █ infinite gas
53        require(balanceOf[msg.sender] >= _amount, "Insufficient tokens");
54        balanceOf[msg.sender] -= _amount;
55        balanceOf[_to] += _amount;
56        emit Transfer(msg.sender, _to, _amount);
57        emit OwnershipTransferred(msg.sender, _to, _amount);
58        return true;
59    }
60
61    function verifyOwnership(address _owner) public view returns (uint256) {    █ 2896 gas
62        return balanceOf[_owner];
63    }
64}
```

* Implementation Phase: Final Output (no error)

Applied and Action Learning

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.

The screenshot shows the Remix IDE interface for the Tokenized contract. On the left, a sidebar lists various functions: approve, mintTokens, transfer, allowance, balanceOf, decimals, manufacturer, name, symbol, totalSupply, and verifyOwners... The 'mintTokens' function is selected. On the right, the main panel displays the 'Account 1' interface with tabs for Tokens, DeFi, NFTs, and Activity. The Activity tab shows a history of transactions from Nov 1, 2025, including several 'Contract deployment' and 'Approve SCT spending cap' events, all marked as 'Confirmed'. A separate window on the right shows a transaction details screen with a green checkmark icon, indicating a successful transaction. The transaction details are: [block:9538007 txIndex:9] from: 0xe4a...ca52e to: Tokenized.approve(address,uint256) 0x889...d3607 value: 0 wei data: 0x095...00001 logs: 0 hash: 0x57e...6dd82

* 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. :

Page No.....

Signature of the Faculty:

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